Improved evolvability in genetic programming with polyandry

Anisa Ragalo, Nelishia Pillay

School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, South Africa

ABSTRACT

This paper proposes Polyandry, a new nature-inspired modification to canonical Genetic Programming (GP). Polyandry aims to improve evolvability in GP. Evolvability is a critically important GP trait, the maintenance of which determines the arrival of the GP at a global optimum solution. Specifically evolvability is defined as the ability of the genetic operators employed in GP to produce offspring that are fitter than their parents. When GP fails to exhibit evolvability, further adaptation of the GP individuals towards a global optimum solution becomes impossible.

Polyandry improves evolvability by improving the typically disruptive standard GP crossover operator. The algorithm employs a dual strategy towards this goal. The chief part of this strategy is an incorporation of genetic material from multiple mating partners into broods of offspring. Given such a brood, the offspring in the brood then compete according to a culling function, which we make equivalent to the main GP fitness function. Polyandry's incorporation of genetic material from multiple GP individuals into broods of offspring represents a more aggressive search for building block information. This characteristic of the algorithm leads to an advanced explorative capability in both GP genotype space and fitness space. The second component of the Polyandry strategy is an attempt at multiple crossover points, in order to find crossover points that minimize building block disruption from parents to offspring. This strategy is employed by a similar algorithm, Brood Recombination.

We conduct experiments to compare Polyandry with the canonical GP. Our experiments demonstrate that Polyandry consistently exhibits better evolvability than the canonical GP. As a consequence, Polyandry achieves higher success rates and discovers globally optimal solutions in significantly fewer generations than the latter. The result of these observations is that given certain brood size settings, Polyandry requires less computational effort to arrive at a global optimum solution than the canonical GP.

We also conduct experiments to compare Polyandry with the analogous nature-inspired modification to canonical GP, Brood Recombination. The adoption of Brood Recombination in order to improve evolvability is ubiquitous in GP literature. Our results demonstrate that Polyandry consistently exhibits better evolvability than Brood Recombination, due to a more explorative nature of the algorithm in both genotype and fitness space. As a result, although the two algorithms exhibit similar success rates, Polyandry consistently discovers globally optimal solutions in significantly fewer GP generations than Brood Recombination. The key advantage of Polyandry over Brood Recombination is therefore faster solution discovery. As a consequence Polyandry consistently requires less computational effort to arrive at a global optimum solution compared to Brood Recombination.

Further, we establish that the computational effort exerted by Polyandry is competitively low, relative to other Evolutionary Algorithm (EA) methodologies in literature. We conclude that Polyandry is a better alternative to both the canonical GP as well as Brood Recombination with regards to the achievement and maintenance of evolvability.

KEYWORDS: genetic programming, evolvability

CATEGORIES: I.2

1 INTRODUCTION

Genetic Programming (GP) is a soft computing technique that borrows from the principles of variation and natural selection in nature [1]. The GP methodology searches a large program space by iteratively selecting and applying variation operators to members of a population. Here the members of the population to which variation is applied are selected according to a fitness function.

The iterative selection and variation in GP is conducted with the aim of evolving the population to a global optimum solution. The arrival of GP at a global optimum solution depends fundamentally on the ability of the algorithm to maintain evolvability throughout its generations [2] [3]. In GP, evolvability is defined as the ability of employed genetic (or variation) operators to produce offspring that are fitter than their parents [2] [3]. Evolvability is a vital concept underlying the workings of GP. A GP that fails to maintain evolvability cannot adapt successfully in response to selection pressure applied by the fitness function [2].

A formidable obstacle towards the maintenance of evolvability is the disruptiveness of the standard crossover genetic operator [4].[1] Standard crossover

Email: Anisa Ragalo 204505443@stu.ukzn.ac.za, Nelishia Pillay pillayn32@ukzn.ac.za

[1] In this article the terms standard (GP) crossover and canonical GP refer to Koza's crossover operator and Koza's original

predominantly produces offspring that are less fit than their parents [4]. We present Polyandry, a modification to the canonical GP. Polyandry aims to improve the standard GP crossover operator, thus improving evolvability. Similarly to GP, the elementary idea underlying Polyandry hails from an analogy in nature. Polyandry is similar to Brood Recombination [2] [3] [5], an existing methodology to improve crossover.

In nature, organisms may produce several offspring as a means of increasing the chances of offspring survival [6] [7] [8] [9] [10]. A collection of these offspring is referred to as a brood or a clutch. The existing Brood Recombination algorithm [2] [3] [5] is drawn from the analogy of a brood of offspring being produced by the same two parents. Such a brood, produced by sexual reproduction involving meiosis and recombination, is genetically diverse [11]. When two parent organisms create a sizeable brood of offspring as opposed to fewer offspring, multiple diverse combinations of the genetic material from the two parents are attempted in the different offspring. As a result the probability that at least one of the offspring will possess an optimal combination of genetic material, and thus a high fitness, is increased [12]. The probability of fitter offspring is directly linked to the concept of improved evolvability in GP.

Our Polyandry algorithm is based on the analogy in nature of a brood of offspring being produced by a single female parent and multiple mating partners. Although multiple matings between two organisms may produce a high level of genetic diversity in a brood, multiple paternities lead to a much higher level of diversity among brood offspring [6] [10] [13]. The distinct offspring in a Polyandry brood represent attempts at combining different sections of genetic material from the female parent with genetic material from diverse mating partners. Multiple paternities represent a more aggressive search for good genetic material. As a result the probability of fitter offspring in a Polyandry brood is even higher than the same for a brood produced from just two parents [6] [11] [13].

We hypothesize that the advantages of Polyandry in nature should translate to GP. This hypothesis is based on the common fundamental goal of both natural evolution and GP; this being the discovery of good genetic material in response to selective pressure. Based on our hypothesis, we implement a Polyandry algorithm and compare the evolvability yielded with the same for the canonical GP algorithm as well as the Brood Recombination algorithm. We conduct experiments comparing the three mentioned algorithms across four GP benchmark problem domains: the Artificial Ant, Even-5 Parity, Symbolic Regression and 11-Multiplexer domains. The experiments demonstrate that Polyandry consistently exhibits better evolvability than the both the canonical GP and Brood Recombination algorithms. High evolvability is associated with better GP performance [2]. Accordingly, given certain brood size settings, the Polyandry algorithm requires less computational effort to arrive at a global optimum solution compared to the canonical GP.[2] Further, Polyandry consistently requires less computational effort to reach a global optimum solution compared to Brood Recombination. These observations are consistent across all four domains tested. We will see that Polyandry's outperformance of the canonical GP stems from the higher success rate and faster solution discovery exhibited by the former. Further Polyandry's outperformance of Brood Recombination stems from the faster solution discovery of Polyandry relative to Brood Recombination.

This paper is organized as follows: Section 2 provides detailed information on the concepts of evolvability, Brood Recombination and Polyandry, and further argues for an algorithm incorporating Polyandry as a superior alternative to Brood Recombination; section 3 describes the novel algorithm; section 4 describes experiments conducted with the aim of demonstrating the better evolvability achieved by our approach with respect to Brood Recombination; section 5 provides results and a subsequent discussion from experiments conducted; section 6 is a conclusion and discusses possible future extensions to the Polyandry algorithm.

2 BACKGROUND

2.1 Evolvability

In natural evolution, evolvability is defined as the ability of a population to adapt in response to selective pressure [14] [15]. A similar definition from [16] establishes that evolvability is a population's capacity to evolve.

The term evolvability represents a similar concept in GP. In GP evolvability is defined as the ability of a population to adapt in response to the selection pressure applied by a fitness function [2]. Specifically, the evolvability of a GP system is the capacity of the employed representation and genetic operators to produce offspring that are fitter than their parents [2] [3]. According to [2], evolvability is perceived as the most fine-grained measurement of the performance of GP.

Evolvability in GP is influenced by the ability of genetic operators, such as the standard crossover operator, to leave existing highly fit building blocks intact while creating further opportunities for adaptation [2]. The maintenance of highly fit building blocks is paramount to fitness gains in GP. Here building blocks determine the fitness value of an individual [1] [2] [17] [18]. Successful intact building block transmission between parents and offspring during crossover yields a high probability of fitter offspring [19]. This is because the undisrupted building blocks transmitted from each parent can jointly form an extended building block in the offspring. Building block extension is what ultimately leads to fitter offspring [1] [17].

Evolvability is realized in GP if, during the creation of offspring, a genetic operator perturbs the

GP algorithm respectively, described in [1].

[2] We measure computational effort according to a modification of the standard metric proposed by Koza in [1]. This is discussed in more detail in Section 5.1.

less adapted sections of individuals in the population that is sections that do not contain highly fit building blocks as opposed to the highly adapted sections. It follows that the typically disruptive nature of the standard crossover operator is a major obstacle towards evolvability. Standard GP crossover predominantly produces offspring that are less fit than their parents as building blocks residing in the parents are often disrupted during crossover events [1] [2] [4]. This disruption occurs because crossover points are chosen at random and thus are likely to be located within building blocks. It follows that crossover mostly fails to transmit intact building blocks from parents to offspring.

2.2 Brood recombination in nature and the brood recombination algorithm

The Brood Recombination algorithm was introduced in [2] [3] [5], with the aim of improving evolvability. In the parallel phenomenon in nature, an organism produces several offspring by mating multiple times with a single partner. As a result of meiosis and recombination the resulting brood is genetically diverse. Due to this diversity, the distinct offspring possess distinct competitive advantages or disadvantages. As a result of selection pressure only a fraction of the brood survives [9].

The Brood Recombination algorithm is a modification to the canonical GP. Fig. 1 outlines the steps involved in the canonical GP, with added detail on the steps conducted by the standard GP crossover operator. In [1] Koza sets the values of the genetic operator probabilities $p1\%$ and $p2\%$ (Fig. 1) to 90% and 10% respectively. Koza observes that aside from crossover and reproduction, other genetic operators, such as mutation, have no substantive effect on the success rate of GP [1]. The canonical GP in Fig. 1 thus employs only crossover and reproduction. Also in this study, we focus on the standard crossover operator.

The Brood Recombination algorithm performs all of the steps detailed in Fig. 1, with the exception that it replaces the standard crossover operator with a Brood Recombination operator, $P_B(n)$. Here $R_B(n)$ produces 2n offspring from two selected parents in GP. Subsequently only the best offspring from the resulting brood are inserted into the GP population. $R_B(n)$ is parameterized by the brood size factor n and the culling function F_B [20]. The brood size factor determines the standard size of a brood. The offspring in a brood compete according to F_B. F_B may differ from, but is often the same as the fitness function applied to the whole GP population. The steps conducted by $R_B(n)$ are listed in Fig. 2.

We note that in the Brood Recombination algorithm, $R_B(n)$ creates $p1\%$ of $[Generation]_{(n+1)}$ in place of standard crossover (line 4a. Fig. 1). We will abbreviate the Brood Recombination algorithm as BR for the remainder of this text. BR minimizes the disruptiveness of the standard crossover operator by making several attempts at creating fit offspring from

1. Initialise an integer n to represent the current generation ($n = 0$).
2. Create the initial GP population ($Generation_0$).
3. Evaluate the fitness of each individual in $Generation_0$.
4. Repeat
   (a) Create $p1\%$ of $Generation_{n+1}$ by standard crossover. The steps involved in each crossover event are:
      i. Select parents P_1 and P_2 from the population via fitness proportionate or tournament selection.
      ii. Randomly select crossover points (subtrees) S_{i1} and S_{i2} from P_1 and P_2 respectively.
      iii. Exchange subtrees S_{i1} and S_{i2} to form brood offspring C_1 and C_2.
      iv. Insert C_1 and C_2 into $Generation_{n+1}$.
   (b) Create $p2\%$ of $Generation_{n+1}$ by reproduction.
   (c) Calculate the fitness of each individual in $Generation_{n+1}$.
   (d) Increment n by 1.
5. Until a global optimum solution is reached or the maximum number of generations have been run.

Figure 1: Pseudo-code for the canonical GP, adapted from [1]

1. Select parents P_1 and P_2 from the population via fitness proportionate or tournament selection.
2. For $i = 1$ to n perform standard crossover between P_1 and P_2 by:
   (a) Randomly selecting crossover points (subtrees) S_{i1} and S_{i2} from P_1 and P_2 respectively.
   (b) Exchanging subtrees S_{i1} and S_{i2} to form brood offspring B_{i1} and B_{i2}.
3. Determine the fitness value of each one of the $2n$ members of the brood according to F_B.
4. Sort the $2n$ members of the brood in ascending order of fitness according to F_B.
5. Select offspring ("children") C_1 and C_2 which are the fittest of the $2n$ brood members according to F_B.
6. Return C_1 and C_2 as the offspring produced by $R_B(n)$, for insertion into the GP population.

Figure 2: Pseudo-code for Algorithm $R_B(n)$, adapted from [16]

two parents [2] [21] [22] [23]. Specifically, BR makes several attempts at finding suitable crossover points in both parents. Under $[R]_B(n)$, each time standard crossover is applied, random crossover points are selected. Thus we expect the offspring in a brood to be genetically diverse, and the product of diverse crossover points with respect to each other. If the culling function, F_B, is the same as the GP fitness function, we select the fittest offspring of the brood for insertion into the GP population. This fitness criteria leads to the selection of offspring in which there is minimal building block disruption, compared to the remainder of the brood. In Subsection 2.1 we established that fit offspring are associated with minimal building block disruption.

In minimizing crossover disruptiveness, BR improves GP evolvability [2] [3]. Under the algorithm, the offspring returned by $R_B(n)$ are those resulting from crossover operations that are likely to have perturbed sections that do not contain building blocks. As a consequence, crossover is no longer associated with disruption, but rather becomes an operator that is more likely to create opportunities for further building block extension [2].

We however note that BR is more computationally expensive than the canonical GP. Given a brood size $2n$, $R_B(n)$ is n times more computationally expensive than standard crossover. This is because the fitness of each of the $2n$ offspring in a brood has to be evaluated, before only the best 2 offspring are selected.

By contrast, standard crossover creates only 2 offspring, both of which are incorporated into the GP population. In literature this problem is resolved by using a cheaper culling function to evaluate the offspring in a brood [21] [23]. For example, in [21], Tackett uses a small fraction of the total number of fitness cases in the main GP fitness function to evaluate competing offspring in a brood.

In [22] it is established that generally a BR GP with a larger brood size finds solutions faster than a BR GP with a smaller brood size. The theory behind this observation is that larger brood sizes correspond to more attempts at finding suitable crossover points and thus lower chances of building block disruption from parents to offspring [22]. According to [22], there however exists a threshold brood size beyond which further increases in brood size do not yield further improvements in algorithm performance. This threshold value is referred to as the brood-diversity point.

Why does a brood-diversity point exist, limiting the correlation between brood size and performance increases? In GP, the total number of distinct possible crossover points in an individual is limited by the size of the individual. Specifically, an individual with m nodes has a total of only m distinct crossover points. Thus beyond a certain brood size, the crossover points attempted by $[R]_B(n)$ will begin to be duplicated [22]. Beyond the brood-diversity point, increases in brood size are not associated with further exploration of unique crossover points. This explains a general lack of performance gains for brood sizes exceeding the threshold.

Zhang et al. conduct experiments to test the efficacy of BR in object classification problems in [22]. The experiments demonstrate a decrease in the classification accuracy of BR for brood sizes beyond the brood-diversity point. Here classification accuracy is measured as the proportion of images in a training set that are correctly classified by a GP. Although the authors do not provide a reason for the decrease in classification accuracy beyond the brood diversity point, we discuss a possible theory. For brood sizes higher than the threshold point, duplicate crossover points attempted by $[R]_B(n)$ should lead to duplicate offspring in a brood. As a result of duplications, the pair of fittest offspring in the brood returned by $[R]_B(n)$ may be identical. This lack of diversity in the offspring returned to the evolving population may contribute to premature convergence.

Some limitations to the level of evolvability achieved by BR emerge. As a GP progresses, larger building blocks are formed [1]. It therefore becomes more difficult to create offspring that are fitter than their parents [22]. This is because most crossover points will be likely to reside within building blocks. Thus if the brood size selected for a BR GP is too small, it translates to an insufficient number of attempts at suitable crossover points in the later stages of the algorithm when larger building blocks exist. Building block disruption may therefore still occur. We observe that because BR attempts to achieve evolvability solely by the trial of multiple crossover points, the maintenance of evolvability becomes difficult in the later stages of the algorithm. We may opt to initialize a BR GP with a large brood size in an attempt to exhaust possible crossover points in the later stages of GP. However due to the existence of a brood diversity point, this may also be to the detriment of the performance of the GP.

We introduce a method inspired by Polyandry in nature. The method employs an aggressive search for good building block material, in order to improve evolvability. Further, similar to Brood Recombination, Polyandry attempts multiple crossover points during the production of offspring. This leads to minimized building block disruption from parents to offspring.

2.3 Polyandry in nature

In nature, Polyandry is defined as the tendency of a female individual to mate with multiple partners within her species, leading to a high level of genetic diversity in the offspring belonging to her clutch [6]. Polyandrous females directly solicit copulations from multiple males [7]. It is argued that this behavior has evolved due to the genetic benefits associated with Polyandry [24]. Multiple paternities lead to a much higher level of diversity among offspring belonging to a female's clutch [6] [10] [13]. The offspring of a multiply mated family are more than twice as diverse as those from a singly mated family [11]. In literature it is hypothesized that the key aim of Polyandrous behavior is the creation of this useful genetic diversity [6] [7] [24] [25].

Polyandrous behaviour is perceived as a bet hedging strategy [26] and pays homage to the adage that advises not to "put all of one's eggs in the same basket". In this context, bet hedging is defined as a strategy where the risk of failure (that is producing less fit offspring) is reduced by increasing the fitness diversity among the offspring in the brood [26]. The increased fitness diversity among the offspring in a brood is a consequence of an increased genetic diversity of the same, the latter being due to multiple paternities of the offspring. In creating a highly fitness diverse brood, the chance that at least some of the offspring produced will have a high fitness is increased.

When a female mates with multiple partners, she reduces the chances of all her eggs being fertilized by a poor quality male [6]. Mating with multiple partners represents a more aggressive search for good genetic material. Since the resulting offspring in the female's clutch bear multiple paternities, the likelihood of at least some of the offspring having being sired by a relatively fit male is higher [6] [11] [13]. The fraction of offspring sired by fitter males is in turn likely to be fit and thus these offspring have a higher chance of survival. Here the diversity generated by Polyandry serves to find fitter genetic material and thus increase the viability of at least some of the offspring generated.

We see therefore that Polyandry in nature is associated with a higher likelihood of fitter offspring. We hypothesize that this can translate to a computational GP model.

2.4 A polyandry algorithm as an alternative to the brood recombination algorithm

In Subsection 2.3, we established that in nature, a brood that is the product of multiple paternities will possess a higher level of diversity than a brood that is the product of single paternity. We seek to investigate the advantages of an algorithm incorporating Polyandry over an algorithm incorporating Brood Recombination. The Polyandry algorithm (described in Section 3) is identical to BR with the exception that each brood produced is the product of crossovers between a single GP individual and multiple individuals.

We hypothesize that, as in the case of nature, the Polyandry algorithm should generate more genetically diverse broods than BR. In GP, a group of parse trees is genetically diverse if the parse trees are structurally diverse. Alternately stated, our hypothesis claims that a Polyandry algorithm will be more explorative in genotype space as compared to BR. In this article we use the term genotype space to describe the space of all possible parse trees, given the function and terminal set for a GP problem domain. This definition follows from the extensive use of the term to describe the space of all possible encodings or genotypes in GP literature [27] [28] [29].

We further hypothesize that the greater structural diversity of a Polyandry algorithm brood should translate to greater fitness diversity in the offspring belonging to the brood. This hypothesis arises from the knowledge that genotype space and fitness space in GP are closely related [30]. The term fitness space describes the space of all possible fitness values for a given problem domain [29].

As in nature, a more diverse brood with respect to fitness space should contain offspring with more extreme fitness values. This means that a brood produced by Polyandry is more likely to contain offspring that are less fit as well as offspring that are more fit than the offspring in a BR brood. Let us then say that the Polyandry algorithm, similarly to BR, uses a culling function F_B to select offspring from the brood to input into the GP population. If the culling function is equivalent to the main fitness function of the GP, the existence of fitter offspring in a Polyandry brood will result in the introduction of fitter offspring into the GP population, compared to BR. Thus generally, the offspring produced by a Polyandry algorithm, following selection from a brood, will be fitter than the same for BR. This translates into the Polyandry algorithm demonstrating better evolvability as compared to BR.

Similarly to BR, the Polyandry algorithm, through the production of multiple offspring in a brood, makes multiple attempts at finding suitable crossover points (this is discussed in detail in Section 3). As a result, the probability of building block disruption is reduced. Over and above this, the Polyandry algorithm employs another strategy to improving evolvability. This strategy is the more aggressive search for building block information, achieved by incorporating genetic material from multiple mates into a brood. We hypothesize the dual strategy towards improving evolvability should translate to a better exhibition of this phenomenon relative to BR.

3 THE POLYANDRY ALGORITHM

Our Polyandry algorithm is similar to BR, as described in Subsection 2.2. As with BR, the algorithm replaces standard crossover in the canonical GP with an operator, $P_B(n)$. The algorithm for $P_B(n)$ is shown in Fig. 3. We note that the full Polyandry algorithm is identical to the canonical GP, with the exception that $P_B(n)$ creates $p1\%$ of $[Generation]_{(n+1)}$ in place of standard crossover (line 4a. in Fig. 1).

From Fig. 3, we observe that the only difference between $P_B(n)$ and $R_B(n)$ is that the former selects a brand new mate from the population to produce each pair of offspring in a brood. Each 'mating' act, which produces a pair of offspring in the brood, is thus performed between the principal parent and a distinct mating partner. Thus n distinct mating partners are involved in the formation of a brood of size $2n$. The probability that the same individual is selected twice as a mate depends on the size of the GP population. Generally, given a small tournament size (e.g., tournament size 2), this probability is small. We will abbreviate the Polyandry algorithm described here as PP for the remainder of this text.

Also from Fig. 3, we note the dual strategy employed by PP towards achieving evolvability. PP at-

1. Select a principal parent P_1 from the population via fitness proportionate selection or tournament selection.
2. For $i = 1$ to n:
   (a) Randomly select a mating partner P_2 from the population via fitness proportionate or tournament selection.
   (b) Perform standard crossover between P_1 and P_2 by:
      i. Randomly selecting crossover points (subtrees) S_{i1} and S_{i2} from P_1 and P_2 respectively.
      ii. Exchanging subtrees S_{i1} and S_{i2} to form brood offspring B_{i1} and B_{i2}.
3. Determine the fitness value of each one of the $2n$ members of the brood according to F_B.
4. Sort the $2n$ members of the brood in ascending order of fitness according to F_B.
5. Select offspring ("children") C_1 and C_2 which are the fittest of the $2n$ brood members according to F_B.
6. Return C_1 and C_2 as the offspring produced by $P_B(n)$, for insertion into the GP population.

Figure 3: Pseudo-code for Algorithm $P_B(n)$

tempts multiple crossover points with respect to the principal parent in order to find a suitable crossover point in the individual. This is due to the randomness involved in selecting a crossover point in the principal parent each time a pair of offspring is created and added to the brood. PP thus minimizes building block disruption in the principal parent, in a similar manner to BR. Further PP performs a wider search for building block material by incorporating genetic material from multiple mates into a brood. As discussed, this also serves to increase evolvability.

We note that for the same brood size the computational effort exerted by PP is the same as that for BR. This is because both algorithms create and evaluate the fitness of the same number of offspring. The improved evolvability we anticipate of PP relative to BR is therefore not at the expense of an increase in computational effort.

4 EXPERIMENT SETUP

Our experiments seek to empirically support three hypotheses:

H1. PP exhibits better evolvability than the canonical GP.

H2. PP exhibits better evolvability than BR.

H3. The better evolvability of PP relative to both the canonical GP and BR is as a consequence of the following:

1. PP produces more structurally diverse broods than BR. Thus PP is more explorative in genotype space than BR. A PP brood is also more structurally diverse than a pair of individuals generated by standard crossover in the canonical GP. Thus PP is also more explorative in genotype space relative to the canonical GP.
2. The more explorative nature in genotype space of PP means that the algorithm conducts a more aggressive search for building block information. Thus the algorithm is more explorative in fitness space than both the canonical GP and BR. Coupled with multiple attempts at locating suitable crossover points in the principal parent, this should lead to PP generally exhibiting better evolvability than the two algorithms.

Four benchmark Genetic Programming domains are analyzed in our experiments; the Artificial Ant, Even-5 Parity, 11-Multiplexer and Symbolic Regression domains. We implement the domains in Java SE 7 as per their original specification in [1]. Details of the problem domains are listed in Table 1.

Table 1: Problem domain details

Domain	**Function set**	**Terminal set**	**Fitness case(s)**	**Fitness function**
Artificial ant (Ant)	{IF FOOD AHEAD, P2, P3}	{MOVE, TURN RIGHT, TURN LEFT}	Santa Fe Trail	Number of pellets left on the trail
Even-5 parity (Parity)	{AND, OR, NAND, NOR}	{d0, d1, d2, d3, d4}	32 Fitness cases (all possible combinations of terminal set)	Number of incorrectly classified fitness cases
11-Multiplexer (Multiplexer)	{AND, OR, NOT, IF}	{a0, a1, a2, d0, d1, d2, d3, d4, d5, d6, d7}	2048 Fitness cases (all possible combinations of terminal set)	Number of incorrectly classified fitness cases
Symbolic regression (Regression) (order 8 polynomial - $x^8+x^7+x^6+x^5+x^4+x^3+x^2+x$)	$\{+, -, *, \%, \cos, \sin, \log, \exp\}$	$\{x\}$	21 equidistant points in the interval $[-1, 1]$	Mean squared error of function represented by parse tree from fitness cases

For all experiments conducted, a standard population size of 1000 individuals is used. Although small population sizes can be effective, large population sizes are recommended for solving difficult GP problems, such as the Even-5 Parity problem, over a small number of generations [31].

Further, we use tournament selection with a tournament size of 2. The small tournament size ensures that dissimilar mating partners are selected for the principal parent in PP. We also use an initial tree

depth of 6 (adapted from [1]) and a maximum tree depth limit of 30. This high value for the maximum depth limit is used to allow large potentially solution-finding trees to be discovered in the difficult Even-5 Parity domain. Koza reported that given a population size below 4000, no solutions are obtainable in this domain using standard GP [1], as the domain requires the formation of highly complex solution parse trees. In fact, in [1] success rates above 0 were only obtained through the use of an extremely large population (8000 individuals), or through Automatically Defined Functions. We hope to find solutions in this domain using both Polyandry and Brood Recombination, in order to demonstrate the ability of BR and PP to form complex parse trees, due to their ability to preserve and extend existing building blocks.

For all experiments, the initial population is generated using Koza's ramped half and half method [1]. We use crossover and reproduction with probabilities of 0.9 and 0.1 respectively. These parameter settings are adapted from [1]. Note that these parameter settings correspond to a probability of 0.9 for $R_B(n)$ and $P_B(n)$ in BR and PP respectively, and a reproduction probability of 0.1.

The details of all the experiments run are listed in Table 2. We choose the brood sizes in Table 2 to facilitate the comparison of BR and PP across a spectrum of brood sizes. Note that a brood size of 2 is equivalent to the canonical GP for both BR and PP. This is indicated by a (C) in Table 2. The experiments that correspond to BR and PP are also indicated in the table. Each experiment consists of 30 runs with the maximum number of generations set to 50.

Table 2: Experiment design

Brood size (brood size factor)	**Number of mates**
200 (100)	1 (BR) 100 (PP)
100 (50)	1 (BR) 55 (PP)
40 (20)	1 (BR) 20 (PP)
20 (10)	1 (BR) 10 (PP)
10 (5)	1 (BR) 5 (PP)
4 (2)	1 (BR) 2 (PP)
2 (1)	1 (C)

The metrics measured in our experiments are divided into two categories:

1. Standard GP performance metrics: Better evolvability is associated with a better algorithm performance [2]. We thus compare BR, PP and the canonical GP with respect to the following standard metrics:
   (a) The algorithm success rate. This is the proportion of the total number of runs of an algorithm that find a solution at or before the maximum number of generations have been reached. This metric is defined as the success proportion in [32].
   (b) Koza's minimum computational effort metric [1]. This is the computational effort exerted by an algorithm in solving a given problem (cf. Section 5.1, Equation 3).
   (c) The number of generations taken to converge to a global optimum solution, given a successful run of the algorithm. The four benchmark problem domains examined in this study are all GP minimization problems [1]. Thus in this study an algorithm is successful (meaning that the algorithm has reached a global optimum solution) when at least one individual in the GP population is able to attain a fitness value of 0. This indicates that the individual has attained the minimal error over all the fitness cases. The fitness cases for the four domains are listed in Table 1.
2. Metrics that investigate the causality of better evolvability: Here we compare BR and PP with respect to the following:

   The explorative capability in genotype space. The number of distinct unique subtrees in a population can be used as a measure of the structural diversity in the population [33] [34]. In this context a unique subtree is a subtree that has no duplicates in the population [34]. We measure the total number of distinct unique subtrees in a brood in order to assess the amount of structural diversity in the brood. The algorithm that consistently yields the more structurally diverse broods is the more explorative algorithm in genotype space.

   The explorative capability in fitness space. We examine the walk of an algorithm through fitness space by looking at the fitness distance between parent individuals and their offspring. Specifically, for PP, we determine $Distance_F(P_1, O)$ the fitness distance between the principal parent P_1 (Fig. 3) and an offspring belonging to the brood generated by P_1. We do the same for BR, whereby the first parent selected, P_1 (Fig. 2), acts as the principal parent. The formula for determining the fitness distance between P_1 and an offspring is shown in Equation 1.

   We note that given a brood, the standard deviation of $Distance_F(P_1, O)$ within the brood is a measure of the diversity of fitness values of the brood members relative to the principal parent. This metric, which we abbreviate as $SD_{Distance_F(P_1,O)}$, is calculated by determining $Distance_F(P_1, O)$ for each member in the brood, and subsequently determining the standard deviation of the values obtained. A high value of $SD_{Distance_F(P_1,O)}$ in-

$$Distance_F(P_1, O) = Fitness(P_1) - Fitness(O)$$

where O is an offspring member of the brood generation.

Equation 1: Fitness distance between principal parent and an offspring member of the brood.

$$Distance_F(P_1, O_{selected}) = \frac{(Fitness(P_1) - Fitness(O_{selected1})) + (Fitness(P_1) - Fitness(O_{selected2}))}{2}$$

where $O_{selected1}$ and $O_{selected2}$ are the pair of offspring selected by the culling function F_B from the brood generated by P_1. Note that $O_{selected1}$ and $O_{selected2}$ are returned into the GP population as a result of the brood operator.

Equation 2: Fitness distance between principal parent and an offspring selected from the brood.

dicates that there is a high diversity in the fitness values of offspring pertaining to the brood. $SD_{Distance_F(P_1,O)}$ is thus a measure of the amount of exploration in fitness space conducted by generating the brood.

The average fitness distance between the offspring returned by the brood operator $P_B(n)$ or $R_B(n)$ and the principal parent. This formula for this metric is provided in Equation 2. In Equation 2, $Distance_F(P_1, O_{selected})$ can be perceived as a direct measure of evolvability. In GP minimization problems, positive values of this metric indicate fitness gains resulting from a brood operator's function. Also large positive values of the metric indicate large fitness gains from resulting from the operator.

5 RESULTS DISCUSSION

5.1 Success rates and minimum computational effort

Table 3 draws a comparison of the success rates achieved for PP relative to BR across brood sizes for the different domains. From the table we observe that the general trend, for both PP and BR, is that larger brood sizes correspond to higher success rates. This is expected as we have established that larger brood sizes are associated with lower probabilities of building block disruption, and hence better evolvability. From Table 3 we note therefore that both algorithms generally exhibit increasingly higher success rates than the canonical GP with increasing brood size across all four domains. From these observations we can conclude that Polyandry delivers an increasingly better performance than the canonical GP with increasing brood size. This supports hypothesis 1 (Section 4).

We observe in Table 3 that given a domain and a fixed brood size, PP generally tends to have a higher or equivalent success rate to BR. This is true for over 90% of the observations in the table.

Given the same domain and brood size, discrepancies in success rates between BR and PP were evaluated by conducting a statistical Z-test for two sample means [35]. The tests did not establish significance. Hence we cannot prove the statistical significance of observations where the success rate for PP is higher than the same for BR. However the success rates for both PP and BR are already high for brood sizes above 2, 4, 4 and 10 in the Regression, Ant, Multiplexer and Parity domains respectively. Therefore the success rate metric seems inadequate for a comparison between BR and PP, as PP does not perform significantly better than an already highly performing BR with regards to the metric. In light of this we will further compare BR and PP with regard to the minimum computational effort as well as the number of generations taken to converge to a global optimum solution.

Table 3: Algorithm success rates

	Domain			
Algorithm (Brood size)	**Ant**	**Parity**	**Multiplexer**	**Regression**
C(2)	0.13	0.00	0.00	0.40
BR(4)	0.27	0.00	0.07	0.77
PP(4)	0.33	0.00	0.13	0.83
BR(10)	0.67	0.13	0.87	0.97
PP(10)	0.73	0.13	1.00	0.93
BR(20)	0.93	0.53	1.00	0.93
PP(20)	0.93	0.67	1.00	0.97
BR(40)	0.97	0.97	1.00	0.90
PP(40)	1.00	0.97	1.00	0.90
BR(100)	1.00	1.00	1.00	0.97
PP(100)	1.00	1.00	1.00	0.97
BR(200)	1.00	1.00	1.00	0.93
PP(200)	1.00	1.00	1.00	0.90

Table 4 shows the minimum number of fitness evaluations (that is the minimum computational effort) required to give a probability of success of 0.99. The minimum computational effort metric in is calculated for PP and BR for the different brood sizes. The values obtained are compared to values of the compu-

tational effort reported in literature for other existing Evolutionary Algorithm methodologies.

Specifically in Table 4 we compare the computational effort for BR and PP with the same for the canonical GP [1], Evolutionary Programming [36], Traceless GP [37], Cartesian GP [38] and Size Fair and Homologous Crossover GP [39].

Four of the observations in Table 4 are marked with "No solution" as a result of the corresponding runs being too difficult for canonical GP or BR and PP given a small brood size. These observations correspond to algorithm success rates of 0.00 in Table 3. The computational effort is indeterminate when an algorithm has a success rate of 0 [1] [40].

The formula for the calculation of the minimum computational effort $I(M, i, z)$, as described on page 194 of [1], is shown in Equation 3.

$$I_{min}(M, i, z) = min_i(MR(z)(i + 1))$$

where

1. z is the probability of success required by the time generation i is reached, at least once in R runs. Here z= 0.99
2. M is the population size
3. i is the generation number
4. $R(z) = ceil\left\{\frac{\log(1-z)}{\log(1-P(M,i))}\right\}$ is the number of runs required to satisfy z
5. $P(M, i) = \left\{\frac{N_S(i)}{N_{total}}\right\}$ is the cumulative probability of success by generation i

Equation 3: Formula for evaluating GP computational effort, taken from [1]

We note that PP and BR perform additional computations compared to the canonical GP. Thus in Table 4 the minimum computational effort $I(M, i, z, n)$ is calculated differently for both algorithms, as shown in Equation 4. Let us discuss Equation 4 briefly. In both PP and BR, the contribution of 2 individuals (that is offspring) to the next GP generation requires the fitness evaluation of all the members of a brood of size $2n$, where n is the brood size factor. As the contribution of 2 GP individuals corresponds to $2n$ fitness evaluations, we can say that the contribution of 1 GP individual corresponds to n fitness evaluations. Thus the contribution of M GP individuals to each new generation requires $M \times n$ fitness evaluations. We note that Equation 4 is in fact a slight overestimate of the computational effort exerted by BR/PP. This is because only 90% of the individuals in a GP generation are created by $R_B(n)$ and $P_B(n)$ in BR in PP respectively (refer to the parameter settings in Section 4). The remaining 10% of the individuals in a GP generation are created by the reproduction operator.

Also in Equation 4, the ceiling operator is ignored in the calculation of $R(z)$, as recommended in [40]. According to [40] this should lead to a more accurate estimation of the true value of the computational effort from the sample data gathered in our experiments.

$$I_{min}(M, i, z) = min_i((M \times n)R(z)(i + 1))$$

where

1. n is the PP/BR brood size factor
2. $$R(z) = \begin{cases} \frac{\log(1-z)}{\log(1-P(M,i))} & if P(M, i) < z \\ 1 & if P(M, i) \geqslant z \end{cases}$$

is the number of runs required to satisfy z.

Equation 4: Formula for evaluating PP/BR computational effort

Equation 4 specifies the value of $R(z)$ when $P(M, i) \geqslant z$ (where $z = 0.99$). This specification is adopted from [32] and is missing in the original formulation for computational effort prescribed in [1]. From Table 1 it is clear that this specification is useful to our analysis, as several experiments attain a success rate of 1.00 by generation 50 (that is $P(M, i) = 1.00$ when $i = 50$ for a large number of our experiments). From Equation 4 we can thus determine the computational effort at generation 50 when $P(M, i) = 1.00$.

The remaining quantities in Equation 4 (that is z, M, i and $P(M, i)$) are determined the same way as in Equation 3. For the remainder of this text the mention of minimum computational effort with respect to BR and PP refers to computational effort calculated according to Equation 4.

Following the recommendations in [40], in Table 4 we list $I_{min}(M, i, z, n)$ for BR and PP along with the generation i in which $I_{min}(M, i, z, n)$ occurs and the corresponding value of $P(M, i)$. These values are listed in the format:

$$I_{min}(M, i, z, n) \quad (i, P(M, i))$$

The first row of results in Table 4 lists the computational effort of the canonical GP for the four domains, taken from [1]. The value of this metric for the 11-Multiplexer problem was not provided in [1]. Therefore the value listed in the table, marked with a $*$, is in fact the computational effort for the 6-Multiplexer problem, according to [1]. We use this value to infer the performance advantages of PP relative to the canonical GP for the 11-Multiplexer problem. Also, from table 4.1 (Section 4), we examined an order 8 polynomial in our experiments. Koza examined an order 4 polynomial in [1]. Therefore the computational effort for the canonical GP in Table 4, marked with a $**$, is the computational effort for the order 4 polynomial in [1], which we use to infer the performance advantages of PP relative to the canonical GP for the order 8 polynomial. We chose to use an order 8 polynomial in our experiments after empirically observing that lower order polynomials present easy problems for the BR and PP algorithms, such that both algorithms perform very well and it is not easy to observe performance

Table 4: Minimal computational effort $I(M, i, z)/1000$

	Domain			
Algorithm (Parameters / Brood size)	**Ant**	**Parity**	**Multiplexer**	**Regression**
Canonical GP [1]	450	7840	245∗	162.5∗∗
Canonical GP	580 (17, 0.13)	No solution	No solution	433 (47, 0.40)
BR(4)	1248 (41, 0.27)	No solution	6275 (46, 0.07)	222 (34, 0.77)
PP(4)	702 (16, 0.20)	No solution	3219 (49, 0.13)	230 (24, 0.63)
BR(10)	981 (38, 0.60)	6115 (37, 0.13)	399 (30, 0.83)	162 (18, 0.93)
PP(10)	524 (18, 0.57)	5632 (34, 0.13)	195 (38, 1.00)	**154 (17, 0.93)**
BR(20)	324 (18, 0.93)	2525 (37, 0.50)	230 (22, 1.00)	188 (10, 0.93)
PP(20)	309 (11, 0.83)	1677 (39, 0.67)	**170 (16, 1.00)**	176 (12, 0.97)
BR(40)	520 (12, 0.90)	1120 (27, 0.90)	280 (13, 1.00)	480 (11, 0.90)
PP(40)	**300 (14, 1.00)**	1040 (25, 0.90)	240 (11, 1.00)	400 (9, 0.90)
BR(100)	450 (8, 1.00)	850 (16, 1.00)	550 (10, 1.00)	677 (9, 0.97)
PP(100)	400 (7, 1.00)	**850 (16, 1.00)**	400 (7, 1.00)	610 (8, 0.97)
BR(200)	700 (6, 1.00)	1400 (13, 1.00)	900 (8, 1.00)	1531 (8, 0.93)
PP(200)	700 (6, 1.00)	1400 (13, 1.00)	700 (6, 1.00)	1400 (6, 0.90)
EP [31]	-	2100	-	-
Traceless GP [32]	-	2417.5	-	-
Cartesian GP (non-neutral, mutation rate = 0.1) [33]	139	-	-	-
Cartesian GP (neutral, mutation rate = 0.4) [33]	511	-	-	-
Cartesian GP (non-neutral, mutation rate = 0.12) [33]	888	-	-	-
Size Fair Crossover GP (initial tree depth within interval [2, 6]) [34]	-	-	270	-
Homologous Crossover GP (initial tree depth within interval [2, 6]) [34]	-	-	290	-

discrepancies between the two algorithms.

In the second row of results in Table 4, we list the computational effort of the canonical GP for the four domains determined empirically from our own experiments. We calculate this using Equation 4 with a brood size factor of 1. The results for the canonical GP are listed in the same format as the same for BR and PP.

For each domain in Table 4 the version of PP that returns the lowest minimum computational effort is highlighted in a bold font. We observe that in the Artificial Ant domain, PP (40) has the lowest value of this metric for the different versions of PP. This value is lower than the minimum computational effort for the canonical GP according to both [1] and our empirical observations. Further, from the table, we observe that the computational effort exerted by PP (40) in the ant domain is competitive compared to other EA methodologies described in literature.

In the Even-5 Parity domain, the best performing PP algorithm, PP (100), has a minimum computational effort that is lower than both the canonical GP and other methodologies described in literature.

In the 11-Multiplexer domain, the best performing PP algorithm, PP (20), requires a computational effort that is lower than the computational effort for the canonical GP for the 6-Multiplexer problem according to [1]. We extrapolate that the computational effort for the canonical GP for the harder 11-Multiplexer problem is substantially higher than the value in the table (245,000) and must therefore be higher than the computational effort for PP (20). From the table, PP (20) also yields a lower computational effort than other algorithms in literature for the 11-Multiplexer problem.

From Table 4, in the Symbolic Regression domain, the best performing PP algorithm, PP (10), requires

a computational effort that is comparable to Koza's value of the computational effort of the canonical GP for the less GP-hard order 4 polynomial [1]. From our own empirical observations, the computational effort required by PP (10) is less than that of the canonical GP.

We can therefore make the overall conclusion that given a suitable brood size, PP requires less computational effort than the canonical GP to arrive at a global optimum GP solution. This further supports hypothesis 1 in Section 4. We also conclude that given a suitable brood size, PP exerts a competitively low computational effort compared to other EA methodologies in literature.

From Equation 4, larger brood sizes for PP should lead to the exertion of more computational effort than smaller brood sizes for the algorithm. The increase in brood size is however offset by the fact that larger brood sizes correspond to higher success rates (Table 1) and faster solution discovery (as we will see in Subsection 5.2). As a consequence, we observe in Table 4 that up to brood sizes of 40, 100, 20 and 10 in the Ant, Parity, Multiplexer and Regression domains respectively, an increase in the brood size parameter for PP actually results in a decrease in computational effort.

However in Table 4 we observe that increasing the brood size of PP does not indefinitely correspond to a decrease in the computational effort of the algorithm. Generally, beyond brood sizes 40, 100, 20 and 10 in the Ant, Parity, Multiplexer and Regression domains respectively, an increase in brood size leads to an increase in computational effort. Beyond these threshold brood sizes, there are no more significant performance gains in the algorithm for larger brood sizes. The effect is that the potential increase in computational effort due to a larger brood size is no longer being offset by performance gains. From Table 4, we observe a similar pattern in the behaviour of BR. These observations provide evidence of the existence of a brood diversity point in both BR and PP.

We now compare BR and PP in terms of the computational effort exerted. From Table 4 we observe that given the same brood size and domain, PP generally requires less computational effort than BR. This is true for over 80% of the observations in the table. We conclude that PP requires less computational effort than BR. This supports hypothesis 2 in Section 4. In Subsection 5.2, we will observe that PP requires less computational effort than BR because the former finds solutions in significantly fewer generations than the latter.

5.2 Number of generations to a solution

Figs. 4, 5, 6, and 7 are Box Whisker Charts contrasting the number of generations taken to find a solution for BR and PP, in the Ant, Parity, Multiplexer and Regression domains respectively. Note that in each chart in the figures, the white line corresponds to the median of the chart. Also note that the Parity problem had a success rate of 0 for brood sizes below

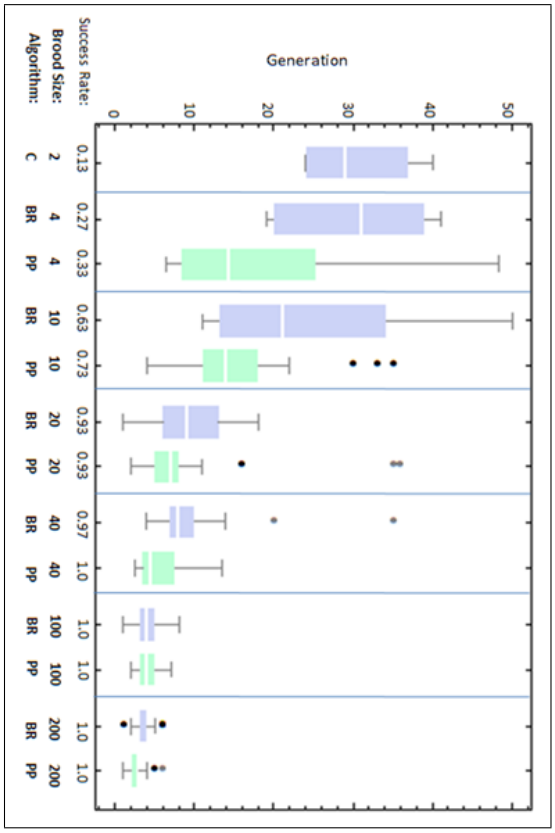

Figure 4: Artificial Ant Domain – PP vs. canonical GP and BR: number of generations taken to find a solution.

10; hence these are not included in Fig. 5. Similarly the Multiplexer problem had a success rate of 0 for the canonical GP. This algorithm is excluded from Fig. 6.

In Figs. 4–7 we observe that in general, the larger the brood size, the fewer the number of generations required to find a solution for both BR and PP. Once again, this is expected as larger brood sizes correspond to better evolvability. Further we observe that the variance in the number of generations required (that is the width of the boxplots) decreases as the brood size is increased. Again this is true for both algorithms. We associate the decrease in the variance with an increase in the effectiveness of the algorithms as the brood size is increased. Larger brood sizes translate to more consistency with respect to preventing building block disruption.

The main observation in Figs. 4–7 is that given a fixed brood size, PP takes consistently fewer generations to find a solution than BR. This observation holds across the different brood sizes and domains. Further, both BR and PP generally take increasingly fewer generations to find solutions than the canonical GP with increasing brood size.

Table 5 is extrapolated from Figs. 4–7. The table draws a comparison of the median number of generations taken by the algorithms to find solutions across brood sizes and domains. Given a domain and a fixed brood size, the difference in the median number of generations between BR and PP is shown in brackets. Here the PP metric is subtracted from the BR metric. We use the median here as this metric is less sensitive

Table 5: Median number of generations to a solution

	Domain			
Algorithm (Brood size)	**Ant**	**Parity**	**Multiplexer**	**Regression**
C(2)	29.0	-	-	28.0
BR(4)	31.0	-	45.5	23.5
PP(4)	14.0 *(17.0)*	-	35.5 *(10.0)*	18.5 *(5.0)*
BR(10)	21.0	46.5	26.5	15.5
PP(10)	13.5 *(7.5)*	32.0 *(14.5)*	17.0 *(9.5)*	10.0 *(5.5)*
BR(20)	9.5	30.5	16.5	10.0
PP(20)	7.0 *(2.5)*	21.0 *(9.5)*	12.0 *(4.5)*	7.5 *(2.5)*
BR(40)	8.0	21.0	11.5	9.0
PP(40)	4.5 *(3.5)*	16.5 *(4.5)*	8.0 *3.5*	6.5 *(2.5)*
BR(100)	4.0	14.5	9.5	7.0
PP(100)	4.0 *(0.0)*	12.0 *(2.5)*	6.0 *(3.5)*	6.0 *(1.0)*
BR(200)	3.5	12.0	6.5	6.0
PP(200)	2.0 *(1.5)*	10.0 *(2.0)*	8.0 *(1.5)*	5.0 *(1.0)*

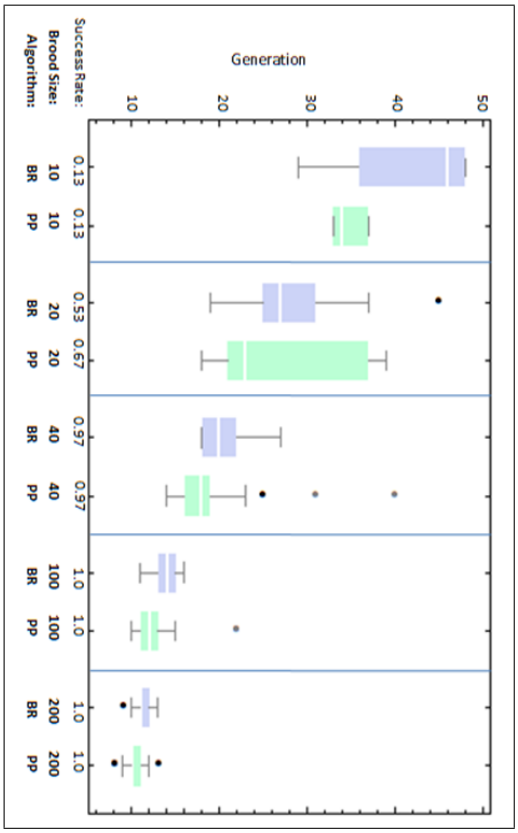

Figure 5: Even-5 Parity Domain – PP vs. canonical GP and BR: number of generations taken to find a solution.

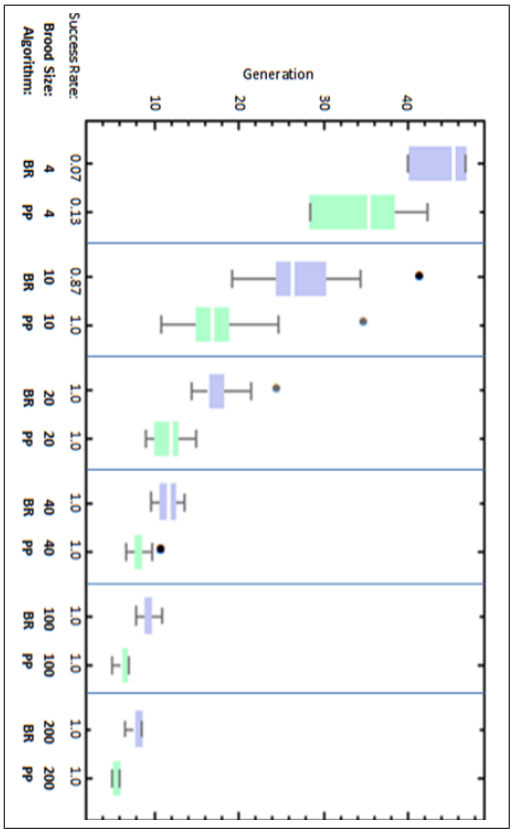

Figure 6: 11-Multiplexer Domain – PP vs. canonical GP and BR: number of generations taken to find a solution.

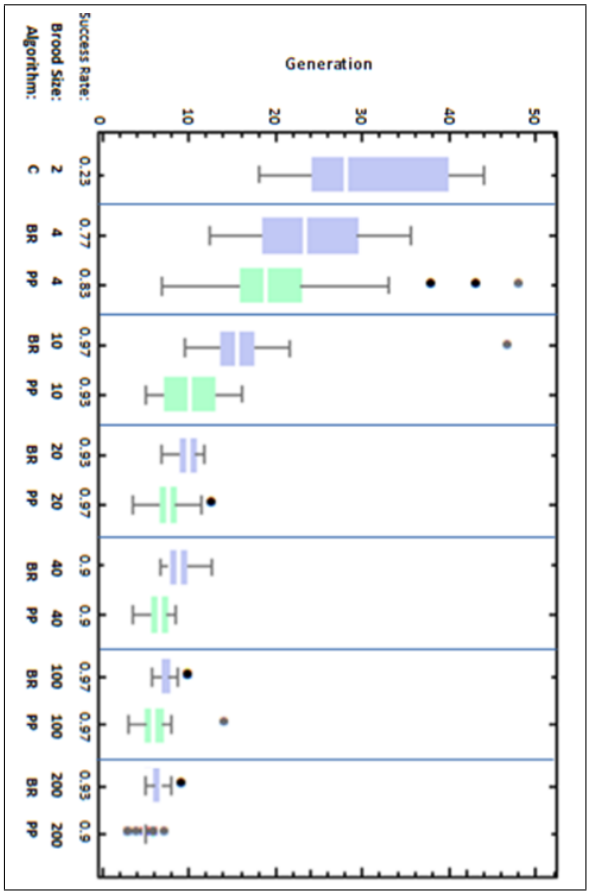

Figure 7: Symbolic Regression Domain – PP vs. canonical GP and BR: number of generations taken to find a solution

to outliers than the mean [35] [41].

Table 5 provides further evidence that for the same brood size PP consistently finds solutions faster than BR. The difference in the median number of generations is substantive, the maximum value of this metric reaching 17.0, 14.5, 10.0 and 5.5 in the Ant, Parity, Multiplexer and Regression domains respectively.

In Table 5 we observe that the smaller the brood size, the faster PP finds solutions relative to BR. This is true across all four domains. We theorize that this is due to BR becoming a more competitive algorithm with increasing brood size.

We conducted statistical tests to verify that PP finds solutions significantly faster than the canonical GP. Also we conducted tests to verify that, given a domain and a fixed brood size, PP finds solutions significantly faster than the BR algorithm with the same brood size. Our tests utilized the one-tail Student's t-distribution [41] to compare the sample means of the number of generations to a solution for PP with the same for both the canonical GP and BR. The p-values resulting from the t-tests are listed in Table 6. In the table, given a domain and a fixed brood size, p_1 is the p-value indicating the significance of the difference in the number of generations to a solution between PP and the canonical GP. Similarly p_2 is the p-value indicating the significance of the difference in the number of generations to a solution between PP and the BR algorithm with the same brood size. Note that because the canonical GP did not find solutions for both Parity and Multiplexer problems, p_1 cannot be determined in these domains. Also BR(4) and PP(4) did not find solutions in the Parity domain and thus the corresponding values of p_1 and p_2 are not listed in the table. In Table 6 p-values that demonstrate significance (at $\alpha = 0.05$) are marked with an (s) symbol.

From Table 6 we observe that at a significance level of 0.05, PP finds solutions faster than the canonical GP, for all brood sizes, in the Ant and Regression domains. This empirically supports hypothesis 1 in Section 4. Also from the values of p_2 in Table 6 we observe that the advantage of PP over BR (visible in Figs. 4—7) is significant for 100% of the brood sizes tested in the Multiplexer domain, over 80% of the brood sizes tested in the Ant domain as well as over 60% of the brood sizes tested in the Parity and Regression domains. We thus have statistical evidence that the likelihood of PP discovering solutions faster than BR is high. This supports hypothesis 2 in Section 4.

We therefore establish that PP discovers solutions in fewer generations than the canonical GP. Further, we establish that given a domain and a fixed brood size although BR and PP exhibit similar success rates, PP finds solutions in fewer generations than BR. These results suggest an underlying better evolvability of PP relative to both the canonical GP and BR.

Table 6: PP vs. canonical GP and BR: significance of difference in numbers of generations to a solution

Brood size	**Ant**	**Parity**	**Multiplexer**	**Regression**
4	$p_1 = 0.041_{(s)}$ $p_2 = 0.022_{(s)}$	- -	- $p_2 = 0.033_{(s)}$	$p_1 = 0.040_{(s)}$ $p_2 = 0.443_{(s)}$
10	$p_1 = 0.000_{(s)}$ $p_2 = 0.030_{(s)}$	- $p_2 = 0.098$	- $p_2 = 0.049_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.040_{(s)}$
20	$p_1 = 0.000_{(s)}$ $p_2 = 0.007_{(s)}$	- $p_2 = 0.270$	- $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.190$
40	$p_1 = 0.000_{(s)}$ $p_2 = 0.044_{(s)}$	- $p_2 = 0.000_{(s)}$	- $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.049_{(s)}$
100	$p_1 = 0.000_{(s)}$ $p_2 = 0.466$	- $p_2 = 0.000_{(s)}$	- $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.022_{(s)}$
200	$p_1 = 0.000_{(s)}$ $p_2 = 0.001_{(s)}$	- $p_2 = 0.000_{(s)}$	- $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$

5.3 Genotype space and fitness space exploration metrics

Having established PP to consistently find solutions faster than BR and the canonical GP, and also having established PP to have a higher success rate than the canonical GP, we seek to establish causality. We do this by empirically supporting hypothesis 3 (Section 4).

Fig. 8 depicts the trend of the average number of unique subtrees within a brood for the broods created in each generation by BR and PP for brood sizes 4, 10, 20, 40, 100 and 200 in the Artificial Ant domain. The results are compared with the trend of the average number of unique subtrees within pairs of individuals resulting from crossover in the canonical GP in each generation over time. Recall that the number of unique subtrees in a brood is a measure of the structural diversity of the brood. In the figure, the termination of a curve is due to the convergence of an algorithm to a global optimum solution.

In Fig. 8 the increasing trend of a given curve is likely to be due to the growth of the parse trees in the GP population, and the discovery of new subtrees in the population as a consequence of the randomness of crossover point selection. Subsequent decreases in the trend are likely to be associated with convergence of the GP population. We use only one version of the canonical GP. Thus the curve of the canonical GP is in fact the same in Fig. 8 a, b, c, d, e and f, (the curve may appear different in the figures due to the different scales of the y-axis employed to accommodate the other curves). We note the early termination of the PP curve for brood sizes 40, 100 and 200. This is due to all the runs of the algorithm having found a global optimum solution.

From Fig. 8 we immediately observe that the broods generated by PP are consistently more structurally diverse than the broods generated by BR across brood sizes, and across generations given a fixed brood size. We obtained similar observations in the Parity, Regression and Multiplexer domains. We observe that both BR and PP tend to produce increasingly more structurally diverse broods than the canonical GP with increasing brood size. For brood sizes 4, 10 and 20 in the figure, BR and PP have an early advantage in structural diversity. The early advantage in structural exploration leads to convergence of the algorithms (that is a subsequent loss of structural diversity) prior to the canonical GP. This is observed in the curves for BR and PP subsequently falling below the curve for the canonical GP after the early generations in Fig. 8 a and b. Similarly the curve for BR falls below the curve for the canonical GP after the early generations in Fig. 8 c. Given empirical results similar to those in Fig. 8 for all the four domains, we conducted one-way ANOVA statistical tests. The aim of the tests was to verify that given a domain and a fixed brood size, PP consistently yields significantly more structurally diverse broods than both BR and the canonical GP across GP generations. The resulting p-values are shown in Table 7.

In Table 7, given a domain and a fixed brood size, p_1 is the p-value indicating the significance of the difference in the average number of unique subtrees between PP and the canonical GP.

Similarly p_2 is the p-value indicating the significance of the difference in the average number of unique subtrees between PP and the BR algorithm with the same brood size. In Table 7 the p-values that demonstrate significance (at $\alpha = 0.05$) are marked with an (s) symbol. From Table 7 we observe that with the exception of the PP algorithms with brood sizes 4 and 10 in the Ant domain, the values of p_1 are all below 0.01. Therefore for the majority of the observations in the table, we are 99% confident that PP is more explorative in genotype space relative to the canonical GP. Also from Table 7 we observe that with the exception of brood size 4 for the Multiplexer and Regression domains, the values of p_2 are all below 0.05. Therefore for the majority of the observations in the table, we are 95% confident that PP is more explorative in genotype space than BR. The observations in Table 7 empirically support hypothesis 3a (Section 4).

Fig. 9 compares the trends in $SD_{Distance_F(P_1,O)}$ (defined in Section 4) for BR and PP over time for

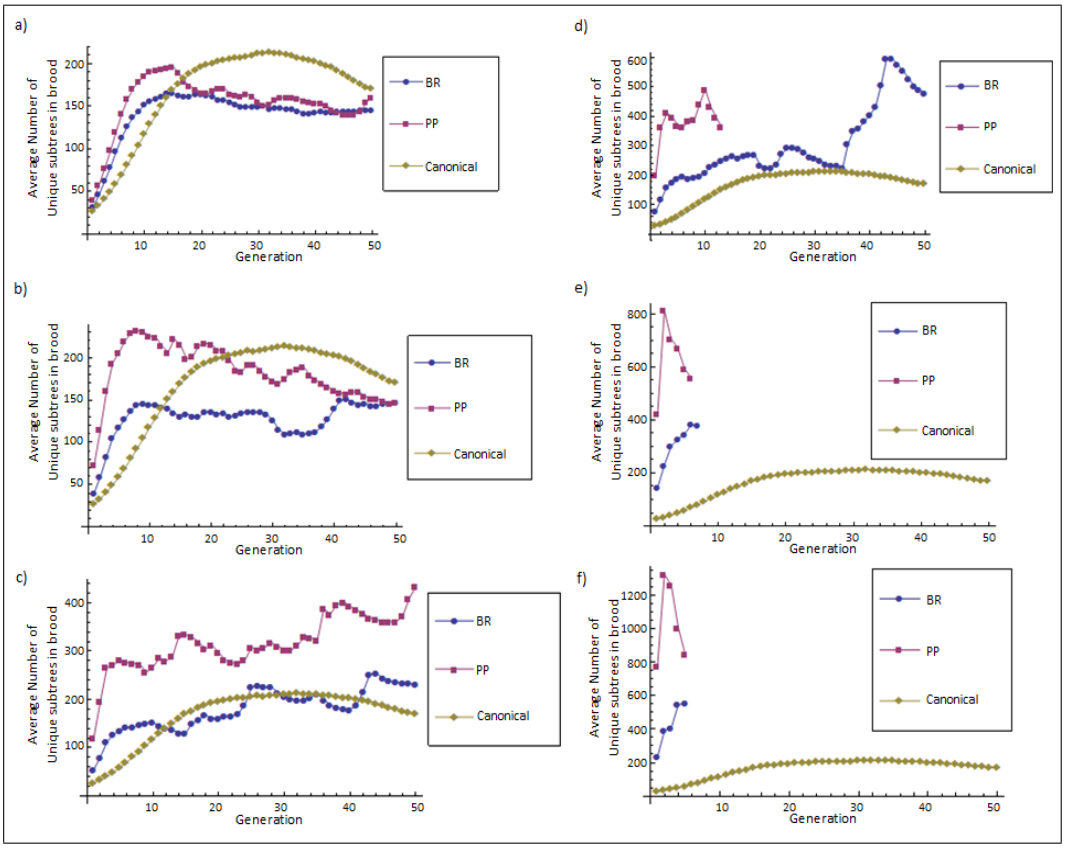

Figure 8: Artificial Ant Domain – PP vs. BR and canonical GP: Average number of unique subtrees in brood over time for brood sizes: a) 4, b) 10, c) 20, d) 40, e) 100, and f) 200.

Table 7: PP vs. canonical GP and BR: significance of difference in average number of unique subtrees

Brood size	**Ant**	**Parity**	**Multiplexer**	**Regression**
4	$p_1 = 0.179$ $p_2 = 0.022_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.009_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.149_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.502_{(s)}$
10	$p_1 = 0.070$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$
20	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000$	$p_1 = 0.000_{(s)}$ $p_2 = 0.006_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000$
40	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.011_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$
100	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.015_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.007_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$
200	$p_1 = 0.000_{(s)}$ $p_2 = 0.001_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.030_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.007_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$

brood sizes 4, 10, 20, 40, 100, 200 in the Artificial Ant Domain. The results are compared with the trend in the average value of $SD_{Distance_F(P_1,O)}$ within pairs of individuals resulting from standard crossover in the canonical GP over time. Recall that given a brood, $SD_{Distance_F(P_1,O)}$ pertaining to the brood is a measure of the amount of exploration conducted in fitness space by generating the brood. In the case of the canonical GP, $SD_{Distance_F(P_1,O)}$ is the amount of exploration conducted in fitness space by generating a pair of individuals from standard crossover. The curve of the canonical GP is the same in Fig. 9 a, b, c, d, e and f.

In Fig. 9 we observe some consistency with respect to the metric being higher for both BR and PP relative to the canonical GP. This is evidence of both algorithms being more explorative in fitness space relative to the canonical GP. Further, we observe that the metric for PP is generally higher than the same for BR, more so in the earlier GP generations. For brood sizes 4, 10 and 20, the curve for PP may tend to fall below the curve for BR in later generations due to PP converging faster. Convergence of the GP population means that the main GP individuals begin to possess similar fitness values. As a result, brood offspring also possess similar fitness values, leading to lower values of $SD_{Distance_F(P_1,O)}$. If PP consistently exhibits better evolvability than BR, we do indeed expect PP to make faster fitness gains and thus reach convergence before BR.

Empirical observations obtained from the other three domains were similar to the ones depicted in Fig. 9. However in the Symbolic Regression domain, the fitness values of some unfit individuals are astronomically large numbers (for example 2.27×10^{184}). As a result, metrics that are computed using average fitness values, such as $SD_{Distance_F(P_1,O)}$, take on astronomically large values. Thus it becomes difficult to compare the values of $SD_{Distance_F(P_1,O)}$ for the different algorithms, and observe subtle discrepancies in the metric, in this domain.

Given empirical results for all the four domains, we conducted one-way ANOVA statistical tests. The aim of the tests was to verify that PP is significantly more explorative in fitness space than both the canonical GP and BR. The p-values resulting from these tests are shown in table 5.6. In the table, given a domain and a fixed brood size, p_1 is the p-value indicating the significance of the difference in the average value of $SD_{Distance_F(P_1,O)}$ between PP and the canonical GP.

Similarly in Table 9 p_2 is the p-value indicating the significance of the average value of $SD_{Distance_F(P_1,O)}$ between PP and the BR algorithm with the same brood size. In the table the p-values that demonstrate significance (at $\alpha = 0.05$) are marked with an (s) symbol. From Table 9, we observe that the tests cannot distinguish between astronomically large values in the Symbolic Regression domain. We will leave this domain out of the ensuing discussion. In the other three domains, we observe that the values of p_1 are all below 0.05. Thus we are 95% confident that PP is a more explorative algorithm in fitness space, compared to the canonical GP, in accordance with hypothesis 3b (Section 4).

The values of p_2 do not show any significance. However it is clear from Fig. 9 as well as the empirical data gathered from the Parity and Multiplexer domains that a higher $SD_{Distance_F(P_1,O)}$ for PP in the earlier generations (as compared to BR) is the general expectation. Therefore we have empirical evidence of the more explorative nature in fitness space of PP relative to BR in accordance with hypothesis 3b. Hypothesis 3 in Section 4 also states that a more explorative algorithm in fitness space should exhibit a better evolvability. In Fig. 10, we use a Gaussian distribution to model the distribution of $Distance_F(P_1, O)$ metrics in a brood. Gaussian distributions are commonly used to model unknown distributions [41] [41]. For a fixed domain and brood size, given an early GP generation, in which $SD_{Distance_F(P_1,O)}$ for PP is higher than $SD_{Distance_F(P_1,O)}$ for BR, a comparison is drawn in the figure between a PP brood, a BR brood and a pair of individuals generated by crossover in the canonical GP. Specifically Fig. 10 compares PP, BR and the canonical GP in generations 2, 4 and 6 for a brood size of 100 in the Artificial Ant Domain. For a given algorithm and domain, the Gaussian distribution of $Distance_F(P_1, O)$ in a generation is modeled by using the average value $Distance_F(P_1, O)$, as well as the standard deviation of $Distance_F(P_1, O)$ (that is $SD_{Distance_F(P_1,O)}$) in the generation.

From Fig. 10, we observe that the generally higher standard deviation of fitness values in a PP brood results in a higher likelihood of more positive extreme fitness values in the brood, compared to the other two algorithms. This in turn means that the culling function F_B will return fitter offspring from a PP brood relative to both a BR brood and a pair of offspring created by standard crossover in the canonical GP. The observations in Fig. 10 are generally true in early GP generations, for all brood sizes in the Ant, Parity and Multiplexer domains. We can therefore establish that when $SD_{Distance_F(P_1,O)}$ for PP is higher than $SD_{Distance_F(P_1,O)}$ for both BR and the canonical GP, PP exhibits better evolvability than the other two algorithms.

Fig. 11 depicts the trend of $Distance_F(P_1, O)$ with time for the different brood sizes in the Ant domain. This metric is plotted for PP, BR and the canonical GP. In the context of the canonical GP, $Distance_F(P_1, O_{selected})$ is the distance between p_1 and the pair of offspring that result from standard crossover with another GP individual. In the figure we observe that the curve for the canonical GP is below the x-axis. This indicates that the canonical GP predominantly produces offspring that are less fit than their parents. The observation is consistent with the theory of the predominantly disruptive nature of the standard crossover operator. A vital observation in the figure is that $R_B(n)$ and $P_B(n)$ are both in fact predominantly constructive operators for brood sizes 10 and above. This is clear as the corresponding curves for the algorithms are above the x-axis, indicating that the offspring returned by the brood operators are fitter

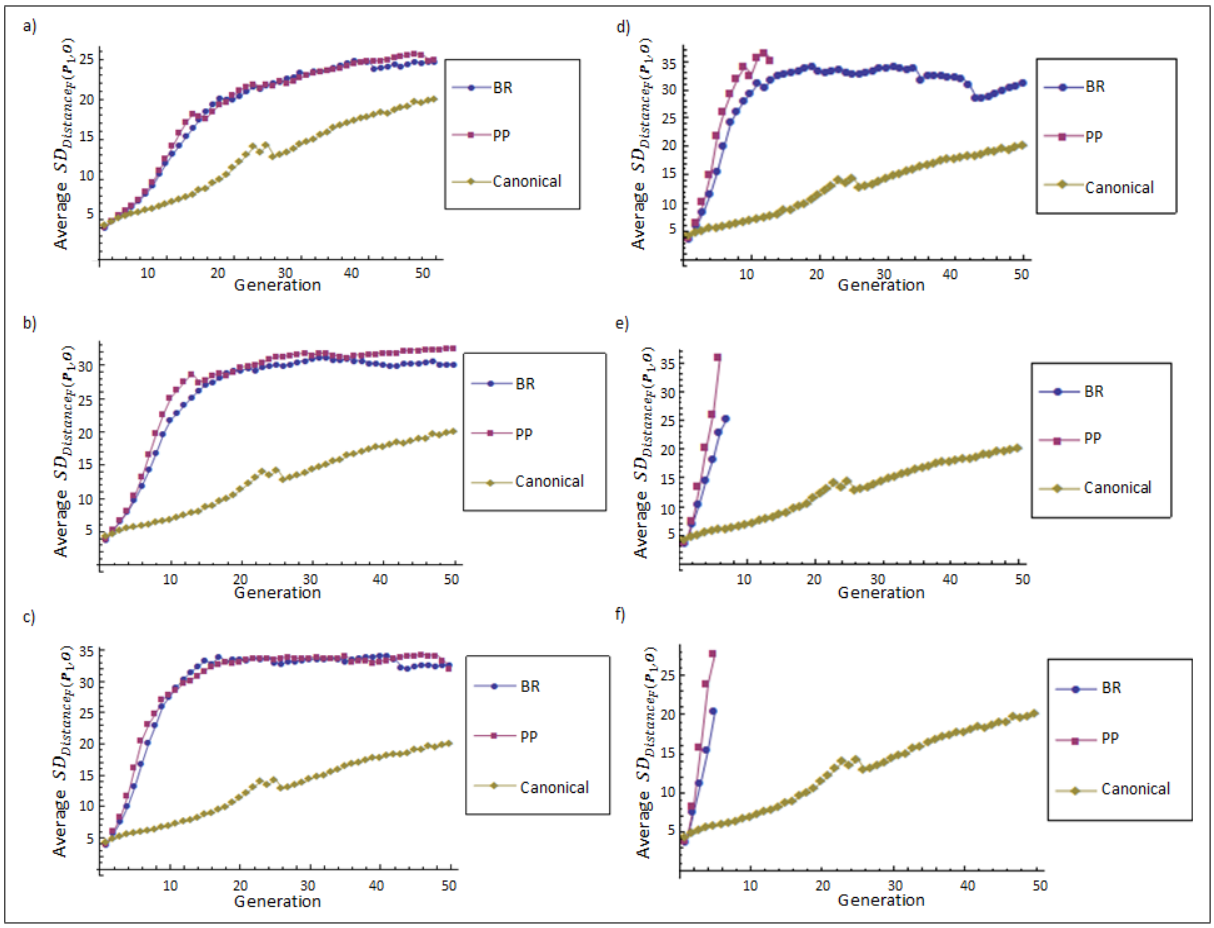

Figure 9: Artificial Ant Domain – PP vs. BR and canonical GP: Average $SD_{Distance_F(P_1,O)}$ within a brood over time for brood sizes: a) 4, b) 10, c) 20, d) 40, e) 100, and f) 200.

Table 8: PP vs. canonical GP and BR: significance of difference in average value of $SD_{Distance_F(P_1,0)}$

Brood size	**Ant**	**Parity**	**Multiplexer**	**Regression**
4	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.653$
	$p_2 = 0.809$	$p_2 = 0.668$	$p_2 = 0.312$	$p_2 = 0.532$
10	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.324$
	$p_2 = 0.442$	$p_2 = 0.896$	$p_2 = 0.434$	$p_2 = 0.631$
20	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.333$
	$p_2 = 0.838$	$p_2 = 0.518$	$p_2 = 0.503$	$p_2 = 0.323$
40	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.545$
	$p_2 = 0.357$	$p_2 = 0.396$	$p_2 = 0.413$	$p_2 = 0.332$
100	$p_1 = 0.027_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.010_{(s)}$	$p_1 = 0.347$
	$p_2 = 0.406$	$p_2 = 0.785$	$p_2 = 0.406$	$p_2 = 0.317$
200	$p_1 = 0.041_{(s)}$	$p_1 = 0.000_{(s)}$	$p_1 = 0.022_{(s)}$	$p_1 = 0.347$
	$p_2 = 0.448$	$p_2 = 0.579$	$p_2 = 0.392$	$p_2 = 0.332$

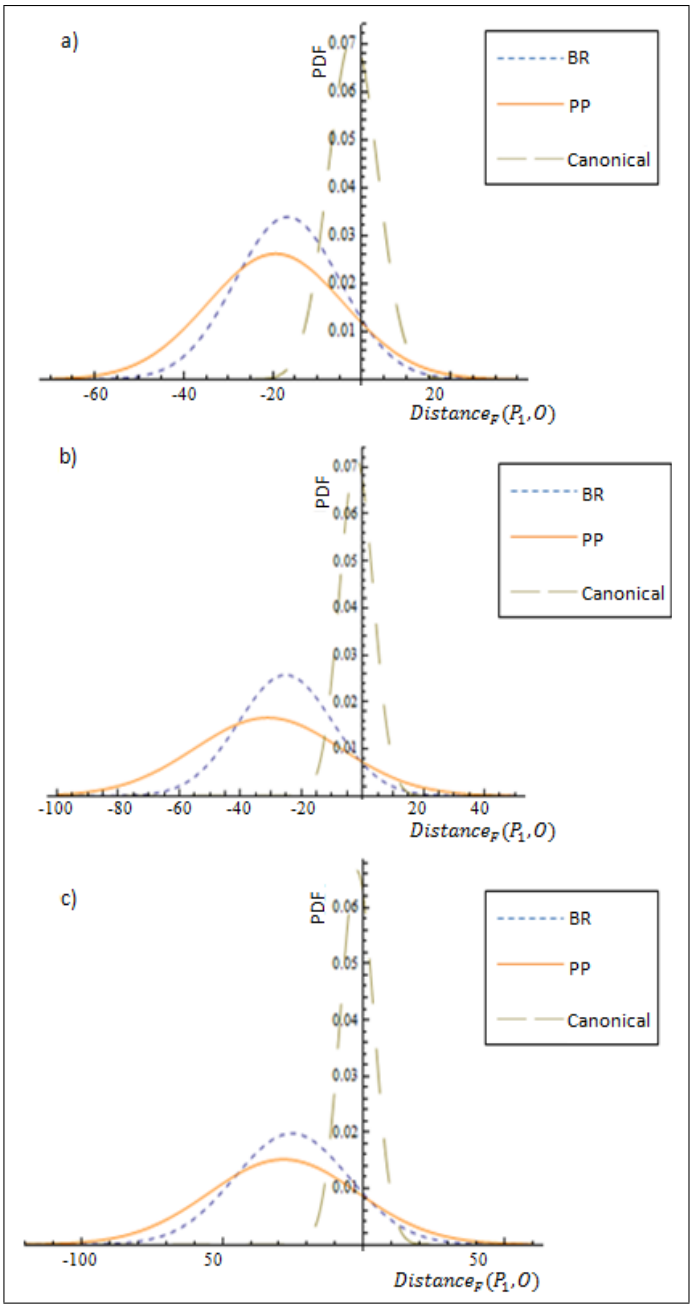

Figure 10: Artificial Ant Domain, brood size 100 – PP vs. BR and canonical GP: Distribution of offspring fitness values for generations a) 2, b) 4, and c) 6.

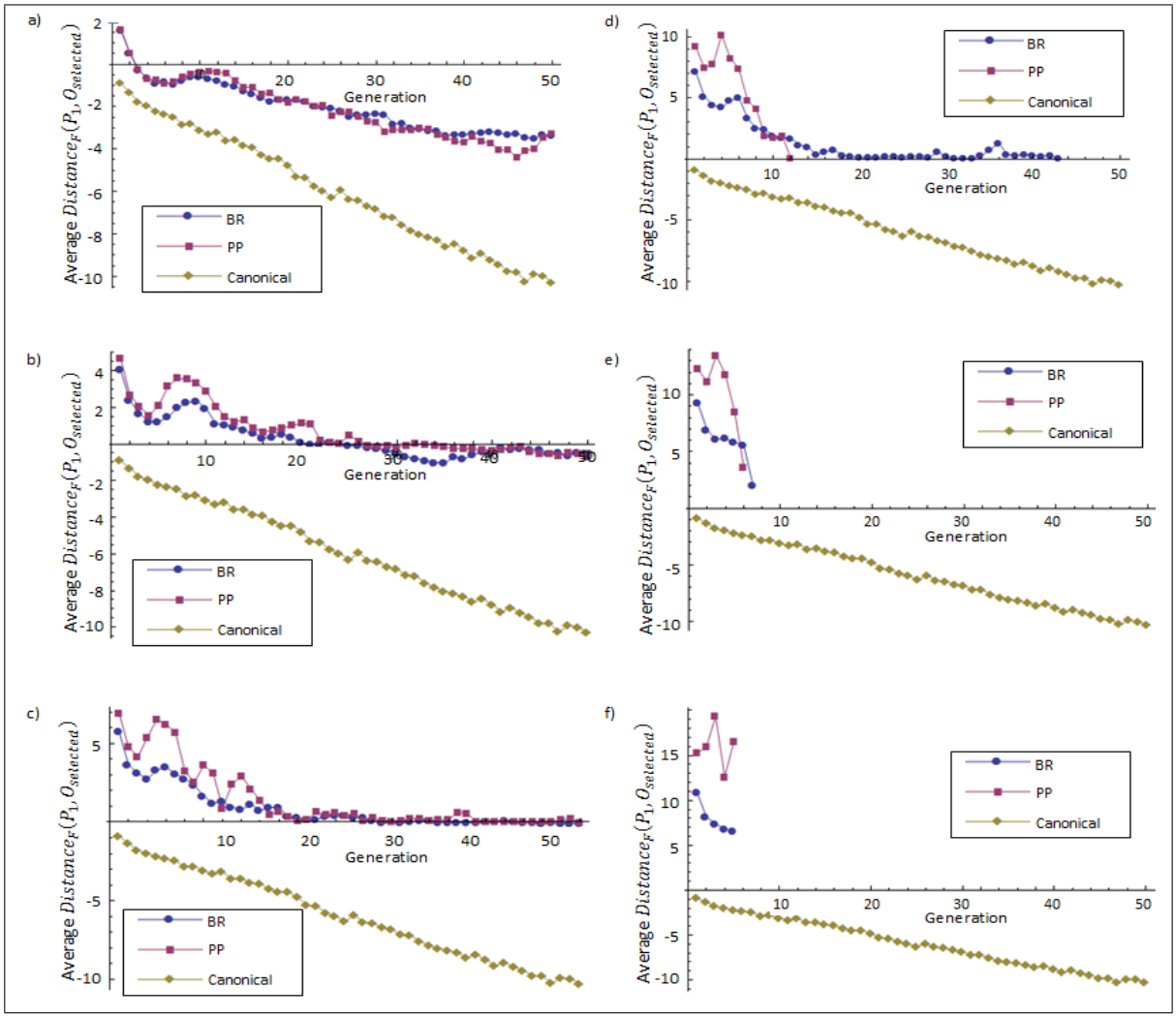

Figure 11: Artificial Ant Domain – PP vs. BR and canonical GP: Average $Distance_F(P_1, O_{selected})$ over time for brood sizes: a) 4, b) 10, c) 20, d) 40, e) 100, and f) 200.

Table 9: PP vs. canonical GP and BR: significance of difference in average $Distance_F(P_1, 0_{selected})$

	Domain			
Brood size	**Ant**	**Parity**	**Multiplexer**	**Regression**
4	$p_1 = 0.000_{(s)}$ $p_2 = 0.608$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.792$	$p_1 = 0.653_{(s)}$ $p_2 = 0.918$
10	$p_1 = 0.000_{(s)}$ $p_2 = 0.064$	$p_1 = 0.000_{(s)}$ $p_2 = 0.240$	$p_1 = 0.000_{(s)}$ $p_2 = 0.802$	$p_1 = 0.000_{(s)}$ $p_2 = 0.230$
20	$p_1 = 0.000_{(s)}$ $p_2 = 0.117$	$p_1 = 0.000_{(s)}$ $p_2 = 0.006_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.728$	$p_1 = 0.000_{(s)}$ $p_2 = 0.002_{(s)}$
40	$p_1 = 0.000_{(s)}$ $p_2 = 0.138$	$p_1 = 0.000_{(s)}$ $p_2 = 0.002_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.099$	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$
100	$p_1 = 0.000_{(s)}$ $p_2 = 0.050_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.029_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.092$	$p_1 = 0.000_{(s)}$ $p_2 = 0.001_{(s)}$
200	$p_1 = 0.000_{(s)}$ $p_2 = 0.000_{(s)}$	$p_1 = 0.000_{(s)}$ $p_2 = 0.092$	$p_1 = 0.000_{(s)}$ $p_2 = 0.075$	$p_1 = 0.000_{(s)}$ $p_2 = 0.001_{(s)}$

than their parents.

In Fig. 11 we further observe that offspring delivered by $P_B(n)$ are generally fitter than the same for both $R_B(n)$ and the canonical GP across brood sizes and domains. This is especially true for the larger brood sizes. This is a further empirical confirmation of the observations in Fig. 10. It is also an empirical confirmation of hypothesis 3b. The Regression, Multiplexer and Parity domains presented similar empirical results.

We conducted one-way ANOVA statistical tests to verify that given the same brood size, PP consistently yields fitter offspring than both BR and the canonical GP. The results are shown in Table 9.

In Table 9, given a domain and a fixed brood size, p_1 is the p-value indicating the significance of the difference in the average value of $Distance_F(P_1, O_{selected})$ between PP and the canonical GP. Similarly p_2 is the p-value indicating the significance of the difference in the average value of $Distance_F(P_1, O_{selected})$ between PP and the BR algorithm with the same brood size. In Table 9 the results that are significant (at $\alpha = 0.05$) are marked with an (s) symbol.

From Table 9, we observe that all the values of p_1 are equal to 0.000. Thus we are 99% confident that PP consistently produces fitter offspring than the canonical GP, for all brood sizes across all the four domains. This empirically supports hypotheses 1 and 3b in Section 4. Also From table 5.7 we observe that for the largest two brood sizes, PP returns significantly fitter offspring than BR, across all four domains. Overall the values of p_2 show significance for 50% of the brood sizes in the Artificial Ant Domain, over 80% of the brood sizes in the Even-5 Parity Domain, 50% of the brood sizes in the Symbolic Regression Domain and over 60% of the brood sizes in the 11-Multiplexer Domain.

From both statistical and empirical proof we conclude that in general a PP brood returns significantly fitter offspring than a BR brood. We thus have further proof of hypotheses 2 and 3b in Section 4.

6 CONCLUSION AND FUTURE WORK

This paper presented Polyandry, a nature inspired methodology to improve evolvability in Genetic Programming. Polyandry modifies the canonical GP by implementing an operator, $P_B(n)$, to replace the typically disruptive standard crossover operator. Polyandry operates in a similar manner to Brood Recombination, a ubiquitous methodology employed in GP literature towards improving evolvability.

The Polyandry algorithm employs a dual strategy towards improving evolvability in GP. Chiefly, an aggressive search for good genetic material is conducted by incorporating structures from multiple population individuals into broods of GP offspring. Here instead of producing a pair of offspring from standard crossover, as in the canonical GP, a brood of offspring is generated by employing multiple standard crossover operations between a principal parent individual and distinct mating partners. Subsequently only the best two offspring in the resulting brood are inserted into the GP population. Polyandry conducts a wider search for good genetic material relative to both the canonical GP, and the Brood Recombination algorithm. This stems from the fact that Polyandry is more explorative of the GP genotype space. This in turn translates to Polyandry being more explorative of the GP fitness space.

The second aspect to the Polyandry strategy is a minimization of building block disruption by attempting multiple crossover points with respect to the principal parent individual. This is due to the randomness involved in selecting a crossover point in the principal parent each time a pair of offspring is created and added to a PP brood. We surmise that this aspect of the Polyandry strategy contributes to evolvability in a similar manner to the Brood Recombination algorithm.

Experiments conducted have established that Polyandry reliably discovers solutions in significantly fewer GP generations, relative to both the canonical GP and Brood Recombination. Further, Polyandry delivers a higher success rate than the canonical GP. The overall ramifications of these observations are that

for certain brood size settings, Polyandry requires less computational effort to arrive at a global optimum GP solution compared to the canonical GP. Further, given a fixed brood size, Polyandry consistently requires less computational effort to arrive at a global optimum solution with respect to Brood Recombination. We also obtained empirical evidence demonstrating that Polyandry consistently exhibits better evolvability than both the canonical GP and Brood Recombination.

In future we hope to investigate the extent to which Polyandry solves the problem of premature convergence in GP. This arises from the success of Polyandry at improving evolvability as well as the notion of the connection between premature convergence and evolvability. A GP that has converged prematurely has lost the ability to produce offspring that are fitter than their parents. By definition of evolvability, such a GP system fails to maintain evolvability.

In future we would also like to explore the concept of dynamic environments or seasons in GP, in conjunction with Polyandry, to minimize the occurrence of premature convergence. Natural evolution is open-ended and makes use of the phenomena of dynamic environments and Polyandry to constantly maintain evolvability. We aim to investigate if the same can translate to a GP computational model.

Ultimately we aim to examine whether the advantages of Polyandry in Genetic Programming will translate to more novel Evolutionary Algorithm methodologies, such as Gene Expression Programming and Multi Expression Programming.

ACKNOWLEDGMENTS

The authors would like to acknowledge the reviewers for their comments that helped to improve this manuscript.

REFERENCES

[1] J. R. Koza. *Genetic Programming: vol. 1, On the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[2] L. Altenberg. "The evolution of evolvability in genetic programming". *Advances in genetic programming*, vol. 3, pp. 47–74, 1994.

[3] G. P. Wagner and L. Altenberg. "Perspective: Complex adaptations and the evolution of evolvability". *Evolution*, pp. 967–976, 1996.

[4] H. Majeed and C. Ryan. "Using context-aware crossover to improve the performance of GP". In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 847–854. ACM, 2006.

[5] L. Altenberg. "Emergent phenomena in genetic programming". In *Evolutionary Programming—Proceedings of the Third Annual Conference*, pp. 233–241. World Scientific Publishing, 1994.

[6] C. W. Fox and C. M. Rauter. "Bet-hedging and the evolution of multiple mating". *Evolutionary Ecology Research*, vol. 5, no. 2, pp. 273–286, 2003.

[7] D. Hosken and W. Blanckenhorn. "Female multiple mating, inbreeding avoidance, and fitness: it is not only the magnitude of costs and benefits that counts". *Behavioral Ecology*, vol. 10, no. 4, pp. 462–464, 1999.

[8] M. D. Jennions and M. Petrie. "Why do females mate multiply? A review of the genetic benefits". *Biological Reviews*, vol. 75, no. 1, pp. 21–64, 2000.

[9] J. Kozlowski and S. C. Stearns. "Hypotheses for the production of excess zygotes: models of bet-hedging and selective abortion". *Evolution*, pp. 1369–1377, 1989.

[10] Y. Yasui. "A 'good-sperm' model can explain the evolution of costly multiple mating by females". *The American Naturalist*, vol. 149, no. 3, pp. 573–584, 1997.

[11] Y. Yasui. "The genetic benefits of female multiple mating reconsidered". *Trends in Ecology & Evolution*, vol. 13, no. 6, pp. 246–250, 1998.

[12] A. R. Templeton. "A frequency dependent model of brood selection". *American Naturalist*, pp. 515–524, 1979.

[13] S. D. Newcomer, J. A. Zeh and D. W. Zeh. "Genetic benefits enhance the reproductive success of polyandrous females". *Proceedings of the National Academy of Sciences*, vol. 96, no. 18, pp. 10236–10241, 1999.

[14] H. Kitano. "Biological robustness". *Nature Reviews Genetics*, vol. 5, no. 11, pp. 826–837, 2004.

[15] M. Pigliucci. "Is evolvability evolvable?" *Nature Reviews Genetics*, vol. 9, no. 1, pp. 75–82, 2008.

[16] P. Marrow, M. Heath and I. I. Re. "Evolvability: Evolution, computation, biology". In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program (GECCO-99 Workshop on Evolvability)*, pp. 30–33. Citeseer, 1999.

[17] J. H. Holland. "Building blocks, cohort genetic algorithms, and hyperplane-defined functions". *Evolutionary computation*, vol. 8, no. 4, pp. 373–391, 2000.

[18] A. W. Ragalo and N. Pillay. "A building block conservation and extension mechanism for improved performance in Polynomial Symbolic Regression tree-based Genetic Programming". In *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*, pp. 123–129. IEEE, 2012.

[19] R. Poli and N. F. McPhee. "General schema theory for genetic programming with subtree-swapping crossover: Part II". *Evolutionary Computation*, vol. 11, no. 2, pp. 169–206, 2003.

[20] W. A. Tackett and A. Carmi. "The unique implications of brood selection for genetic programming". In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 160–165. IEEE, 1994.

[21] W. A. Tackett. *Recombination, selection, and the genetic construction of computer programs*. Ph.D. thesis, University of Southern California, 1994.

[22] M. Zhang, X. Gao and W. Lou. "Experiments on brood size in GP with brood recombination crossover for object recognition". Tech. rep., Victoria University of Wellington, 2006.

[23] M. Keijzer and V. Babovic. "Dimensionally aware genetic programming". In *Proceedings of the Genetic and Evolutionary computation Conference*, vol. 2, pp. 1069–1076. 1999.

[24] J. A. Zeh and D. W. Zeh. "The evolution of polyandry II: post–copulatory defenses against genetic incompatibility". *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 264, no. 1378, pp. 69–75, 1997.

[25] T. Tregenza and N. Wedell. "Polyandrous females avoid costs of inbreeding". *Nature*, vol. 415, no. 6867, pp. 71–73, 2002.

[26] T. Philippi and J. Seger. "Hedging one's evolutionary bets, revisited". *Trends in Ecology & Evolution*, vol. 4, no. 2, pp. 41–44, 1989.

[27] N. X. Hoai, R. I. McKay and D. Essam. "Representation and structural difficulty in genetic programming". *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 2, pp. 157–166, 2006.

[28] R. Poli and W. B. Langdon. "On the search properties of different crossover operators in genetic programming". *Genetic Programming*, pp. 293–301, 1998.

[29] C. Igel and K. Chellapilla. "Investigating the influence of depth and degree of genotypic change on fitness in genetic programming". In *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, pp. 1061–1068. 1999.

[30] N. F. McPhee, B. Ohs and T. Hutchison. "Semantic building blocks in genetic programming". In *Genetic Programming*, pp. 134–145. Springer, 2008.

[31] C. Gathercole, P. Ross et al. "Small populations over many generations can beat large populations over few generations in genetic programming". *Genetic programming*, vol. 97, 1997.

[32] M. Walker, H. Edwards and C. Messom. "Success effort and other statistics for performance comparisons in genetic programming". In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 4631–4638. IEEE, 2007.

[33] E. K. Burke, S. Gustafson and G. Kendall. "Diversity in genetic programming: An analysis of measures and correlation with fitness". *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 1, pp. 47–62, 2004.

[34] M. Keijzer. "Efficiently representing populations in genetic programming". In *Advances in genetic programming*, pp. 259–278. MIT Press, 1996.

[35] N. A. Weiss. *Elementary statistics*. Pearson, 5th ed. edn., 1989.

[36] K. Chellapilla, J. R. Koza and W. Banzhaf. "A preliminary investigation into evolving modular programs without subtree crossover". In *Genetic Programming 1998: Proceedings of the Third*, pp. 23–31. Morgan Kaufmann, 1998.

[37] M. Oltean. "Solving even-parity problems using traceless genetic programming". In *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, pp. 1813–1819. IEEE, 2004.

[38] J. F. Miller and P. Thomson. "Cartesian genetic programming". In *Genetic Programming*, pp. 121–132. Springer, 2000.

[39] W. B. Langdon. "Size fair and homologous tree crossovers for tree genetic programming". *Genetic programming and evolvable machines*, vol. 1, no. 1-2, pp. 95–119, 2000.

[40] S. Christensen and F. Oppacher. "An analysis of Koza's computational effort statistic for genetic programming". In *Genetic programming*, pp. 182–191. Springer, 2002.

[41] G. Casella and R. L. Berger. *Statistical inference*. Duxbury Press Belmont, CA, 2nd ed. edn., 1990.