# Bayesian Inference to Sustain Evolvability in Genetic Programming

Ahmed Kattan[1] and Yew-Soon Ong[2]

[1] AI Real-World Application Lab, UQU, Saudi Arabia
[2] School of Computer Engineering, Nanyang Technological University, Singapore
ajkatta@uqu.edu.sa, asysong@ntu.edu.sg

**Abstract.** This paper proposes a new framework, referred to as Recurrent Bayesian Genetic Programming (rbGP), to sustain steady convergence in Genetic Programming (GP) (i.e., to prevent premature convergence) and effectively improves its ability to find superior solutions that generalise well. The term 'Recurrent' is borrowed from the taxonomy of Neural Networks (NN), in which a Recurrent NN (RNN) is a special type of network that uses a feedback loop, usually to account for temporal information embedded in the sequence of data points presented to the network. Unlike RNN, our algorithm's temporal dimension pertains to the sequential nature of the evolutionary process itself, and not to the data sampled from the problem solution space. rbGP introduces an intermediate generation between each subsequent generation in order to collect information about the offspring's fitness distribution of each parent. Placing the collected information into a Bayesian model, rbGP predicts the probability of any individual to produce offspring fitter than its parent. This predicted probability (calculated by the Bayesian model) is used by the tournament selection instead of the original fitness value. Empirical evidence, from 13 problems, against canonical GP, demonstrates that rbGP preserves generalisation in most cases.

## 1 Introduction

In our previous work [5], we introduced a new framework for Genetic Algorithm (GA) referred to as *Recurrent Genetic Algorithms* (RGA). RGA guided the evolutionary process of GA using a reverse form of fitness inheritance [3]. Smith et al. [13] first introduced the technique of fitness inheritance identifying two types of inheritance: the first which takes the average of the fitness values of the two parents and the second takes the weighted average according to the similarity between offspring and their parents. While the standard notion of fitness inheritance presented as a reward for offspring based on their parents' performance (assuming a level of smoothness in the search space), RGA uses a reversed concept of the standard fitness inheritance in which the parents' fitness values are readjusted based on the fitness of their offspring, thus presenting an indication of individuals' level of evolvability. To this end, RGA uses an intermediate population (called $\hat{P}$) between each subsequent generation. This intermediate population is used as a feedback loop that recurrently adjusts the fitness values of individuals in population $P$, at the $i^{th}$ generation, based on the fitness of their offspring in population $\hat{P}_i$ (i.e., the intermediate population). Empirical evidence illustrated that this recurrent process of fitness adjustment reinforces the evolvability of subsequent generations by ensuring that parents at $P_i$ are rewarded for producing fit offspring and then given a second chance to reproduce.

In this work, we extend the RGA framework presented in [5] to Genetic Programming (GP) [12] and present a new framework referred to as *Recurrent Bayesian Genetic Programming* (rbGP). rbGP, also, introduces an intermediate population between each subsequent generation. However, away from the reversed fitness inheritance concept adopted by RGA, rbGP uses a Bayesian model [4] to readjust individuals' fitness values based on their probability to produce fitter offspring. To this end, rbGP forces each selected individual in population $P_i$, where $i$ is the number of generation, to produce $k$ number of offspring to generate population $\hat{P}_i$ (i.e., the intermediate population). Hence, rbGP collects information about the offspring fitness distribution of each selected parent and utilise this information to build a Bayesian model. rbGP employs the Bayesian model as method of inference to readjust the fitness values of individuals in population $P_i$. Thus, rbGP uses each individual's probability of producing fitter offspring (as measured by the $k$ offspring when generating the intermediate population), and the likelihood of the population to produce fitter offspring, in the Bayesian model to rank individuals. To this end, individuals that may lead the search to premature convergence (i.e., their immediate fitness gain may not lead to long-term improvement in the search) receive lower rankings in order to prevent sudden premature convergence. Details of this process are provided in Section 3. In this paper, the term 'successful parent' will refer to parents that can produce fitter offspring.

This paper is organised into six sections. Section 2 reviews some related works. Section 3 explains rbGP in detail. Sections 4 and 5 discuss the experimental setup and the results, respectively. Finally, some conclusive remarks and future directions for research are presented in Section 6.

## 2 Related-Work

As mentioned earlier, rbGP uses Bayesian model as a method of inference to readjust the fitness values of individuals in order to prevent premature convergence. The whole process adopted by rbGP sustains evolvability. Therefore, the literature review focuses on previous works related to works that define the concept of evolvability and Bayesian models in GP.

### 2.1 Evolvability

The notion of "evolvability" is defined as *"the ability of a population to produce variants fitter than any yet existing"* [1]. Hence, generally, the choice of selection, search operator and representation is vital to the performance of GP because they control the creation of new individuals throughout the evolutionary process. One aim of researchers in the Evolutionary Computation (EC) field is to discover new methods for increasing evolvability of evolutionary systems. The term evolvability does not only refer to how often offspring are fitter than their parents but also to the entire distribution of fitness values among offspring produced by a group of parents [1].

The concept of evolvability has been an active research area in both evolutionary biology and computer science for the past several decades. Hu and Banzhaf in [9] have argued that adopting new knowledge about natural evolution generated in areas such as molecular genetics, cell biology, developmental biology, and evolutionary biology would benefit the field of evolutionary computation. The authors discussed evolvability and methods for accelerating artificial evolution by introducing notions from biology and their potential in designing new algorithms in EC.

It has been recorded that the evolvability property has good effect on the search process. For example, in [2], the authors suggested that evolvability can effectively reduce the bloat in evolutionary algorithms that use variable length representations. In their work, the authors noted the similarity of bloat causes and evolvability theory, thus, they argue that reproductive operators with high evolvability will be less likely to cause bloat.

With the importance of evolvability as a research topic, several measurements have been proposed to quantify it. Wang and Wineberg [14], suggested two measures of evolvability one based on fitness improvement and the other based on the amount of genotypic change. The authors divided the population into three sub-populations, where the size of each sub-population is determined dynamically. The first sub-population uses selection based on fitness directly; the second sub-population is based on the fitness-improvement-ratio; finally, selection for the third sub-population is based on genotypic change. Each sub-population is filled by selecting chromosomes from the parent's generation under its own selection functions. Thereafter, the three sub-populations are merged, and the standard GA search operators are applied to form the next generation. Experiments with several continuous optimisation functions showed that the proposed approach has higher evolvability (and consequentially achieves better solutions) than standard GA.

Hu in [8], proposed a new measurement for evolvability called "rate of genetic substitutions". This measurement method was used to investigate the effects of four major configuration parameters in EC (namely, mutation rate, crossover rate, tournament selection size, and population size) to show the effectiveness of these parameters with respect to evolution acceleration. In his work, Hu has developed a new indicator based on this proposed measurement for adjusting population size dynamically during evolution.

### 2.2 Bayesian Models for GP

Bayesian probability model is an interpretation of the concept of probability and belongs to the category of evidential probabilities. To evaluate a hypothesis' probability, the Bayesian probability model needs to specify a prior distribution of probabilities (i.e., training data), which can then be updated in the light of new relevant data. Relatively few works in the literature have used Bayesian probability model to enhance GP process.

Zhang [16,17] proposed a Bayesian framework for GP based on the Bayesian approach in which, under GP, individuals are viewed as models of the fitness data. Bayes theorem is used to estimate the posterior probabilities of programs based on their prior probabilities and likelihood of fitness in observed cases. Offspring programs are then generated by sampling from the posterior distribution by using genetic operators. This work presented two methods for Bayesian GP: 1) GP with the adaptive Occam's razor designed to evolve parsimonious program, and 2) GP with incremental data inheritance designed to accelerate evolution by active selection of training cases. All these methods are implemented as adaptive fitness functions that take into account the dynamics of evolutionary processes.

Yanai and Iba [15] proposed Estimation of Distribution Programming (EDP) based on GP extension. In their work, a probability distribution expression using a Bayesian network was used to generate individuals instead of standard search operators and the Bayesian network described the dependency relationship of probabilistic nodes. Truncation selection selects the individuals with the top fitness, analyses their structure, and estimates the probability distribution of these superior individuals is estimated. Later, Hasegawa and Iba [7] introduced a new tree-like program evolution algorithm employing a Bayesian network for generating new individuals. It employs a special chromosome called the expanded parse tree, which significantly reduces the size of the conditional probability table.

As can be seen, most previous works used a Bayesian model or Bayesian network as a method to create individuals, in a similar manner to Estimated Distribution Algorithms (EDA). In this paper, we utilise the Bayesian model to enhance the GP evolutionary process in a novel way that, to the best of our knowledge, has never been proposed before. To this end, Bayesian rule is used to prevent premature convergence that could occur in the canonical GP iteration process. Further details of this process are provided in Section 3.

## 3 Recurrent Bayesian Genetic Programming

Generally, premature convergence occur because selection pressure may encourages dense congregations of homogeneous solutions, a key characteristic of premature convergence [10]. It is reasonable to hypothesise that mature convergence is inhibited by the loss of potentially useful genetic material due to the replacement strategies undertaken by search operators wherein worst individuals are replaced by new offspring. This dynamic may allow some individuals, that seems potential, to exist for multiple evolutionary cycles within the population and may hinder the exploration of superior areas in the search space. rbGP techniques aim to reduce premature convergence by ranking individuals in the population based on their level of evolvability relative to the performance of the whole population.

The process adopted by rbGP is broadly outlined in figure 1. Similar to standard GP, rbGP starts by randomly initialising a population $P_0$, where the number of generations is $i = \{0, ..., n\}$, and calculates their fitness values using the given fitness measure. However, unlike standard GP, instead of driving the population to generate the next generation, rbGP generates an intermediate population $\hat{P_i}$ to collect observations about the population's performance. To this end, rbGP applies standard tournament selection to identify potential individuals where it forces each individual to generate $k$ number of offspring to constitute $\hat{P_i}$ (i.e., the intermediate population). Hence, the size of the $\hat{P_i} = k \times size(P_i)$. rbGP uses standard tournament selection to select the parents of the individuals in $\hat{P_i}$. Naturally, some individuals in $P_i$ might never be selected while other individuals might be selected more than once. During the creation process of $\hat{P_i}$, rbGP notes the number of successful and unsuccessful offspring generated by each selected individual participated in $\hat{P_i}$. By 'successful' offspring we mean the ones that their fitness values are better than their parents while 'unsuccessful' means the opposite. Using the collected observations, rbGP builds two sets for the population $P_i$. First, *numbers of successful offspring* and second, *numbers of failure offspring.* Let the set of successful offspring for $P_i$ be represented as $D_s(P_i) = \{|o_0^s|, |o_1^s|, ..., |o_m^s|\}$ where $|o_j^s|$ is the number of successful offspring generated by the $j^{th}$ individual participated in generating $\hat{P_i}$ and $m$ is the population size. In addition, let the set of failure offspring of $P_i$ be denoted as $D_f(P_i) = \{|o_0^f|, |o_1^f|, ..., |o_m^f|\}$ where $|o_j^f|$ is the number of failure offspring generated by the $j^{th}$ selected individual from $P_i$. Note that both sets $D_s(P_i)$ and $D_f(P_i)$ are built using information from the selected individuals from $P_i$ and those individuals that never selected will be ignored from both sets. The sets $D_s(P_i)$ and $D_f(P_i)$ can represent the convergence state of the population $P_i$. Hence, naturally, a high mean of $D_s(P_i)$ and a low mean of $D_f(P_i)$ may indicate that the individuals in $P_i$ are scattered in the search space and that the population remains far from the global optimum. A low mean of $D_s(P_i)$ and a high mean of $D_f(P_i)$ though, might indicate the opposite, that $P_i$ individuals are already approaching optimum solutions (perhaps a local optimum), and thus, it is difficult to find further superior offspring.

Using a Bayesian model, rbGP analyses the *probability of evolvability* (i.e., probability of success) for each individual participated in constitution of the intermediate population based on their number of successful offspring versus their number of failure offspring relative to the whole population and ranks them accordingly. This ranking process is calculated as following:

$$P(I_j^f|P_{if}) = \frac{P(P_{if}|I_j^f)P(I_j^f)}{P(P_{if}|I_j^f)P(I_j^f) + P(P_{is}|I_j^s)P(I_j^s)} \tag{1}$$

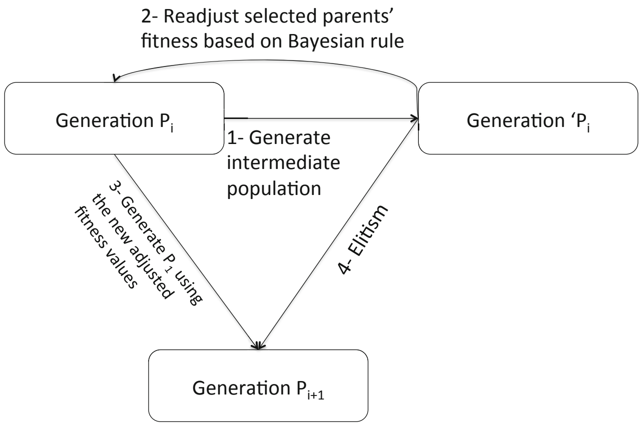
2- Readjust selected parents' fitness based on Bayesian rule
Generation P_i
1- Generate intermediate population
Generation 'P_i
3- Generate P_1 using the new adjusted fitness values
4- Elitism
Generation P_i+1

**Fig. 1.** rbGP process outline

where $P(I_j^f)$ is the probability of the $j^{th}$ individual to produce inferior offspring. This variable is calculated as $\frac{|o_j^f|}{|o_j^f|+|o_j^s|}$. The term $P(P_{if}|I_j^f)$ refers to the likelihood of individual $I_j$ produce inferior offspring given $P_{if}$ (i.e., $P_{if}$ refer to the probability of the whole population $P_i$ to produce inferior offspring). The term $P(I_j^s)$ refer to the probability of the individual $I_j$ producing better offspring. The $P(I_j^s)$ variable can be calculated as $\frac{|o_j^s|}{|o_j^f|+|o_j^s|}$. Finally, $P(P_{is}|I_j^s)$ is the likelihood of individual $I_j$ produce better offspring given $P_{is}$ (i.e., $P_{is}$ refer to the probability of population $P_i$ producing better offspring).

We assumed that the set $D_f(P_i)$ have a Gaussian distribution and that the likelihood is calculated as follows:

$$P(P_{if}|I_j^f) = \frac{1}{\sqrt{2\pi\sigma_f}} e^{\frac{(|o_j^f|-\mu_f)^2}{2\sigma_f}} \tag{2}$$

where $\mu_f$ and $\sigma_f$ denote the mean and variance of $D_f(P_i)$, respectively. Also, the likelihood of $P(P_{if}|I_j^f)$ in $D_s(P_i)$ is calculated in similar manner.

$$P(P_{is}|I_j^s) = \frac{1}{\sqrt{2\pi\sigma_s}} e^{\frac{(|o_j^s|-\mu_s)^2}{2\sigma_s}} \tag{3}$$

where $\mu_s$ and $\sigma_s$ denote the mean and variance of $D_s(P_i)$, respectively. Now, according to Equation 1, individuals are ranked based on their probability of evolvability in relation to the whole population. Individuals that have a higher potential evolvability level than the whole population receive lower ranks. To this end, rbGP readjusts individuals' fitness values according to their probability of evolvability relative to the whole population and their fitness values as:

$$Rank(I_j) = P(I_j^f|P_{if}) \times fitness(I_j) \tag{4}$$

Individuals that have been ignored by the selection process during generating the intermediate population will automatically receive rank of 0. This is because the system has no information about their level of evolvability.

The main disadvantage of rbGP is that it requires to produce several offspring for each selected parent when generating the intermediate generation (in our case it is 100 offspring) in order to constitute meaningful distributions for $D_f(P_i)$ and $D_s(P_i)$ which may be computationally expensive. However, as we will see in the experiments in Section 5, rbGP managed to evolve good solutions in small number of generations.

### 3.1 Elitism

The ranking process, described in Equation 4, could underestimate some potential solutions, that appear in early stages of the search, and assign them lower ranks, which will reduce their chances of participating in $P_{i+1}$, thus hindering progress of the search. Therefore, rbGP copies the best individual from $P_i$ to $P_{i+1}$ to preserve potentially useful genetic material from being lost.

In addition, as illustrated in figure 1, rbGP copies the best individuals from $\hat{P}_i$ to $P_{i+1}$. The logic for this is that rbGP has already devoted considerable computational efforts to generate the intermediate population ($\hat{P}_i$) and it is reasonable to utilise this efforts in the search process.

## 4 Experimental Setup

Experiments have been devised to compare the proposed rbGP model against standard GP. The main aim of the experiments is to evaluate the performance of the rbGP and to assess the algorithms behaviour under a variety of circumstances. Our experimental study included 15 problems, 12 symbolic regression problems and 3 time-series problems. The symbolic regression covered a variety of functions: polynomial, trigonometric, logarithmic and square-root and complex functions. These functions selected because they represent different landscapes. Thus we stress rbGP under different search conditions. For each symbolic regression problem, we uniformly sampled 200 data points from the interval $[-5, 5]$. These points were divided into 50 for training, 50 for validation, and 100 for testing. Table 1 shows the problems included in our experimental study. GP evolved solutions to minimise the average absolute error on the training set. The best individual in each generation is further tested on the validation set and the best individual across the whole run (i.e., the one with best performance on the validation set) is tested with the testing set.

For the real-world time-series problem, we used data from *Google Trends* service [6], a free service offering data about the search terms that people enter into Google's search engine. The service provides free downloadable historical time-series data about any keyword. It, also, offers the flexibility to restrict the search by country. One use of Google Trends is for E-Marketing managers to monitor how often people type certain keywords related to their products at different times of the year. Using this information, E-Marketing managers can determine the best time to release their marketing campaigns so their advertisements coincide with peoples searches and eventually achieve higher hit rates. For the purpose of our experiments, we imported time-series data about searches for the following keywords; *Jobs, Holidays,* and *Cinema* and we restricted the search to get data from *USA, USA* and *UK*, respectively. All the imported data from Google Trends represent the weekly frequencies of these keywords between January 2004 and May 2013. The data yielded 490 data points. We used a sliding window of size 5 to capture the average values of 5 consecutive weeks, which was input to GP in order to predict the value of the next week. GP received inputs of averages for weeks $w_i : w_{i+5}$ where $i = \{1...490\}$ and the expected output is the value of point in location $w_{i+6}$. Data were divided into 50% Training, 20% Validation, and 30% Testing sets. Here, the aim is to predict the frequency of keywords searches (treating the testing set as unseen weekly observations in the future) so as to help employers to select the best time to advertise their new jobs, travel agencies to predict the best time to release holidays packages, and media companies to preview new shows at the most appropriate time.

We compared rbGP against standard GP (SGP). Both systems received exactly the same settings and the same number of evaluations, to ensure fair comparison, as illustrated in table 2. In our experiments, we considered the number of consumed evaluations in the intermediate generations in rbGP and allocated exactly the same evaluation budget to SGP. To this end, rbGP was set to search

**Table 1.** Test problems included in the experimental study

| *Problem* | *Notation* | *Type* | *Variables* |
|---|---|---|---|
| F0 | $f(x) = 5x^3 + 2x^2 + x + 5$ | Polynomial | 1 |
| F1 | $f(x) = 5x^2 + 2x^2 + x$ | Polynomial | 1 |
| F2 | $f(x) = tan(x) + sin(x)$ | Trigonometric | 1 |
| F3 | $f(x) = 5\sqrt{(}\lvert x\rvert)$ | Square root | 1 |
| F4 | $f(x) = 1 - log(x^2 + x + 1)$ | Logarithmic | 1 |
| F5 | $f(x) = \frac{1}{100+log(x^2)+\sqrt{\lvert x\rvert}}$ | Logarithmic | 1 |
| F6 | $f(x) = log(x^3)$ | Logarithmic | 1 |
| F7 | $f(x, y) = sin(atan(y, x)\sqrt{x^2 + y^2} \times 6\pi)$ | Complex | 2 |
| F8 | $f(x) = 5x^4 + 5x^3 + 2x^2 + x + 5$ | Polynomial | 1 |
| F9 | $f(x, y) = (1.5 - x + xy)^2 + (2, 25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$ | Complex | 2 |
| F10 | $f(x_n) = 10 \times \sum_n^{i=1} {x_i}^2 - 10 \times cos(2\pi x_i)$ | Complex | 5 |
| F11 | $f(x_n) = 10 \times \sum_n^{i=1} {x_i}^2 - 10 \times cos(2\pi x_i)$ | Complex | 10 |
| F12 | Time-Series, Google Trends (Jobs, USA) | Prediction | 490 |
| F13 | Time-Series, Google Trends (Holidays,USA) | Prediction | 490 |
| F14 | Time-Series, Google Trends (Cinema, UK) | Prediction | 490 |

**Table 2.** Parametric settings of the algorithms considered in the experiments

| *Parameter* | *Standard GP Setting* |
|---|---|
| Sub-tree Mutation | 70% |
| Sub-tree Crossover | 30% |
| Tournament size | 2 |
| Population Size | 5050 |
| Generations | 20 |
| Elitism | 2% |
| *Parameter* | *rbGP Settings* |
| Sub-tree Mutation | 70% |
| Sub-tree Crossover | 30% |
| Tournament size | 2 |
| Population Size | 50 |
| Intermediate Population Size | 5000 (100 offspring for each parent) |
| Generations | 20 |
| Elitism ($\hat{P}_i$) | 1% |
| Elitism ($P_i$) | 1% |

the search space using 50 individuals. For each selected individual, rbGP generates 100 different offspring to constitute the intermediate generation. SGP was set to search the search space using 5050 individuals. Thus, received exactly the same search budget as rbGP. For every problem, we tested each system through 50 independent runs.

## 5 Results

Tables 3, 4 and 5 summarise 1500 independent runs. As stated, for each problem, we tested and compared each system in 50 independent runs and report the *mean* and *median* of the best evolved solutions by each system. In addition, we report the *best* solution found by each system across the whole 50 runs. As can be seen, for the symbolic regression problems, rbGP achieved the best mean and median in 7 problems and the best solution in 8 problems. Both rbGP and SGP almost have similar performance in problem $F7$. We observed that in the cases that rbGP outperforms its competitor, it finds solutions better than SGP at margins varying from 0.02% to 54%. Note that results are obtained from the performance of the best tree on an unseen testing set to reflect the generalisation ability of each system. To further verify the statistical significance of our results, a Kolmogorov-Smirnov two-sample test [11] has been performed. Table 6 reports the *P-value* for the tests. In all cases where rbGP achieved better results, P-value is statistically significantly superior to SGP at the standard 95% significance level. Interestingly, the P-value also show a statistical significance in the three problems where rbGP was outperformed by SGP (namely, $F2$, $F10$, and $F11$).

**Table 3.** Summary results of 600 independent runs (for problems F0 - F5). Results are sampled from 50 independent runs form each system in each problem.

| F0 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | **12.558** | **5.252** | **0.469** |
| GP | 22.628 | 16.985 | 3.533 |

| F1 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | **1.254** | **3.342E-01** | **4.586E-05** |
| GP | 1.882 | 1.900E+00 | 0.318 |

| F2 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | 2.532 | 6.693E-01 | 0.001 |
| GP | **1.748** | **6.374E-01** | 0.001 |

| F3 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | **0.009** | **2.457E-06** | **8.839E-07** |
| GP | 0.020 | 1.159E-02 | 6.018E-05 |

| F4 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | **0.232** | **0.174** | **0.052** |
| GP | 0.350 | 0.329 | 0.159 |

| F5 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | **0.329** | **0.309** | **0.060** |
| GP | 0.493 | 0.482 | 0.286 |

***Bold** numbers are the lowest.

**Table 4.** Summary results of 600 independent runs (for problems F6 - F11). Results are sampled from 50 independent runs form each system in each problem.

| F6 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | **0.292** | **0.289** | **0.038** |
| GP | 0.296 | 0.292 | 0.170 |

| F7 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | **0.866** | 0.867 | 0.762 |
| GP | 0.868 | 0.867 | **0.737** |

| F8 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | 163.436 | **48.896** | **3.898** |
| GP | **115.343** | 111.453 | 13.628 |

| F9 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | 53430.451 | 13927.500 | **848.112** |
| GP | **13706.556** | **12422.200** | 4300.640 |

| F10 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | 30.875 | 24.376 | 15.949 |
| GP | **21.714** | **18.614** | **15.789** |

| F11 | | | |
|---|---|---|---|
| | Mean | Median | Best |
| rbGP | 37.730 | 36.177 | 26.759 |
| GP | **33.159** | **33.377** | **25.031** |

***Bold** numbers are the lowest.

The only two cases that P-Value shows statistical significance below 95% is in problems $F6$ and $F7$.

For the time-series prediction problems, rbGP served best as measured by mean and median in 2 out of 3 problems and achieved the best solution in only one problem. rbGP's improvement margins varying from 0.02% to 6% and the loss margins from 0.04% to 0.007%. Generally, both SGP and rbGP achieved almost equal performance on the three time-series prediction problems.

## 5.1 Discussion

The results are encouraging in the sense that rbGP is already doing well compared to SGP and it is not too far behind in the cases that it loses the compression. We believe rbGP still has room for further improvements. Apart from

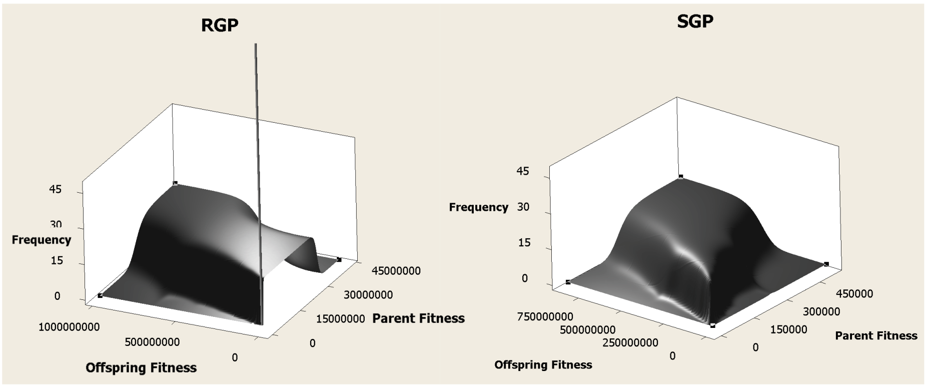

**Fig. 2.** Fitness distribution for problem $F0$ accumulated from 50 independent runs. The graph show the fitness values of selected parent against the fitness values of their offspring and the frequency of their occurrence in the search process.

rbGP's good generalisation ability, we noted that rbGP is outperforming SGP in all the experiments on the training cases. Naturally, the performance on unseen testing data is much more important than the performance on any given training set. However, it should be noted that despite being competitive on unseen testing sets, rbGP shows remarkable resistance to the over-fitting problem.

Surprisingly, we observed that rbGP produces larger trees, in some cases, than SGP. This is interesting because rbGP effectively explores the search space using 50 individuals while using the intermediate generation as indicator re-rank individuals based on their level of evolvaiblity (as described in Section 3) while SGP explores the search space using 5050 individuals (the same exploration budged allocated to rbGP). Therefore, it is natural to assume that SGP's population will bloat faster. However, as the demonstrated by the experiments, this is not the case. We believe that rbGP bloat at faster rate than its competitor for two reasons. The first reason, as stated in Section 3.1, is because rbGP copies the best individuals from the intermediate generation $\hat{P}_i$ to $P_{i+1}$. This can accelerate the bloat. The second reason is that the whole process undertaken by rbGP ensures to enhance the evolvability. The fitness improvement, generally, occur across the whole population. This is further confirmed in figure 2 where the fitness distribution of both rbGP and SGP for problem $F0$ is visualised. The figure shows the fitness values of the selected parents against the fitness values of their offspring and the frequency of their occurrence in the search process, accumulated from 50 runs. For SGP (on the left side of the figure), the figure shows that SGP's search, generally, dominated by poor parents that produce poor offspring and then some parents produce good offspring which make the search converges toward an optimum. In SGP most of the search budget is wasted in areas where poor parents produce poor offspring. However, in rbGP (on the right side of the figure) parents that produce good offspring are more and then the whole population become dominated by good individuals (shown by the large peak on

**Table 5.** Summary results of 300 independent runs (for problems $F12$ - $F14$). Results are sampled from 50 independent runs for each system in each problem.

| Jobs - US | | | | |
|---|---|---|---|---|
| | Mean | Best | Meidean | StD |
| rbGP | **1.127** | 1.102 | **1.133** | 0.012 |
| GP | 1.133 | 1.100 | 1.136 | 0.011 |

| Holidays - US | | | | |
|---|---|---|---|---|
| | Mean | Best | Meidean | StD |
| rbGP | 3.901 | **3.707** | 3.908 | 0.117 |
| GP | 3.898 | 3.808 | 3.908 | 0.037 |

| Cinema - UK | | | | |
|---|---|---|---|---|
| | Mean | Best | Meidean | StD |
| rbGP | **9.051** | **8.449** | **9.039** | 0.334 |
| GP | 9.096 | 8.632 | 9.050 | 0.226 |

***Bold** numbers are the lowest.

**Table 6.** Summary of Kolmogorov-Smirnov two-sample test.

| *Problem* | *P-Value* |
|---|---|
| F0 | **2.18E-05** |
| F1 | **2.59E-08** |
| F2 | **4.23E-04** |
| F3 | **7.23E-16** |
| F4 | **3.63E-06** |
| F5 | **1.08E-08** |
| F6 | 0.3584 |
| F7 | 0.9541 |
| F8 | **1.39E-08** |
| F9 | **6.43E-01** |
| F10 | **1.26E-07** |
| F11 | **4.89E-05** |
| F12 | 0.8253 |
| F13 | **0.0440** |
| F14 | **0.0440** |

***Bold** numbers are less than 5%.

the figure's corner). This indicates that most of the allocated search budget was well utilised and rbGP directed the search effectively. Thanks to the re-ranking process of parents' fitness values.

## 6 Conclusions and Future Work

In this paper, introduced a new framework, referred to as Recurrent Bayesian Genetic Programming (rbGP), to sustain evovability. rbGP generates an intermediate population to collect statistical observations about the populations performance and incorporate this information into a Bayesian model. The Bayesian model is trained with the collected observations and used as a method of inference to readjust fitness values of individuals based on their performance and likelihoods of driving the population into premature convergence.

To verify the usefulness of rbGP, we conducted an experimental study that included 15 non-trivial problems. rbGP has been compared against standard GP. The results were promising in the sense that rbGP outperformed its competitor in most cases and when it lost the comparison it is not too far behind. Moreover, we believe that rbGP performance can still be improved in future work. Furthermore, results indicate the Bayesian ranking process make rbGP remarkably resistance to the over-fitting.

For the future work, one direction is to explore the idea of using a metric that establishes the best GP trees to breed from in one stage method rather than this two-stage process. Another direction to explore is eliminating the intermediate population and allowing the Bayesian model to continuously learn population distributions as new evidences emerges.

## References

1. Altenberg, L.: The evolution of evolvability in genetic programming. In: Kinnear Jr., K.E. (ed.) Advances in Genetic Programming, ch. 3, pp. 47–74. MIT Press (1994)
2. Bassett, J.K., Coletti, M., De Jong, K.A.: The relationship between evolvability and bloat. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO 2009, pp. 1899–1900. ACM, New York (2009)
3. Ducheyne, E., De Baets, B., De Wulf, R.: Is fitness inheritance useful for real-world applications? In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 31–42. Springer, Heidelberg (2003)
4. Ellison, A.M.: Bayesian inference in ecology. Ecology Letters 7(6), 509–520 (2004)
5. Fakeih, A., Kattan, A.: Recurrent genetic algorithms: Sustaining evolvability. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012. LNCS, vol. 7245, pp. 230–242. Springer, Heidelberg (2012)
6. Google. Google insights (June 2013), `http://www.google.com/trends/`
7. Hasegawa, Y., Iba, H.: A Bayesian network approach to program generation. IEEE Transactions on Evolutionary Computation 12(6), 750–764 (2008)
8. Hu, T.: Evolvability and Rate of Evolution in Evolutionary Computation. PhD thesis, Department of Computer Science, Memorial University of Newfoundland, ST. John's, Newfoundland, Canada (May 2010)
9. Hu, T., Banzhaf, W.: Evolvability and speed of evolutionary algorithms in light of recent developments in biology. J. Artif. Evol. App. 2010, 1:1–1:28 (2010)
10. Murphy, G.P.: Manipulating Convergenc. In: Evolutionary Systems. PhD thesis, University of Limerick, Ireland (May 19, 2009)
11. Peacock, J.A.: Two-dimensional goodness-of-fit testing in astronomy. Royal Astronomical Society, Monthly Notices 202, 615–627 (1983)
12. Poli, R., Langdon, W.W.B., McPhee, N.F.: Field Guide to Genetic Programming. Lulu Enterprises Uk Limited (2008)
13. Smith, R.E., Dike, B.A., Stegmann, S.A.: Fitness inheritance in genetic algorithms. In: Proceedings of the 1995 ACM Symposium on Applied Computing, SAC 1995, pp. 345–350. ACM, New York (1995)
14. Wang, Y., Wineberg, M.: The estimation of evolvability genetic algorithm. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2302–2309 (September 2005)
15. Yanai, K., Iba, H.: Estimation of distribution programming based on bayesian network. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 3, pp. 1618–1625 (2003)
16. Zhang, B.-T.: Bayesian genetic programming. In: Haynes, T., Langdon, W.B., O'Reilly, U.-M., Poli, R., Rosca, J. (eds.) Foundations of Genetic Programming, Orlando, Florida, USA, pp. 68–70 (July 13, 1999)
17. Zhang, B.-T.: Bayesian methods for efficient genetic programming. Genetic Programming and Evolvable Machines 1(3), 217–242 (2000)