Computing and Informatics, Vol. 32, 2013, 63–87

A NEW DYNAMIC POPULATION VARIATION IN GENETIC PROGRAMMING

Yanyun Tao

School of Urban Rail Transportation
Soochow University
Yangchenghu Campus, Jiexue Rd. No. 8
Xiangcheng Area, Suzhou 215137, China
e-mail: taoyanyun@suda.edu.cn

Minglu Li, Jian Cao

Computer Science, School of Electronic, Information and Electrical Engineering
Shanghai Jiao Tong University
&
Shanghai Key Laboratory of Scalable Computing and Systems
800 DongChuan Road, Minhang
Shanghai 200240, China
e-mail: cao-jian@sjtu.edu.cn

Communicated by Jiří Pospíchal

Abstract. A dynamic population variation (DPV) in genetic programming (GP) with four innovations is proposed for reducing computational effort and accelerating convergence during the run of GP. Firstly, we give a new stagnation phase definition and the characteristic measure for it. Secondly, we propose an exponential pivot function (EXP) in conjunction with the new stagnation phase definition. Thirdly, we propose an appropriate population variation formula for EXP. Finally, we introduce a scheme using an instruction matrix for producing new individuals to maintain diversity of the population. The efficacy of these innovations in our DPV is examined using four typical benchmark problems. Comparisons among the different characteristic measures have been conducted for regression problems and the proposed measure performed best in all characteristic measures. It is demonstrated that the proposed population variation scheme is superior to fixed and proportion-

ate population variation schemes for sequence induction. It is proved that the new DPV has the capacity to provide solutions at a lower computational effort compared with previously proposed population variation methods and standard genetic programming in most problems.

Keywords: Evolutionary algorithm, exponential pivot function, dynamic population variation, instruction matrix, computational effort

Mathematics Subject Classification 2010: 68T01, 68T05, 68T20

1 INTRODUCTION

Genetic programming (GP), based on the theory of biological evolution, extends the representation of a genetic algorithm (GA) with a lisp-tree structure. Koza [1] put forward both the original form of genetic programming, in which the genotype and the phenotype are both parser trees, and automatically defined function-GP (ADF-GP) for more complex system modeling. In the latter [2], a sub-program or sub-module evolves during the operation run of a GP to implement a sub function. It was proved by many researchers to be a feasible tool for solving problems in many fields such as complex system modeling [3, 4], automatic design and optimization of combinatorial circuits [5, 6, 7, 8], pattern recognition [9] and robotics research [10, 11]. One of the main drawbacks of standard type GP is that a large amount of computational effort is often required to address complex modeling problems. Computational effort is calculated as the number of fitness evaluations needed to find a solution of a problem with 99 % probability as presented by Koza. Koza's concept of computational effort $I(M, i, z)$ and three methods for producing a confidence interval for the computational effort are studied in [12]. Other research focuses on the conjunction of computational effort and steady-state algorithms instead of standard GP. It is believed that it is possible to reduce the difference between a theoretical effort value and the measured one with this approach [13].

Various researchers have investigated the effects of population size on the GP algorithm and have reported the relationship between population size and computational effort. A variable population-size genetic algorithm (VPGA) technique which introduces a "dying probability" for the individuals was proposed [14] and it is similar to the plague operator introduced in [15]. Population variation methods (PV) have been proposed for reducing computational effort [16, 17, 18]. They can be classified in two categories. One is static population variation (SPV), such as PV-R, PV-I and PV-RAN, in which the size of the population is varied according to various schemes and the population increment or decrement is constant during the run of the GP system. The disadvantage of SPV is that the constant size variation prevents the system from achieving fast convergence to a solution. The other class is dynamic population variation (DPV), such as DIV, SUB and GRAD, in which

the population is varied dynamically according to the stagnation phase definition for GP and the pivot function. It is reported that these approaches can perform satisfactorily in system modeling in terms of computational effort and they outperformed SPVs in most cases. DIV and SUB measured the stagnation phase by the best performing individual and GRAD by the mean of high-performing individuals. However, neither measure fully reflects the actual situation during the run of a GP system; the measurements are not exact. Therefore, how to exactly define the stagnation phase in a GP system and its characteristic measure is a critical issue in DPV. Population size variation is controlled by the pivot function, which is at the core of the DPV technique. DIV, SUB and GRAD define their pivot functions using division, subtraction and gradient. DIV and SUB perform poorly when using high-performing individuals as the measure of stagnation and GRAD when best performing individual is used. Hence, finding an appropriate pivot function that can exactly judge whether the GP system is in the stagnation phase is a significant task for DPV. In addition, population variation and the increment/decrement scheme also have great influence on performance of a DPV. We believe that by some innovation for dynamic population variation, the computational effort for system modeling problems can be further reduced and at the same time a system model with high accuracy can be achieved.

Our aim is to present a novel dynamic population variation method in genetic programming for computational effort reduction; this method can also provide a similar accuracy with other PV methods. The definition of computational effort and previously proposed population variation genetic programming are briefly introduced in Section 2. Section 3 describes our four innovations in dynamic population variation. In Sections 4 and 5 the method is applied to six system modeling problems including a regression problem, sequence induction, digital circuit design and an artificial ant problem. The computational effort and average number of evaluations for system modeling are compared with other methods and standard genetic programming. Implications of the current work are discussed in Section 6.

2 POPULATION VARIATION

2.1 Computational Effort

The main aim of population variation methods in genetic programming is to reduce the computational effort and possibly accelerate convergence towards a solution. A method should perform at least as well as SGP for an unsuccessful run and better for a successful run.

Computational effort (CE) and average number of evaluation (AES) are two main measures for the efficiency of genetic programming. A lower computational effort and a smaller AES are considered to be a favorable result. The computational effort as defined in this paper can be expressed (1) as follows:

$$E_g = \sum_{g=0}^{G} S(g) \cdot N \tag{1}$$

where $S(g)$ is the total number of individuals (the population size) at any generation g, N represents the number of sample points in an interval, and G represents the convergence generations for a normal run.

AES to a solution is another efficient assessment for performance of GP as well as *CE*, which is defined in [16]. In our study, ε is defined as *AES* before a successful termination. The value to the run that cannot converge is not included in *AES*. *AES* can be computed by Equation (2) as follows:

$$\varepsilon_g = \frac{1}{R} \cdot \sum_{i=0}^{R} \sum_{g=0}^{M} S(g) \cdot N, M < G_{\max} \tag{2}$$

where R is the number of successful runs of GP, $G_{\max}$ denotes the maximum number of generations, and M is the generations for convergence in a successful run.

Success rate (*SR*) is another important assessment for performance of a GP system. *SR* is defined as the rate of the number of the successful runs with the number of all the normal runs shown as Equation (3):

$$SR = \frac{\xi'}{\xi} \tag{3}$$

where ξ' indicates the number of the successful runs and ξ represents the total number of all the normal runs. It is believed that higher *SR* represents better performance of PV methods.

2.2 Previous Approaches

First, we introduce three previously proposed SPVs [16]. They are population variation reduction (PV-R), population variation increment (PV-I) and random population variation (PV-RAN).

In the PV-R scheme [16], the initial population size $S_{PV}(0)$ is larger than SGP and the population $S(g)$ in generation g is reduced during the run of GP linearly. The computational effort at the end of the run is equal to that of SGP. The PV-I scheme [16] starts off with an initial population size which is smaller than SGP and the population is increased as the generations progress. In the PV-RAN scheme [16], the population is randomly altered around a mean value around the initial size $S_{SGP}(0)$ of SGP within a minimum and maximum range ($S_{\min} < S(g) < S_{\max}$).

In the PV-R, PV-I and PV-RAN schemes, SPV changes its population size at a constant rate regardless of fitness variation. We believe constant variation is a serious drawback of SPV because adding individuals will require computational effort when the GP system has continuous improvements and removing individuals will

make it harder for the GP system to converge to a solution when the improvements are reduced.

DPV can counter the drawback of constant variation by dynamically controlling the size of the population for computational effort reduction [17]. The DPV technique consists of five components. They are a stagnation phase which is a period during which the improvement of the fitness is largely reduced in GP system, a characteristic measure that is a measurement of the stagnation phase, a pivot function which decides whether the new individuals need to be added, a population variation scheme which controls the number of individuals added or removed and a population increment/decrement scheme which is used to generate new individuals and remove bad individuals [18].

Here, we will introduce three previously proposed DPVs [17, 18]. Let Δ be the general delta function between the characteristic measure ϕ of generation g and generation $g-1$. Δ is defined as an assessment on the stagnation phase in GP system, which can be expressed as $\Delta_g = \phi_g - \phi_{g-1}$.

The first DPV is referred to as DIV (division pivot function) [17] and the pivot function is defined by Equation (4):

$$Pivot_{DIV}^{<\Delta_{g-1},\Delta_g>} = \begin{cases} \frac{\Delta_{g-1}}{\Delta_g}, & \text{if } \Delta_g \neq 0 \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

The second one is SUB (subtraction pivot function) [17] and is defined by Equation (5):

$$Pivot_{SUB}^{<\Delta_{g-1},\Delta_g>} = \Delta_{g-1} - \Delta_g \tag{5}$$

Let $\Delta S(g)$ be the population variation of DPV calculated by population variation scheme in generation g. If $\Delta_g \geq pivot$; then $\Delta S(g) < 0$ and the $|\Delta S(g)|$ bad performing individuals are eliminated. If $\Delta_g < pivot$; then $\Delta S(g) > 0$ and the $\Delta S(g)$ new individuals generated by population increment/decrement scheme are inserted into population.

It is noted that there exists an error in the DIV pivot function. For example, the case $\Delta_g = -0.2$ and $\Delta_{g-1} = 3$ implies the fitness improvement has reduced rapidly in the problem and new individuals should be added to the population to counter the reduction of improvement. However, according to the DIV pivot function: $\Delta_g = -0.2 > \frac{3}{-0.2} = \frac{\Delta_{g-1}}{\Delta_g}$ the GP system will be considered to be progressing in this case and will continue removing individuals. For the SUB pivot function, a rapidly improving fitness $\Delta_g \geq \Delta_{g-1} \cdot \cdot \geq \Delta_1$ will lead to a large negative pivot value of SUB. For example, with $\Delta_g = 100$ and $\Delta_{g-1} = 4$, the pivot value is -96.

The third DPV technique GRAD "gradient pivot function" [18] is defined as follows:

$$Pivot_{GRAD}^{\langle\Delta_{g-1},\Delta_g\rangle} = \Delta_{g-1}, \quad \frac{d\Delta_g}{dg} \geq \frac{d\Delta_{g-1}}{dg}. \tag{6}$$

The DIV, SUB and GRAD can use the best performing individual (BP), the mean of high-performing individuals (HP mean) or the fitness upper quartile (UPQ) as their characteristic measures of stagnation in a GP system. It was illustrated that GRAD pivot function had better performance than DIV and SUB pivot functions in a 4-bit even parity generator and a Boolean 6-symmetry problem using high-performing individuals as characteristic measure.

3 EXP POPULATION VARIATION

Methods	Type	Characteristic measure	Generic population	Pivot function	Variation scheme
PV-R	Static	–	Constant	–	Standard
PV-I	Static	–	Constant	–	Standard
PV-RAN	Static	–	Constant	–	Standard
DIV	Dynamic	BP, HP mean, UPQ	Fixed	Division	Standard
SUB	Dynamic	BP, HP mean, UPQ	Fixed	Subtraction	Standard
GRAD	Dynamic	BP, HP mean, UPQ	Dynamic	Gradient	Standard
EXP	Dynamic	HP mean & BP	Proportionate	Exponent	*IM* scheme

Table 1. Summary of various population variation methods

In this section, we investigate four innovations to the DPV technique, namely a new definition of the stagnation phase along with the characteristic measure that relies on a weighted sum of the BP and HP mean measures, an exponential pivot function (EXP), an appropriate formula for the population variation and an instruction matrix (*IM*) scheme based on an estimation-distribution model [21, 22]. We conduct a study on the effects of population variation using the new technique and discuss and compare the performance of the new approach with other methods, both static and dynamic. The details of the new approach and the comparison with other PV methods are given in Table 1.

3.1 Stagnation Phase and Characteristic Measure

The stagnation phase of a GP system denotes a period during which the improvement of the fitness is greatly reduced. When a GP system is considered to be in the stagnation phase, it needs to add new individuals to the population to counter this reduction.

DIV and SUB measured the stagnation phase by BP and GRAD by HP mean; but neither BP nor HP mean can reflect the actual stagnation in the GP. When BP has no improvements across several generations which may be considered as

stagnation, the fitness of whole population may still make large progress. When the HP mean has no improvements, the best performing individual may still make significant progress to promote the evolution. Therefore, DPV methods cannot perform well using only BP or HP mean as characteristic measure. We believe the stagnation phase in a GP system should depend on the combination of BP and HP mean. Our stagnation phase definition is that only when the combination of the mean of BP and HP mean during the run is not making progress, the GP system can be considered to be in the stagnation phase.

For example, let f_i^{best} be fitness of BP and $\overline{f_i}$ be fitness of HP mean in generation i. if $f_i^{best} > f_j^{best}$ and $\overline{f_i} < \overline{f_j}$ $(i > j)$, then DIV and SUB will consider the GP system to be progressing, while GRAD will consider the GP system to be in the stagnation phase. According to our definition, if $\overline{f_i} + f_i^{best} > \overline{f_j} + f_j^{best}$, then the system is making progress, else the system is in the stagnation phase. Figure 1 shows an example of our definition.

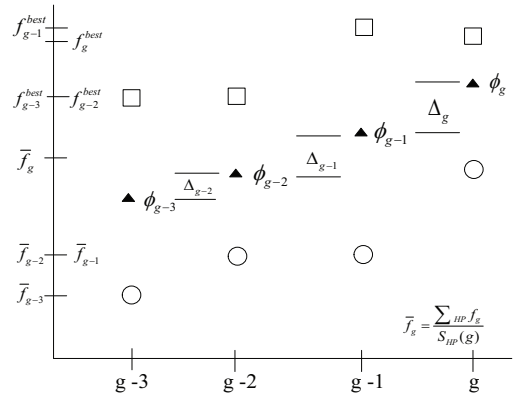


Fig. 1. Average fitness of high performing individuals and best fitness versus generation-maximum problem

To recognize the stagnation phase in GP, a characteristic measure is defined. Three types of characteristic measures were proposed in [18]. They were based on BP, HP mean and UPQ, which are defined by Equations (7), (8) and (9)

$$\phi_g = f_g^{best} \tag{7}$$

$$\phi_g = \frac{\sum_{HP} f_g}{S_{HP}(g)} \tag{8}$$

$$\phi_g = upq_g \tag{9}$$

where f_g indicates the fitness of an individual in the population at generation g and $\sum_{HP} f_g$ is the sum of the fitnesses of high-performing individuals. $S_{HP}(g)$ represents the number of individuals with performance in the top 50 %. According to our stagnation phase definition, we propose a new characteristic measure, which can be

defined as (10):

$$\phi_g = \alpha \cdot \frac{\sum_{HP} f_g}{S_{HP}(g)} + \beta \cdot f_g^{best} \tag{10}$$

where α and β represent the weights for the average fitness of HP and BP individuals, respectively. f_g^{best} represents the fitness of the best individual in generation g. $\sum_{HP} f_g$ denotes the top 50 % high-performing individuals sorted by fitness descending in generation g. $S_{HP}(g)$ indicates the number of $\sum_{HP} f_g$.

3.2 EXP Pivot Function

The pivot function decides whether new individuals should be inserted into the population or some poor individuals in population must be removed. Here, we present an exponential pivot function (EXP) which is defined as follows:

$$Pivot_{EXP}^{\langle g,g+T-1\rangle} = \Delta_{g-1} \cdot \exp^{-|\Delta_g|}. \tag{11}$$

The relationship expressed by Equation (12) is presented for EXP pivot function as follows:

$$EXP : \Delta_g \geq \Delta_{g-1} \cdot \exp^{-|\Delta_g|} \Rightarrow \Delta_g \cdot \exp^{-|\Delta_g|} \geq \Delta_{g-1}. \tag{12}$$

It is noted that $\Delta_g > 0 > \Delta_{g-1}$ implies the population has ideal fitness improvements and thus $\Delta_g > pivot_{EXP}$ always holds. The GP system will remove a small number of individuals to reduce computational effort. $\Delta_g < 0 < \Delta_{g-1}$ implies the population is in the stagnation phase and thus $\Delta_g < pivot_{EXP}$ holds. The GP system will add new individuals to promote performance and accelerate convergence. This illustrates that the EXP pivot function can add new individuals or remove bad individuals correctly according to the actual situation of evolution.

It is expected that the DIV can perform better than EXP, SUB and GRAD for the best performing individuals in terms of AES, because when the fitness improvements for the best performing individuals normally diminish greatly $\Delta_g \ll \Delta_{g-1} \ll \cdots \ll \Delta_1$, the pivot of DIV is most probably larger than other pivot functions, so it adds new individuals to promote the fitness of the population and counter the stagnation of evolution while the SUB, GRAD and EXP may still consider the GP system to be progressing and continuously remove individuals. We are interested in whether the EXP can outperform other DPVs using our stagnation definition and characteristic measure.

3.3 Population Variation Scheme

An appropriate population variation scheme can be a benefit for the evolution by low computational effort. A population variation scheme with constant parameters was defined by (13) [17]:

$$\Delta S(g) = \begin{cases} 0.2\,\% \cdot S(g) \cdot \Delta_g, & \Delta_g < Pivot \\ -1\,\% \cdot S(g) \cdot \Delta_g, & \Delta_g \geq Pivot. \end{cases} \tag{13}$$

It is noted that (13) is not correct because $\Delta_g > 0$ does not always hold. For example, if $\Delta_g = -2$ and $Pivot = 1$, then $\Delta_g < pivot$ and the evolution system should increase individuals to deal with this case; but Δ_g is a negative value calculated by Equation (13) and in this case the system will remove individuals incorrectly.

Another dynamic population variation scheme without any empiric constant was defined by (14) [18]:

$$\Delta S(g) = (-1)^n \cdot S(g) \cdot \frac{|\Delta_g - \Delta_{g-1}|}{f_{optimal}} \tag{14}$$

where $n = 1$, if $\Delta_g \geq Pivot$; $n = 2$, if $\Delta_g < Pivot$. $f_{optimal}$ denotes the fitness of the global solution of the problem. According to (14), if $\Delta_g \simeq \Delta_{g-1} > 0$, $\Delta S(g) \simeq 0$. The population size change is so small that the effect of computation effort reduction is not ideal.

We propose a novel proportionate population variation scheme for EXP defined as follows:

$$\Delta S(g) = (-1)^n \cdot S(g) \cdot \frac{2|\Delta_g|}{f_g^{best} + f_{g-1}^{best}}, \quad S(g) > Z \tag{15}$$

where $n = 1$, if $\Delta_g > Pivot$; $n = 2$, if $\Delta_g \leq Pivot$. Z is an empiric constant. If $\Delta S(g) \leq Z$, $\Delta S(g)$ is set to zero. This constraint is used to avoid the population becoming too small. According to (15), size variation is smooth and size undergoes a large change only when both f_g^{best} and f_{g-1}^{best} have large variation.

3.4 *IM* Population Increment/Reduction Scheme

A standard scheme deployed for the population increment/reduction is described as follows [18]: Given the set of N individuals $T = \{T_1, T_2, \cdots T_N\}$. If the fitness improvements are small, the $m(m < N)$ best individuals are selected and mutated in the next generation. If the performance of GP is progressing, the k worst individuals will be washed out. This scheme destroys the diversity of the population since it only keeps individuals which perform well. Maintaining or increasing diversity is generally and conventionally considered as beneficial in GP [19, 20]. Therefore, we investigate a new scheme, the Instruction Matrix or *IM*, to replace the standard scheme.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	+	×	×	÷	−	÷	+	**d**	**f**	**a**	**e**	**f**	**b**	**c**	**d**
2	−	×	×	×	**b**	**d**	**d**	**a**	**e**	*b*	*f*	*f*	*c*	*c*	*d*
3	×	÷	−	+	×	÷	**c**	**b**	**e**	**a**	**d**	**e**	**b**	**a**	*e*
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
20	÷	×	×	−	+	**e**	**f**	**b**	**a**	**b**	**c**	**d**	**e**	**f**	**d**

Table 2. An instance of instruction matrix (*IM*)

IM structure: The i^{th} row of *IM* represents the i^{th} ($i \leq m$) individual in the set T and the j^{th} column of *IM* represents the j^{th} node in the individual. m denotes the number of the high performing individuals selected. Let W and H be the width and height of *IM* and h be the depth of complete binary tree of W nodes, i.e. $W = 2^h - 1$, $H = m$. An example is given in Table 2. There are 20 individuals are selected and a complete tree contains 15 nodes. $W = 15$, $H = m = 20$. The bold symbols in *IM* are the actual leaves of the tree. The non-bold symbols behind bold symbols are randomly produced to fill the blanks in *IM*.

Create *IM*:

a) Select $m(m < S(g))$ high-performing individuals (HP) at generation g.
b) Select individual i in HP and fill its symbols into the i^{th} row of *IM*.
c) If all individuals in HP have been selected, then it's the end; else go to b).

Extract an individual from *IM*:

a) Create a new individual.
b) Select a symbol from the i^{th} column of *IM* using roulette wheel selection and set it to the i^{th} node.
c) If all the nodes of individuals have been created, then it's the end; else go to b).

If $\Delta S(g) > 0$, $\Delta S(g)$ new individuals generated from *IM* will be added to population. If $\Delta S(g) < 0$, $|\Delta S(g)|$, bad individuals will be removed from population.

4 APPLICATIONS

We chose four representative problem types for the experimental investigations, namely a symbolic regression problem, digital logic problem, sequence induction and an artificial ant problem. There are two instances used in digital logic problem, the 4-bit odd parity checker problem and the 6-multiplexer problem. Santa Fe landscape is used in the artificial ant problem. All these problems are benchmarks in GP research [23, 24, 25, 26]. The genetic programming paradigm was implemented as a Lisp-tree GP using the VC++ programming environment; the experiments were executed on PC with an Intel dual-core processor (only one is used). The crossover probability P_c is set to 0.8 and mutation probability P_m is set to 0.15. Let $G_{\max}$ be the maximum generation and $S(0)$ be the initial size of population. $Z = 0.25 \cdot S(0)$, $\alpha = 0.4$ and $\beta = 0.6$. The experiments' parameters are described in Table 3. All the problem tests use 20 independent runs except the artificial ant problem; we only ran it 10 times because it takes more than three hours for an independent run.

4.1 Symbolic Regression Problem

For the symbolic regression problem type a polynomial equation of degree-4 is defined by $f_1(x) = 1.5x^4 + 1.5x^3 + x, x \in [0, 2]$ and a polynomial equation of degree-3

	Symbolic regression	Sequence induction	Digital logic circuits	Artificial ant problem
$G_{\max}$	400	200	600	600
$S(0)$	200 (SGP, DPV), 100 (PV-I), 700 (PV-R), 400 (PV-RAN)	200 (SGP, DPV), 100 (PV-I), 500 (PV-R), 300 (PV-RAN)	200 (SGP, DPV), 100 (PV-I), 900 (PV-R), 500 (PV-RAN)	200 (SGP, DPV), 100 (PV-I), 900 (PV-R), 500 (PV-RAN)
The termination criteria of a GP run	$f_g^{best} - f_{optimal} < 0.05$ or $g > G_{\max}$	$\lvert f_g^{best} - f_{optimal} \rvert < 0.8$ or $g > G_{\max}$	$f_g^{best} = 16$ (odd parity checker), $f_g^{best} = 64$ (6-multiplexer) or $g > G_{\max}$	$f_g^{best} = 89$ or $g > G_{\max}$

Table 3. Experiments parameters

is defined by $f_2(x) = -x^3 + x^2 + x, x \in [-2, 3]$. The terminal set is defined as $T = \{1, 2, 3, R, x\}$ and the function set is given by $F = \{+, \times, -\}$, where R is a random float constant in the interval $[-5, 5]$.

4.2 Digital Logic Problem

The first digital logic problem is the Odd Parity 4-Checker problem with $k = 4$ Boolean arguments and $2^4 = 16$ fitness cases described by the function $F(a, b, c, d)$, which returns true '1' if an odd number of its Boolean arguments is evaluated as true, otherwise it returns false '0'. Terminal Set $T = \{a, b, c, d\}$, function set $F = \{NAND, NOR, OR, AND, NOT\}$. The second digital logic problem applied is the 6-multiplexer problem with $k = 6$ Boolean arguments and $2^6 = 64$ fitness cases described by the function $F(a, b, c, d, e, f)$, Terminal Set $T = \{a, b, c, d, e, f\}$, function set $F = \{NAND, NOR, OR, AND, NOT\}$.

4.3 Sequence Induction Problem

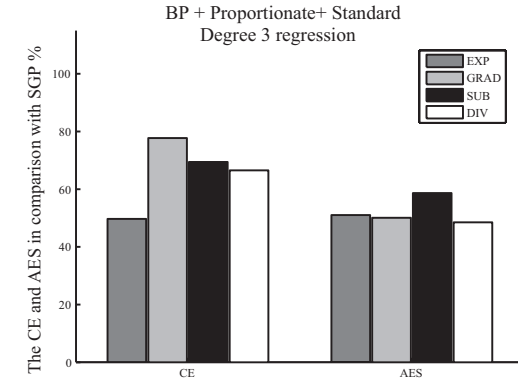
A sequence defined as $S = \{6, 11, 18, 27, 38, 51, 76, 93, 112, 133 \cdot \cdot\cdot\}$ is used for this problem in [15]. The terminal set is defined as $T = \{1, 2, 3, x\}$ and the function set is given by $F = \{+, -, \times\}$.

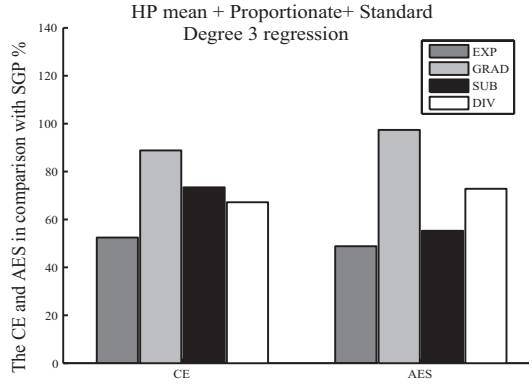
4.4 Artificial Ant Problem

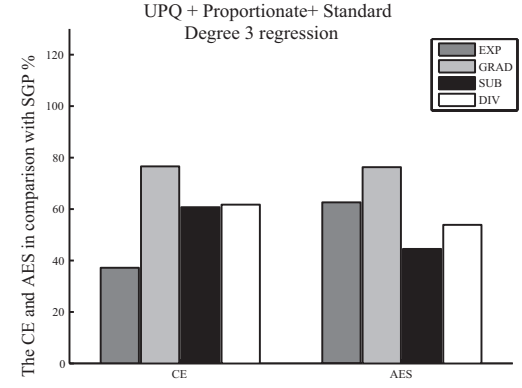
The Santa Fe trail is often used for the artificial ant problem. The Santa Fe trail consists of 89 food elements on a two dimensional, 3 232 toroidal grid. The ant is limited to 600 time steps in this landscape. The terminal set is defined as $T = \{Move, Left, Right\}$ and the function set is given by $F = \{IF\text{-}FoodAhead, Prog2, Prog3\}$.

5 RESULTS AND DISCUSSION

5.1 Comparison Among Different Measures







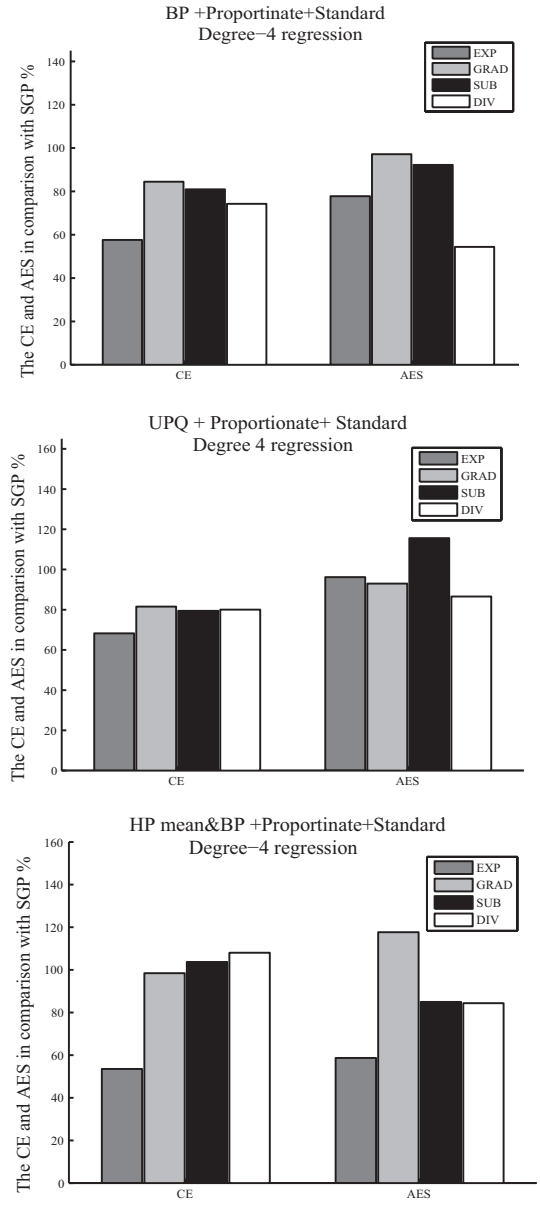


Fig. 2. DPV-BP, HP mean, UPQ, and HP mean & BP as characteristic measure with proportionate and standard scheme (regression problem)

Figure 2 shows the comparative *CE* and *AES* of degree-4 and degree-3 regression problems. The pivot functions DIV, SUB, GRAD and EXP use different measures (BP, HP mean, UPQ and hybrid of HP mean and BP) along with proposed increment/reduce scheme and the standard population variation schemes are compared with SGP algorithm.

It is noted that computational effort and *AES* did not always follow the similar trend in some instances. The reason is that a rapid convergence (small generations) yields a small *AES* but the small success rate of runs can lead to the high computational effort. A low computational effort must depend on a high success rate and rapid convergence to solutions according to (1). Similar results were obtained for four pivot functions in degree-3 and degree-4 regression problems for using BP as characteristic measure. DIV inserted new high fitness individuals to boost the performance of GP and attempted to accelerate the convergence, especially when the improvement of current generation is very small, whereas the EXP, GRAD and SUB may continually have removed individuals in these instances and therefore made it harder for the GP system to converge towards the solution. Therefore, it can be seen in Figure 2 the DIV outperformed other pivot functions in terms of *AES* for the BP measure in two regression problems.

In general, GRAD performed worst among all the pivot functions when the measure was UPQ or HP mean. DIV presented the worst performance when the measure was a hybrid of HP mean and BP. EXP was superior to the other three pivot functions in terms of *CE* when the measure was UPQ, HP mean, or our measure. For the high-performing individuals, although there are large improvements during the initial stage of the run, eventually changes become smooth and small towards the end of a generation. For the proposed measure ϕ, the average fitness of high-performing individuals shows more or less improvement during the run of GP and the best individual can be reserved for several generations if an elite strategy is applied, thus there were continuous improvement in ϕ, for $\Delta_g < \Delta_{g-1}$ and $\Delta_g < 1$, EXP tended to remove a small number of individuals as a result. However, DIV and GRAD considered the GP system to be stagnated according to their pivots and increased the population size to promote the mean fitness of the population. These new individuals created became an extra computational burden and led to poor results. This is the main reason for the poor performance of DIV and GRAD in these instances.

5.2 Comparison Among Different Weights in Measure

The main feature of the proposed measure ϕ distinguished from HP mean and UPQ is that the fitness improvements for this measure are as smooth as those for HP mean but BP is included so it can also exert influence on determining the stagnation. The parameter settings ($\alpha = 0.4$, $\beta = 0.6$) in the proposed measure were used in previous experiments. However, this setting cannot guarantee the best performance in these instances. DPV using other measure (α, β) settings were observed for odder even 4-checker problem in Figure 3. In general, the $\alpha = 0.75$, $\beta = 0.25$ performed best

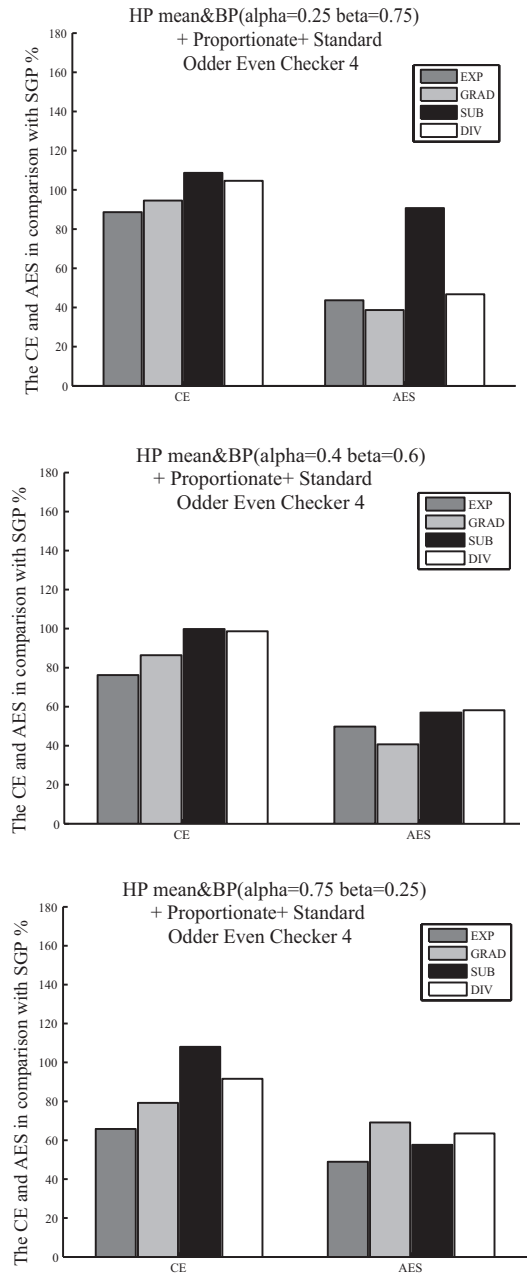


Fig. 3. DPV-HP mean & BP as characteristic measure using different setting (α, β) (odd parity 4-checker problem)

among all the settings, the *CE* and *AES* results approximated those using HP mean because the proposed measure ϕ is closer to the HP mean. When $\alpha = 0.4$, $\beta = 0.6$ and $\alpha = 0.25$, $\beta = 0.75$, the performance reduction in terms of *CE* was observed and in these two cases the measure ϕ performed worse than HP mean. In this problem the fitness for measure ϕ was not so steep as HP mean but very smooth and the improvements for measure ϕ were reduced so much that DPV methods incorrectly forced the system to insert extra individuals when the weight of HP mean was too small in measure ϕ. Although $(\alpha = 0.4, \beta = 0.6)$ is not the best setting for our measure ϕ, it is used for other tests because it can give the full play of the role of the BP in our measure ϕ.

5.3 Comparison Among Different Schemes

The *CE* and *AES* results of DPV methods using a fixed increment/decrement scheme (Equation (13)), dynamic increment/decrement scheme (Equation (14)) and our proportionate increment/decrement scheme (Equation (15)) for sequence induction problem along with proposed measure ϕ are given in Figure 4. The performance of dynamic variation scheme observed was superior to that of the fixed population variation scheme, but our proportionate scheme was superior to the other two population variation schemes. When the GP system had low improvement $\Delta_g > \Delta_{g-1} > \Delta_{g-2}$, DPV methods modified the size of the population according to a dynamic variation scheme (Equation (14)) by a very small amount. In this case new individuals injected were too few to promote the performance of GP and the poor individuals left over in population impeded the system from achieving convergence to a solution; but for our proportionate variation scheme, DPV changed the population size according to Δ_g. New individuals were inserted to accelerate the convergence whereas poor performing individuals were eliminated to enhance average fitness. This is the main reason for the superiority of our proportionate scheme.

As shown in Figure 5, the *IM* population variation scheme proposed performed very similarly to the standard scheme in terms of CE used by the EXP and GRAD. The computational effort and *AES* of DIV and SUB were obviously lowered by using *IM* scheme. This is due to the fact that when some poor individuals are eliminated to reduce the computational effort, the diversity of the population may be destroyed at the same time so that the convergence time (generations) of GP may increase and thus counteract the reduction of computational effort. The *IM* scheme collects excellent building blocks of high fitness individuals to produce new individuals to maintain diversity of the population in the next generation and promote the convergence of the GP system. Therefore, the *AES* results of DIV, SUB and EXP that represent the fast convergence were significantly improved when *IM* was used as the population variation scheme.

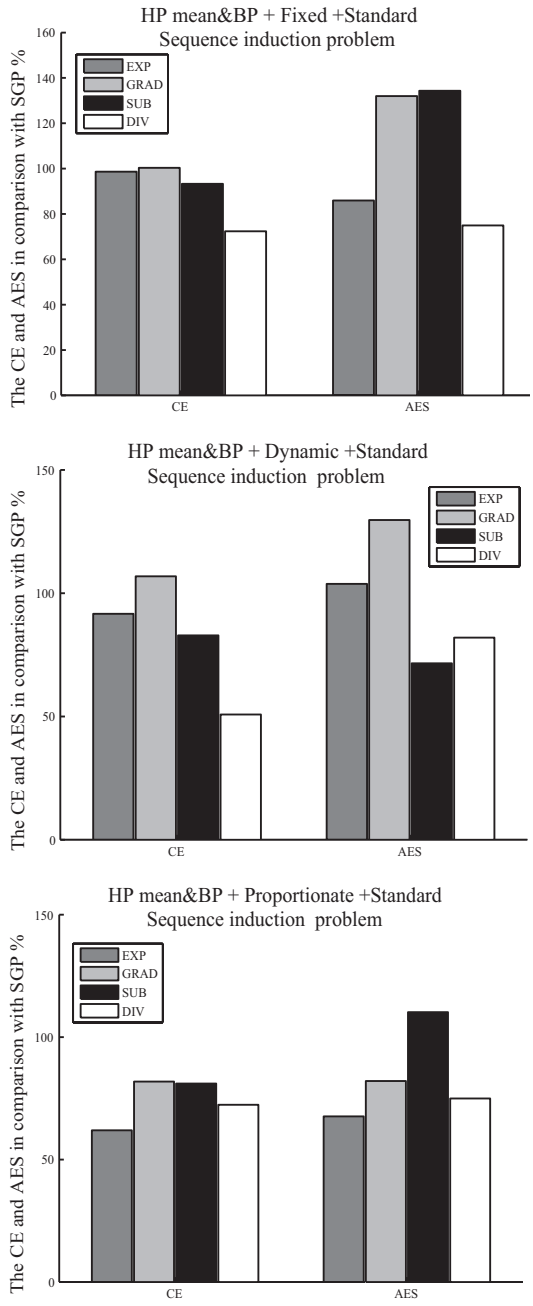


Fig. 4. DPV- HP mean & BP as characteristic measure with different population variation schemes (sequence induction)

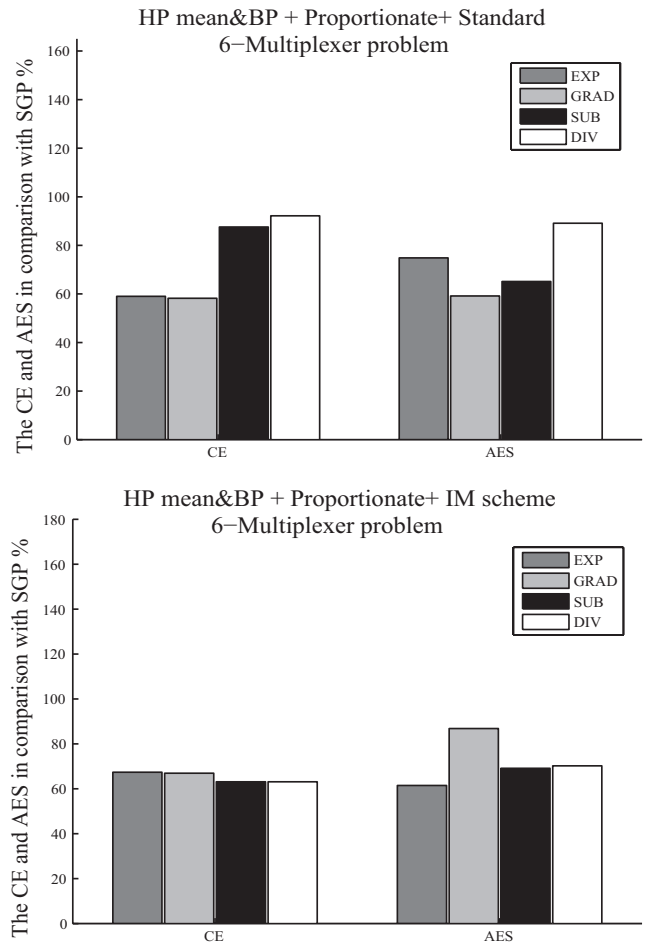


Fig. 5. DPV-HP mean & BP as characteristic measure with *IM* and standard schemes (6-multiplexer problem)

5.4 Comparison Among All PV Methods

Figure 6 shows performance comparison among all PV methods for degree-4 regression, sequence induction, 6-multiplexer and artificial ant problem. It is seen that for degree-4 regression and sequence induction problem, EXP performed better than other methods and GRAD showed the worst capability among DPV methods. For the 6-multiplexer problem, DIV and SUB performed a little better than other DPV methods and the performance of the EXP approximately equalled to GRAD in terms of *CE*. For the artificial ant problem, EXP was the best pivot function.

EXP reduced computational efforts by continuously removing individuals when GP system had a large progress at the beginning of run and created new individuals with excellent block to boost up the diversity for faster convergence to solution when the improvements of fitness dropped greatly. Therefore, we can see that for all these cases EXP gave a more excellent behavior in terms of *AES*. For all the test problems, the static PV methods had poorer performance than DPV methods. The redundant individuals for the static PV methods became the extra computational burden on GP, whereas the DPV methods eliminated redundancy individuals during the initial stage of evolution when improvements were large.

The performance of PV-R was superior to other two static PV methods. Because of its large initial population size, these individuals enhanced the diversity of population and lead to a high success rate. The best *CE* result was a bonus for this high success rate (this was proved in Section 5.5). PV-I performed worse than other static PV methods in terms of *CE* in these instances, because the size of the population increased for PV-I step by step with a constant value. The size increment was reverse to the fitness improvements during the run and the computational efforts increased as the individuals grew; but PV-I had a capacity for achieving a fast convergence to solution with this feature and this case was demonstrated by *AES* in Figure 6. For PV-RAN, the population size changed randomly, that had no relationship with the improvements of individuals' fitness and hence the performance of the PV-RAN may be better or poorer than with other PV methods.

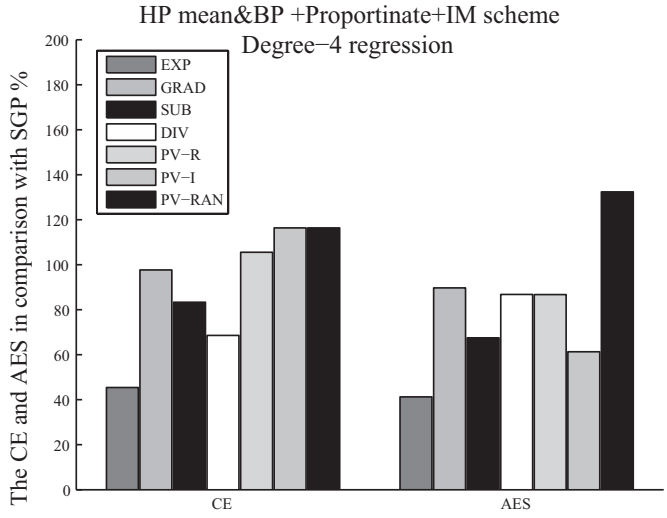
5.5 Success Rate for the PV Methods

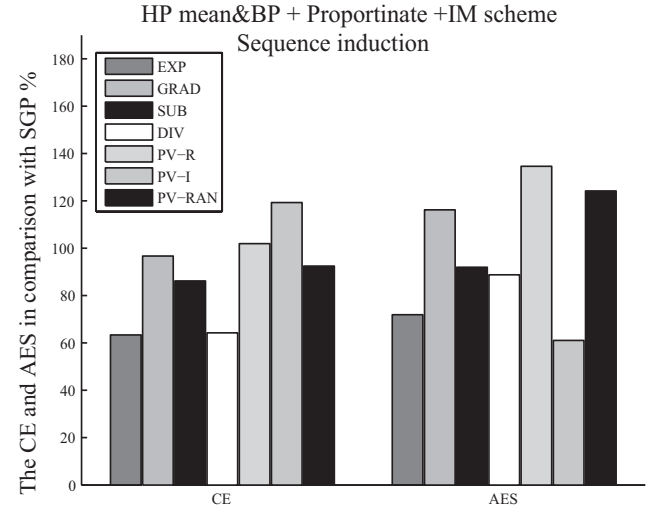
In terms of success rate, since increasing the running time (generations) of a GP system can increase its success rate, besides the $G_{\max}$ used in Table 3, we will try a large value of $G_{\max}$ for a fair comparison with the initial size of population as given in Table 4. In these tests, DPV methods use the proposed measure along with the proposed proportionate variation and *IM* scheme. The details of *SR* for PV methods in six problems using $G_{\max}$ in Table 3 are given in Table 5. Table 6 shows the details of *SR* for various PV methods in six problems using $G_{\max}$ in Table 4.

	Symbolic regression	Sequence induction	Digital logic circuits	Artificial ant problem
$G_{\max}$	600	400	800	800
$S(0)$	200 (SGP, DPV), 100 (PV-I), 700 (PV-R), 400 (PV-RAN)	200 (SGP, DPV), 100 (PV-I), 500 (PV-R), 300 (PV-RAN)	200 (SGP, DPV), 100 (PV-I), 900 (PV-R), 500 (PV-RAN)	200 (SGP, DPV), 100 (PV-I), 900 (PV-R), 500 (PV-RAN)

Table 4. Experiments parameters for a large $G_{\max}$

It was noted that *SR* of all the PV methods outperformed SGP in most cases except odd parity 4-checker. The reason was that PV methods could keep diversity and achieved high average fitness by removing poor individuals and inserting new





better individuals into the population. In static PV methods, PV-I had the worst success rate in all these problems. This is due to the fact that the initial size was so small that the diversity could not be maintained to promote the performance of GP. The success rates of PV methods for odd parity 4-checker were all much lower than for 6-multiplexer problem. The main reason was that the fitness value for odd parity 4-checker is much smaller than 6-multiplexer. In odd parity 4-checker problem, the subtraction $(\Delta_g - \Delta_{g-1})$ of improvement between generation $g-1$ and g was very small and new individuals would not be created and inserted to promote

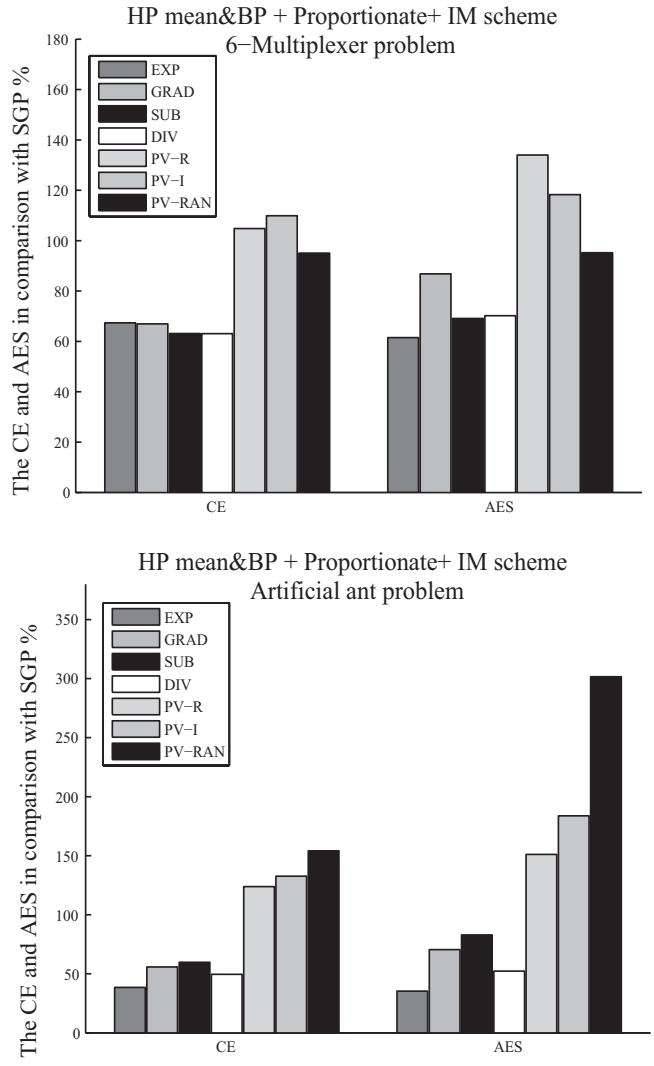


Fig. 6. Performance comparison among all PV methods and SGP (degree-4 regression, sequence induction, 6-multiplexer and artificial ant problem)

the progress. Hence it was harder for GP system to achieve the convergence in this case.

On the whole, EXP had a high success rate in these instances. This received benefit from its special measure of stagnation. It is believed that the success rate of the GP algorithm will increase as the generations grow. This is demonstrated by Table 6. When $G_{\max}$ was set to a larger value, the success rate correspondingly became higher in most cases; but it is noted that the *SR* of EXP has a drop in the

	SGP	DIV	SUB	GRAD	EXP	PV-R	PV-I	PV-RAN
Degree-4 regression	25 %	30 %	25 %	40 %	40 %	30 %	15 %	25 %
Degree-3 regression	25 %	40 %	30 %	45 %	45 %	35 %	20 %	30 %
Odd parity 4-checker	40 %	20 %	25 %	15 %	25 %	10 %	30 %	40 %
6-multiplexer	80 %	90 %	90 %	95 %	95 %	95 %	60 %	75 %
Sequence induction	20 %	20 %	35 %	50 %	30 %	65 %	30 %	70 %
Artificial ant	10 %	70 %	90 %	40 %	70 %	50 %	50 %	10 %

Table 5. *SR* of various PV methods and SGP for six problems using $G_{\max}$ in Table 3

	SGP	DIV	SUB	GRAD	EXP	PV-R	PV-I	PV-RAN
Degree-4 regression	30 %	50 %	35 %	35 %	40 %	45 %	50 %	55 %
Degree-3 regression	25 %	50 %	45 %	70 %	50 %	70 %	20 %	45 %
Odd parity 4-checker	45 %	30 %	30 %	35 %	30 %	35 %	25 %	60 %
6-multiplexer	95 %	85 %	80 %	95 %	75 %	90 %	80 %	85 %
Sequence induction	75 %	70 %	55 %	60 %	65 %	60 %	60 %	55 %
Artificial ant	80 %	80 %	60 %	20 %	50 %	50 %	50 %	10 %

Table 6. *SR* of various PV methods and SGP for six problems using $G_{\max}$ in Table 4

6-multiplexer and artificial ant problems while the *SR* of SGP has improvements in each problem using a long evolution time. When the GP system is making progress, EXP will remove some individuals to ease computational effort and a long evolution time may result in losing diversity as the number of individuals continuous to decrease, which is bad for accelerating convergence; but for SGP, better performing individuals may be obtained as the evolution time grows.

6 CONCLUSIONS

The aim of population variation is to save computational effort by increasing or decreasing the population size during a run of GP. A new DPV using a feedback mechanism in genetic programming called EXP is proposed in this paper to improve the DPV. A new stagnation phase definition based on best and high-performing individuals, a population variation scheme, the *IM* scheme and exponential pivot function are suggested to improve the performance of DPV. The performances of various population variation (PV) approaches have been analyzed and discussed in our study. We have experimented with various PV methods on different types

of problems. Comparison among the different characteristic measures has been conducted for regression problems and the proposed measure performed best. It has been demonstrated the new proposed population variation scheme is superior to fixed and proportionate population variation schemes for sequence induction. Experimental evidence shows that the individuals' increment scheme using an *IM* proposed can be a benefit for diversity of the population. It has been proved the proposed EXP approach can achieve better results than other PV methods and can outperform the standard genetic programming (SGP) in most problems. It is also found that the PV methods have a larger success rate than SGP in most problems. Furthermore, applying the proposed algorithm to solve other modeling and optimization problems is also possible in further research.

REFERENCES

[1] Koza, J. R.: Genetic Programming: On the Programming of Computers by the Means of Natural Selection. MIT Press, Cambridge, MA. 1992.

[2] Koza, J. R.: Genetic Programming II: Automatic Discovery of Reusable Subprograms. MIT Press, Cambridge. MA. 1994.

[3] Zhang, B.—Ohm, P.—Muhlenbein, D.: Evolutionary Neural Trees for Modelling and Predicting Complex Systems. Engineering Applications of Artificial Intelligence, Vol. 10, 1997, pp. 473–483.

[4] Zhou, A. M.—Cao, H. Q.: the Automatic Modeling of Complex Functions Based on Genetic Programming. Journal of system simulation, Vol. 15, 2003, pp. 797–799.

[5] Miller, J. F.—Job, D.—Vassilev, V. K.: Principles in the Evolutionary Design of Digital Circuits – Part I. Journal of Genetic Programming and Evolvable Machines, Vol. 01, 2000, pp. 08–35.

[6] Miller, J. F.—Job, D.—Vassilev, V. K.: Principles in the Evolutionary Design of Digital Circuits – Part II. Journal of Genetic Programming and Evolvable Machines, Vol. 03, 2000, pp. 259–288.

[7] Miller, J. F.—Thomson, P.: Cartesian Genetic Programming. Proceedings of Europe Genetic Programming, Edinburgh, Scotland, UK 2000, pp. 121–132.

[8] Sykulski, J. K.: Reducing Computational Effort in Field Optimization Problems. The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, Vol. 23, No. 01, 2004, pp. 159–172.

[9] Chongstitvatana, P.: Improving Robustness of Robot Programs Generated by Genetic Programming for Dynamic Environments. Proceedings of IEEE Asia Pacific Conference on Circuits and Systems 1998, pp. 523–526.

[10] Daida, J. M.—Bersano-Begey, T. F.—Ross, S. J.: Evolving Feature-Extraction Algorithms: Adapting Genetic Programming for Image Analysis in Geoscience and Remote Sensing. Proceedings of Geoscience and Remote Sensing Symposium: Remote Sensing for a Sustainable Future 1996, pp. 2077–2079.

[11] Lee, K. J.—Zhang, B. T.: Learning Robot Behaviors by Evolving Genetic Programs. Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society (IECON2000), Nagoya, Japan 2000, pp. 2867–2872.

[12] Niehaus, J.—Banzhaf, W.: More on Computational Effort Statistics for Genetic Programming. EuroGP 2003, LNCS 2610, 2003. pp. 164–172.

[13] Walker, M.—Edwards, H.—Messom, C.: Confidence Intervals for Computational Effort Comparisons. Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Portland, Oregon, USA 2010. pp. 975–976.

[14] Shi, X. H.—Liang, Y. C.—Lee, H. P.: An Improved GA and a Novel PSO-GA-Based Hybrid Algorithm. Information Processing Letters, Vol. 93, 2005, pp. 255–261.

[15] Fernandez, F.—Tomassini, M.—Vanneschi, L.: Saving Computational Effort in Genetic Programming by Means of Plagues. Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003), Canberra, Australia 2003, pp. 2042–2049.

[16] Kouchakpour, P.—Zaknich, A.—Branul, T.: Population Variation in Genetic Programming. Information Sciences, Vol. 177, 2007, pp. 3438–3452.

[17] Tomassini, M.—Vanneschi, L.—Cuendet, J.: a new Technique for Dynamic Size Populations in Genetic Programming. Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC2004), Oregon, Portland 2004, pp. 486–493.

[18] Kouchakpour, P.—Zaknich, A.—Branul, T.: Dynamic Population Variation in Genetic Programming. Information Sciences, Vol. 179, 2009, pp. 1078–1091.

[19] Li, G.—Wang, J. F.—Lee,K. H.: Instruction-Matrix-Based Genetic Programming. IEEE Transactions on Systems, Man, and Cybernetics – Part B. Vol. 38, 2008, pp. 1036–1049.

[20] Yanai, K.—Iba, H.: Estimation of Distribution Programming: EDA-Based Approach to Program Generation. StudFuzz., Vol. 192, 2006, pp. 103–122.

[21] Burke, E. K.—Gustafson, S.—Kendall, G.: Diversity in Genetic Programming: An Analysis of Measures and Correlation with Fitness. IEEE Transactions on Evolutionary Computation. Vol. 8, 2004, pp. 47–62.

[22] Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. University of New Mexico, Albuquerque 1995.

[23] Hoai, N. X.—McKay, R. I.—Essam, D.: Solving the Symbolic Regression Problem with Tree-Adjunct Grammar Guided Genetic Programming: The Comparative Results. Proceedings of Congress on Evolutionary Computation (CEC2002), Honolulu, Hawaii, USA 2002, pp. 1326–1331.

[24] Candida, F.: Gene Expression Programming: A new Adaptive Algorithm for Solving Problems. Complex Systems, Vol. 13, 2001, pp. 87–129.

[25] Cheang, S. M.—Lee, K. H.—Leung,K. S.: Applying Genetic Parallel Programming to Synthesize Combinational Logic Circuits. IEEE Transactions on Evolutionary Computation, Vol. 11, 2007, pp. 503–520.

[26] Christensen, S.—Oppacher, F.: Solving the Artificial ant on the Santa fe Trail Problem in 20 696 Fitness Evaluations. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, England 2007, pp. 1574–1579.

Yanyun Tao works at Department of School of Urban Rail and Transportation in Soochow University from 2012. Between 2010–2012, he worked as a postdoctoral research fellow of computer science in Shanghai Jiaotong University. In 2010 he received his Ph. D. from East China University of Science and Technology, and his B. Sc. from Hohai University. His research fields are evolutionary computation (GA,GP, CGP, GEP), evolution of digital circuit and evolutionary modeling of complex systems. He has 11 publications in evolutionary computation.

Minglu Li is a Professor of computer science of Shanghai Jiao Tong University. He is the Vice President of School of Electronic, Information and Electrical Engineering in Shanghai Jiao Tong University. Between 1993 and 1996 he studied in Shanghai Jiao Tong University and received the Ph. D. in computer science. Between 1981 and 1985 he studied at PLA Information Engineering University and received his B. Sc. in computer science. In 2001 he was a Visiting Scholar in the Department of Information System of City University of Hong Kong. His research fields include grid computing, services computing, wireless sensor networks, and cloud computing. He has published more than 120 research papers in these fields.

Jian Cao is a Professor of computer science in Shanghai Jiaotong University. In 2000 he received his Ph. D. from Nanjing University of Science and Technology. 2000–2002, he worked as a Postdoctoral Research fellow in Shanghai Jiao Tong University. Since 2002, he has been with Shanghai Jiao Tong University. In 2004, he worked as a Visiting Scholar in Stanford University. His research fields include service oriented computing, network computing, and multiagent system and decision support systems He has published more than 110 research papers in these research fields.