The Convergence Mechanism and Strategies for Non-Elitist Genetic Programming

He Ni
Simulation Research Certain of Naval Ship Engineering
Naval University of Engineering
Wuhan, P. R. Chin a
E-mail: elegance2006@sina.com

Bo Yu
School of Humanities and Arts
Nanchang Institute of Technology
Nanchang, P. R. Chin a
E-mail: july8511@sina.com

Fanming Zeng
Department of Marine Engineering
Naval University of Engineering
Wuhan, P. R. China
E-mail: zengfanming@public.wh.hb.cn

Fengrui Sun
Department of Gas and Steam Power Engineering
Naval University of Engineering
Wuhan, P. R. China
E-mail: fengrui_sun@public.wh.hb.cn

***Abstract*—Genetic programming is an evolutionary algorithm that proposed to solve the automatic computer program design problem by J.R.Koza in the 1990s. It has good universality and intelligence, and has been widely applied in the field of computer engineering. But genetic programming is essentially a stochastic optimization algorithm, lack theoretic basis on the convergence of algorithm, which limit the scope of its application in some extent. The convergence mechanism of non-elitist genetic programming was studied in this paper. A recursive estimation of the probability of population contains satisfactory solution with the evolution algebra was established by the analysis of operators' characteristic parameters, then a sufficient condition of population converge in probability was derived from this estimation, and thereby some operational convergence strategies for many common evolution modes were provided.**

Keywords- non-elitist genetic programming; convergence mechanism; convergence strategy; algorithmic pause time

I. Introduction

1990s, John R Koza[1,2] in Stanford university proposed an evolutionary algorithm, this algorithm adopts hierarchical computer program to express the problem, was known as genetic programming algorithm (GP). This algorithm can actually be regarded as a kind of genetic algorithm for computer program optimization and automatic generation. The basic idea of GP algorithm is quiet similar to genetic algorithm, but its operation objects is not traditional GA code strings, but the computer programs that solving problem.

As a program optimizing and automatic generate method, GP algorithm has been successfully applied in artificial intelligence[3], expert diagnostic system[4], and the natural evolution of computer languages[5], Etc. Through a series of engineering practice, no scholar doubts the feasibility of GP algorithm as an automatic programming method, but its mathematical foundation is weakness, especially in the convergence of algorithm. It is mainly because GP algorithm is a stochastic optimization algorithm, its convergence mechanism has essentially uncertainty. This issue is worthy of further exploration, because an algorithm that can't or can not easily converge doesn't make sence to program design.

The so-called algorithm convergence is actually a merit-based capacity, i.e. the ability to find the individual that can better meet certain requirements from populations. The conditions required for convergence is called convergence conditions and the strategies that adopted to achieve convergence is called convergence strategy. In the existing related theories, most studies[6-8] tend to believe that genetic programming can not find the optimal solution within a determined time, but so far, there is no strict proof of this assertion. Vosef used a nonlinear functional operator to describe the evolution process of population under certain assumptions[9,10], and forecast the behaviors of population through the analysis of this operator, get an conclusion that population will eventually converge to a steady state when adopted a linear fitness function, but he did not specify whether this steady state corresponds to a satisfactory solution of problem. Rudolph[11] adopted the Markov property of population evolution to calculate the probability of population contains a satisfactory solution at any moment, concluded that the canonical genetic programming does not converge, while adopted elitist selection strategy convergent. But elitist genetic programming can only be applied to the typical global optimization problems, for the more general situations, Rudolph did not propose operational convergence strategy, just recommended the population should experience different feasible solutions as much as possible in a limited period of time. After Rudolp, many other scholars[12,13] have tried to get the convergence conditions of non-elitist genetic programming through the modification of algorithm, but the genetic programming is a very complex Markova process, these studies have not get universal conclusions. Furthermore, there were researchers[14,15] tried to attribute GP algorithm to a special case of generalized simulated annealing algorithm by adjusting the operator parameters, and then derived its convergence conditions. But it is well known that these two algorithms are quiet different in the basic ideals, so there are still a lot of work to be done to blend them together.

On the basis of previous studies[16-18], this paper used the pure probability estimation to study the convergence of genetic programming, by the estimation of characteristic parameters to predict the population's behaviors, and then derived its convergence conditions.

II. Encoding and operators of Genetic programming

A. Encoding method

In GP algorithm, the individual is a tree structure that composed of algorithms and parameters, called algorithm tree. Inside the computer, algorithm tree can be expressed by a 2^h-1 bits decimal integer code string, where h is the maximum tree depth. Every locus in code string corresponds to a determined node position of algorithm tree, and there is a fixed correspondence between the code values and algorithm elements. When encoding, in order to ensure every locus of the code string is non-empty, we introduced transfer function *tr*: $x=tr(x)$, and provide the left and right sub-tree of unary operator are same.

Taking $f(x_1,x_2)=ax_1+b\sin x_2$ as example, when we defined algorithm collection S_O is $\{+,-,\times,/,sin,tr\}$ and parameter collection S_D is $\{x_1, x_2, C\}$ (where: x_1 and x_2 are independent variables, a and b are parameters, C is constant), its algorithm tree and encoding is shown in Figure.1.

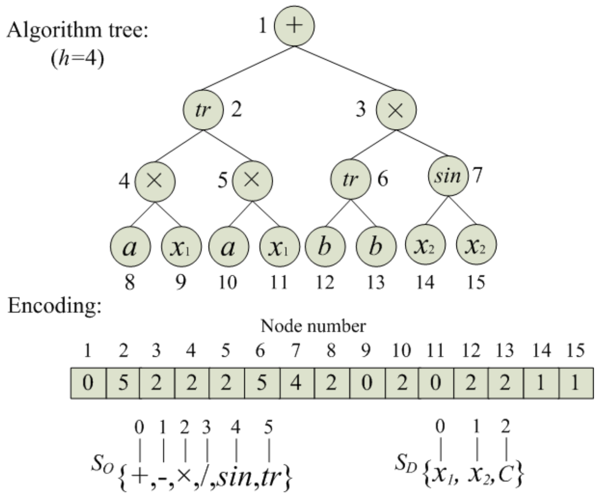

Figure 1. Example of algorithm tree and encoding method

B. Selection operator

Non-elitist GP algorithm based on the individuals' fitness function to assign the survival probability of individuals, and then determined the composition of the next generation population according to this probability. Assuming the feasible region of problem is S and the current population is x, then selection operators can be defined as a functional operator $C(x,f)$ that mapping from space S^N to S^M. Because the selection process is based on the fitness function f and only act on the current population, so the selected population should be included in the original ones:

$$C(x,f) \subseteq x, \forall x \subset S^N \tag{1}$$

The common selection operators in genetic programming are the following:

1) Roulette selection

It is also known as proportional selection; the survival probability of individuals is proportional to their fitness.

$$P\{x_i \in C(x,f)\} = f(x_i) \Big/ \sum_{z\in x} f(Z); f(x_i) \ge 0, \forall x_i \in x \tag{2}$$

2) Sequence selection

There are linear and nonlinear two kinds, this selection operator mainly focus on the numerical values of individual fitness, and do not concern about their numerical difference. Sequence selection first to sort the individuals in accordance with their fitness values, and then determine the survival probability of every individuals based on this sort.

$$\begin{cases} \text{linear: } P\{x_i \in C(x,f)\} = N^{-1} + k_1[(N+1)/2 - i] \\ \text{nonlinear: } P\{x_i \in C(x,f)\} = k_2\,(1-k_2)^{i-1} \Big/ [1-(1-k_2)^N] \end{cases} \tag{3}$$

Where: N is population size, i is the location of individual, k_1 and k_2 are the strength parameters of linear and nonlinear sequence selection.

3) Breeding pool selection

This selection is carried out in a deterministic way. First, the expected survival number of individuals is calculated based on the fitness by the follow formula:

$$N_i = N\,f(x_i) \Big/ \sum_{z\in x} f(Z) \tag{4}$$

Then, the survival probability of every individual in next generation is calculated as follows:

$$\begin{cases} N_i \ge 1: P\{x_i \in C(x,f)\} = 1 \\ N_i < 1: P\{x_i \in C(x,f)\} = N_i \Big/ \sum_{i=1}^{N} [N_i - fix(N_i)] \end{cases} \tag{5}$$

Where: N_i is the expected survival number of individuals, $fix(N_i)$ is the integer portion of N_i.

4) Boltemann selection

This selection method is similar to simulated annealing algorithm, randomly selected two individuals x_i and x_j from the population, the survival probability of individuals x_i is:

$$P\{x_i \in C(x,f)\} = e^{f(x_i)/T} \Big/ [e^{f(x_i)/T} + e^{f(x_j)/T}] \tag{6}$$

Where: T is a selection parameter that similar to the simulated annealing temperature.

5) Competition selection

This selection can be generally divided into two kinds: cellular competition selection and probability competition selection. Cellular competition selection is to select the optimum from a certain number of individuals that adjacent to x_i, while the probability competition selection is first randomly sample several individuals out of population, and then select the optimal one to the next generation.

C. Evolution operator

In genetic programming, the new algorithm tree is generated by population evolution; this process can be defined as a functional operator $E(x)$ that mapping from space S^M to S^N. For the GP algorithm, evolution operator include mutation operator and crossover operator two kinds, are respectively expressed by $E_m(x)$ and $E_C(x)$.

1) Mutation operator

a) Point mutation

This operator only changes the values of mutation point itself, do not influence the rest part of coding. Point mutation is essentially an inversal bit-to-bit allelic variation, the probabilities of individuals do not change and generate new individual under point mutation meet the following estimations:

$$\begin{aligned} &P\{E_m(y) = y, \forall y \in x\} \geq (1 - P_m^{'})^L \\ &P\{E_m(y) \notin x, \forall y \in x\} \geq P_m^{'}[\min(1 - P_m^{'}, P_m^{'})]^{L-1} \end{aligned} \quad (7)$$

Where: $L=2^h-1$ is the encoding length, P_m is mutation rate, $P'_m=P_m(1-n_S^{-1})$, $n_S=max[num(S_O),num(S_D)]$, which $num(S_O)$ and $num(S_D)$ are the element numbers of algorithm collection S_O and parameter collection S_D.

b) Sub-tree mutation

Randomly choose a mutation point in algorithm tree according to mutation rate P_m , and using a new generated sub-tree to instead the original sub-tree that takes the mutation point as root node. Referring to Figure.1, the probabilities of individuals do not change and generate new individual under sub-tree mutation meet the following estimations:

$$\begin{aligned} &P\{E_m(y) = y, \forall y \in x\} \geq 1 - P_m(1 - n_S^{-L}) \\ &P\{E_m(y) \notin x, \forall y \in x\} \geq P_m n_S^{-L} \end{aligned} \quad (8)$$

c) Arrangement mutation

The arrangement mutation can be seen as a special sub-tree mutation, it achieve mutation by changes the elements' order of each nodes in algorithm tree. Referring to Figure.1, the probabilities of individuals do not change and generate new individual under arrangement mutation meet the following estimations:

$$\begin{aligned} &P\{E_m(y) = y, \forall y \in x\} \geq 1 - P_m\,(L-1)(1 - L!^{-1})\big/(2L) \\ &P\{E_m(y) \notin x, \forall y \in x\} \geq P_m\,(L-1)\big/(2LL!) \end{aligned} \quad (9)$$

2) crossover operator

a) Same point crossover

This operator is similar to single-point crossover operator in genetic algorithm, randomly or non-randomly select two parent individuals from population, choose a cross point in the parent algorithm trees according to crossover rate P_C, and then exchange the two sub-trees that take the crossing point as root node, thereby obtaining two new individuals.

Referring to Fig.1, the non-random same point crossover can be regarded as a modification of genetic algorithm's single point crossover operator, the probabilities of individuals do not change under non-random same point crossover meets the following estimation:

$$P\{E_C(y) = y, \forall y \in x\} \geq 1 - P_C(1 - L^{-1}) \quad (10)$$

For the random same point crossover, if the selected parents are exactly the same one, the newborn individuals are still the original ones, and thereby the probabilities of individuals do not change under random same point crossover meets the following estimation:

$$P\{E_C(y) = y, \forall y \in x\} \geq 1 - P_C(1 - N^2) \quad (11)$$

Where: $L=2^h-1$ is the encoding length, P_C is crossover rate, N is population size.

b) Different point crossover

The basic operation of different point crossover is similar to the same point crossover operator, the difference is that the selection of cross points is mutually independent. Similar to the same point crossover, the different point crossover can operate in random and non-random two ways, the probabilities of individuals do not change under non-random different point crossover meets the following estimation:

$$P\{E_C(y) = y, \forall y \in x\} \geq 1 - P_C(1 - L^{-2}) \quad (12)$$

For the random different point crossover, if the selected parents and the cross point position of parents are exactly the same, the crossed individuals are still the original ones, so the probabilities of individuals do not change under random different point crossover meets the following estimation:

$$P\{E_C(y) = y, \forall y \in x\} \geq 1 - P_C(1 - L^{-2}N^{-2}) \quad (13)$$

c) Multi-parental crossover

Multi-parental crossover uses the structural information of multiple parents to guide the generation of new individual. First, randomly sample m parents from the population, and calculated the frequency of element values in every node of each parent algorithm trees. And then, determining the selection probabilities of each node elements in offspring algorithm trees according to this frequency. Finally, roulette selection method is adopted to select the appropriate elements combined into new individuals. In multi-parental crossover, we consider the newborn individual does not change when it is same to any parent, this probability meets:

$$P\{E_C(y) = y, \forall y \in x\} \geq 1 - P_C(1 - m^{1-L}) \quad (14)$$

III. A SUFFICIENT CONDITION OF POPULATION CONVERGE

Genetic programming is a typical Markov process, the changes in population is only depends on the selection and evolution operators that act on the current population, and has nothing to do with the generations. When the selection and evolution operators are clear, the individual distribution in population of the next generation can be determined based on the status of the current population. Therefore, we can fully predict the behaviors of population by the analysis of operators, and thus to study the performance of algorithm.

In order to analyze the convergence, the convergent target should be first determined. We took the collection of problem's satisfactory solutions as the convergent target:

$$B = \{x; f(x) \geq f(y), \forall y \notin B\} \quad (15)$$

For any initial population $x(0) \in S$, if the populations $x(n)$ meets:

$$\lim_{n \to \infty} P\{x(n) \cap B \neq \varnothing\} = 1, \forall B \neq \varnothing \quad (16)$$

Then it said that the population converges to satisfaction set B in probability.

In order to quantitative analysis the abilities of operators that make the population reach some extent satisfactory solution in the process of evolution, several characteristic parameters that describe the evolution abilities of operators were established as follows.

For the selection operators, if satisfactory solutions exist in the original population, then we hope that the selection operation can retain them to the next generation. Therefore, the lower probability bound of selection operators retain the satisfactory solutions in original population was defined as the strength coefficient of selection operator:

$$\tau_C = \inf[P\{C(x) \cap B \neq \varnothing; x \subset S, x \cap B \neq \varnothing\}] \quad (17)$$

As for the evolution operators, we hope that it will be able to generate satisfactory solutions as much as possible on the basis of retain the existing satisfactory solutions.

Therefore, the lower probability bound of the population reach any satisfactory solution after evolution when the original population does not contain satisfactory solution was defined as the aggregation rate of evolution operator:

$$\alpha_E = \inf[P\{E(x)\cap B \neq \varnothing; x \subset S, x\cap B = \varnothing\}] \qquad (18)$$

And the upper probability bound of the population lose original satisfactory solution after evolution was defined as the divergence rate of evolution operator:

$$\delta_E = \sup[P\{E(x)\cap B = \varnothing; x \subset S, x\cap B \neq \varnothing\}] \qquad (19)$$

Furthermore, the lower probability bound of population still belongs to original population after evolution was defined as the stable rate of evolution operator:

$$\gamma_E = \inf[P\{E(y)\in x; x \subset S, \forall y \in x\}] \qquad (20)$$

Among them, aggregation rate α_E reflects the operator's ability to generate new satisfactory solution, the divergence rate δ_E reflects the operator's ability to destroy satisfactory solution, and the stable rate γ_E reflects the limit of operator's evolution ability.

Assuming n is evolution algebra, and the Nth generation population is $x(n)$, the intermediate population that generated from $x(n)$ by select operation is $y(n)$, the probability of $x(n)$ and $y(n)$ converges to any satisfaction set are respectively $P(n)$ and $P'(n)$. Based on the total probability formula, $P'(n)$ can be estimated as follows:

$$\begin{aligned}
&P'(n) = P\{y(n)\cap B\neq\varnothing \mid x(n)\cap B\neq\varnothing\}\cdot P\{x(n)\cap B\neq\varnothing\}\\
&+P\{y(n)\cap B\neq\varnothing \mid x(n)\cap B=\varnothing\}\cdot P\{x(n)\cap B=\varnothing\} \qquad (21)\\
&\geq P\{y(n)\cap B\neq\varnothing \mid x(n)\cap B\neq\varnothing\}\cdot P\{x(n)\cap B\neq\varnothing\} \geq \tau_C(n)P(n)
\end{aligned}$$

Considering that $y(n)\cap B\neq\varnothing$ and $y(n)\cap B=\varnothing$ are two incompatible events, $P(n+1)$ meets the following estimation:

$$\begin{aligned}
&P(n+1) = P\{x(n+1)\cap B\neq\varnothing \mid y(n)\cap B\neq\varnothing\}\cdot P\{y(n)\cap B\neq\varnothing\}\\
&+P\{x(n+1)\cap B\neq\varnothing \mid y(n)\cap B=\varnothing\}\cdot P\{y(n)\cap B=\varnothing\} \qquad (22)\\
&\geq [1-\delta_E(n)]\cdot P'(n) + \alpha_E P\{y(n)\cap B=\varnothing\}
\end{aligned}$$

According to formula (1), when $x(n)\cap B=\varnothing$, there must be $y(n)\cap B=\varnothing$, then:

$$P\{y(n)\cap B=\varnothing\} \geq P\{x(n)\cap B=\varnothing\} = 1-P(n) \qquad (23)$$

Substituting the formula (21) and (23) to formula (22), the recursive estimation of probability $P(n)$ with evolution algebra was derived as follows:

$$P(n+1) \geq [1-\delta_E(n)]\tau_C(n)P(n) + \alpha_E(n)[1-P(n)] \qquad (24)$$

Let $a(n)=\alpha_E(n)$ and $b(n)=[1-\delta_E(n)]\tau_C(n)-\alpha_E(n)$, then the convergence problem of genetic programming becomes a mathematical problem: Under what conditions, the Markov series $P(n+1)\geq a(n)+b(n)P(n)$ converges to 1?

It can be proved that when $a(n)$ and $b(n)$ meet[16]:

$$\lim_{n\to\infty} a(n)/[1-b(n)] = 1;\quad \sum_{n=1}^{\infty}[1-b^2(n)] = \infty \qquad (25)$$

There is $\lim_{n\to\infty}P(n)=1$, and then it said $P(n)$ converges to 1 in probability.

A sufficient term for population converges in probability can be derived from formula (25):

$$\lim_{n\to\infty}\{1-\tau_C(n)[1-\delta_E(n)]\}/\alpha_E(n)=0;\ \sum_{n=1}^{\infty}\alpha_E(n)[2-\alpha_E(n)]=\infty \qquad (26)$$

IV. PARAMETER ESTIMATION OF OPERATORS

The parameters τ_C, α_E, δ_E and γ_E in the formula (26) are all the random probabilities that related to operators. In this section, the ranges of those characteristic parameters that describe the evolution abilities of operators were estimated to establish the relationship with specific operators.

A. Selection operators

1) Roulette selection

The survival probability of individuals in population after roulette selection follows formula (2), then the probability of retain the satisfactory solutions in population is:

$$P\{C(x)\cap B\neq\varnothing; x\in S, x\cap B\neq\varnothing\} = \sum_{y\in B} f(y) \Big/ \sum_{z\in x} f(Z) \qquad (27)$$

Let $\rho=max\{f(a)/f(b); f(b)>f(a)\geq 0, \forall a,b\in x\}$ is the maximum fitness ratio of population, then the strength coefficient's estimation of roulette selection operator meets:

$$\tau_C = \inf[\sum_{y\in B} f(y) \Big/ \sum_{z\in x} f(Z)] \geq [1+(N-1)\rho]^{-1} \qquad (28)$$

2) Sequence selection

Deriving from formula (3), the survival probabilities of the optimal individual in original population under linear and nonlinear sequence selection were respectively $N^{-1}+k_1(N+1)/2$ and $k_2/[1-(1-k_2)^N]$, then the strength coefficient's estimations of sequence selection operator meet:

$$\begin{cases}\text{linear: } \tau_C \geq N^{-1}+k_1(N+1)/2\\ \text{nonlinear: } \tau_C \geq k_2/[1-(1-k_2)^N]\end{cases} \qquad (29)$$

3) Breeding pool selection

According to formula (5), when the optimal individual's expected survival number is no less than 1, it will certainly survive in the next generation. Otherwise, this selection will be similar to roulette selection, let $f_{max}=max\{f(a), \forall a\in x\}$, the estimations of strength coefficient meet:

$$\begin{cases} f_{max} \geq \sum_{z\in x} f(Z)/N:\ \tau_C = 1\\ f_{max} < \sum_{z\in x} f(Z)/N:\ \tau_C \geq [1+(N-1)\rho]^{-1}\end{cases} \qquad (30)$$

Where: ρ is the maximum fitness ratio of population.

4) Boltzmann selection

Deriving from formula (6), the survival probability of the optimal individual in population after M times independent Boltzmann selection meets:

$$P\{x_{opt}\in C(x,f)\} = \frac{MN^{-1}}{1+e^{[f(z)-f(x_{opt})]/T}},\ z\in x \qquad (31)$$

Let $\rho'=min\{|f(a)-f(b)|, \forall a,b\in x\}$ is the minimum fitness difference of population, the strength coefficient's estimation of Boltzmann selection meets:

$$\tau_C \geq M[N(1+e^{-\rho'/T})]^{-1} \tag{32}$$

5) Competition selection

Cellular competition selection can guarantee the optimal individual of population survive in probability 1, so its τ_C=1. And for the probability competition selection, the survival probability of optimum is actually its selected probability in the random sampling process, the estimation of its strength coefficient meets:

$$\tau_C \geq M[1-(1-N^{-1})^m] \tag{33}$$

B. Evolution operator

Considering the two events, one is the population lost the original satisfactory solution after evolution, and the other is the population changed after evolution operation. Obviously, the former is a sub-event of the latter, because lost satisfactory solution means that the population will inevitably change, so the divergence rate δ_E of evolution operators must not be greater than the lower probability bound of the individuals change in evolution, that is:

$$\delta_E \leq \inf\{\prod_{z\in B}[1-P\{E(Z)=Z\}]\} = [1-P\{E(y)=y, \forall y \in x\}]^N \tag{34}$$

Similarly, consider the following two events: population generate new individual and population generate satisfactory solution after evolution. The former is a sub-event of the latter, because generate satisfactory solution means a new individual was inevitably generated in population. So the aggregation rate α_E of evolution operators must not be less than the upper probability bound of generating new individuals in population, that is:

$$\alpha_E \geq \sup\{1-\prod_{y\in x}[1-P\{E(y)\notin x\}]\} = 1-[1-P\{E(y)\notin x\}]^N \tag{35}$$

Further, when all individuals in population are not change, the population will apparently stable, thereby the estimation of stable rate γ_E of evolution operators meet:

$$\gamma_E \geq \inf[\prod_{y\in x} P\{E(y)=y, \forall y \in x\}] = [P\{E(y)=y, \forall y\in x\}]^N \tag{36}$$

Substituting the formula (7)-(14) to formula (34)-(36), the parameter estimations of various common evolution operators were derived as shown in Table.1.

TABLE I. THE PARAMETER ESTIMATION OF COMMON EVOLUTION OPERATORS

Operator	**Parameter Estimations**
Point mutation	$\alpha_{Em} \geq N[P_m(1-n_S^{-1})]^L$, $\delta_{Em} \leq [LP_m(1-n_S^{-1})]^N$ $\gamma_{Em} \geq 1-NLP_m(1-n_S^{-1})$
Sub-tree mutation	$\alpha_{Em} \geq NP_m n_S^{-L}$, $\delta_{Em} \leq [P_m(1-n_S^{-L})]^N$ $\gamma_{Em} \geq 1-NP_m(1-n_S^{-L})$
Arrangement mutation	$\alpha_{Em} \geq NP_m(L-1)/(2LL!)$ $\delta_{Em} \leq [P_m(L-1)(1-L!^{-1})/(2L)]^N$ $\gamma_{Em} \geq 1-NP_m(L-1)(1-L!^{-1})/(2L)$
Non-random same point crossover	$\delta_{Ec} \leq [P_C(1-L^{-1})]^N$, $\gamma_{Ec} \geq [1-P_C(1-L^{-1})]^N$
Random same point crossover	$\delta_{Ec} \leq [P_C(1-N^{-2})]^N$, $\gamma_{Ec} \geq [1-P_C(1-N^{-2})]^N$
Non-random different point crossover	$\delta_{Ec} \leq [P_C(1-L^{-2})]^N$, $\gamma_{Ec} \geq [1-P_C(1-L^{-2})]^N$
Random different point crossover	$\delta_{Ec} \leq [P_C(1-L^{-1}N^{-2})]^N$, $\gamma_{Ec} \geq [1-P_C(1-L^{-1}N^{-2})]^N$
Multi-parental crossover	$\delta_{Ec} \leq [P_C(1-m^{1-L})]^N$, $\gamma_{Ec} \geq [1-P_C(1-m^{1-L})]^N$

For the compound evolution operator that composed by crossover and mutation operators, taking into accounting that $P\{E(x)\cap B=\varnothing \mid x\cap B=\varnothing\} \geq P\{E(y)\in x,\ \forall y\in x \mid x\cap B=\varnothing\}$, its parameter estimations can be derived by the total probability formula, that is:

$$\begin{cases} \alpha_E \geq \alpha_{Ec}(1-\delta_{Em})+\alpha_{Em}(1-\alpha_{Ec}) \geq \alpha_{Em}\gamma_{Ec} \\ \delta_E \leq \delta_{Ec}(1-\alpha_{Em})+\delta_{Em},\ \gamma_E \geq \gamma_{Em}\gamma_{Ec} \end{cases} \tag{37}$$

V. CONVERGENCE STRATEGIES OF GENETIC PROGRAMMING

Based on the conclusions of the section 3 and section 4, the convergence strategies of genetic programming under different evolution modes have been derived in this section.

1) Roulette selection

According to formula (28), the strength coefficient τ_C of roulette selection under any determined fitness function is always have a lower bound that greater than 0. Therefore, in order to meet the convergence conditions (formula (26)), the fitness function of roulette selection should be corrected to make the maximum fitness ratio of population can be adjusted with the evolution algebra.

Introduced simulated annealing operator:

$$g(x) = e^{f(x)\{T[1+f(x)]\}^{-1}} \tag{38}$$

Then the maximum fitness ratio ρ meets:

$$\rho \leq \max\{e^{\frac{f(a)-f(b)}{[1+f(a)][1+f(b)]T}}; f(a), f(b) \geq 0\} \leq e^{1/T} \tag{39}$$

Corresponding thereto, there is:

$$\tau_C \geq [1+(N-1)e^{1/T}]^{-1} \tag{40}$$

Substituting the formula (37) and (40) to formula (26), the convergence strategies of roulette selection under different evolution operators were derived from the estimations in Table.1:

$$\begin{cases} \text{only point mutation: } P_m = n^{-1/L}, N > L, T = -(2\ln n)^{-1} \\ \text{only sub-tree mutation: } P_m = n^{-1}, N \geq 2, T = -(2\ln n)^{-1} \\ \text{only arrangement mutation: } P_m = n^{-1}, N \geq 2, T = -(2\ln n)^{-1} \\ \text{any crossover and point mutation:} \\ P_m = n^{-1/L}, P_C = n^{-2/N}, N > L, T = -(2\ln n)^{-1} \\ \text{any crossover and sub-tree mutation:} \\ P_m = n^{-1}, P_C = n^{-2/N}, N \geq 2, T = -(2\ln n)^{-1} \\ \text{any crossover and arrangement mutation:} \\ P_m = n^{-1}, P_C = n^{-2/N}, N \geq 2, T = -(2\ln n)^{-1} \end{cases} \tag{41}$$

2) Sequence selection

When the strength parameters meet:

$$\begin{cases} \text{linear: } k_1 = 2[N(1-n^{-2})-1]/[N(N+1)] \\ \text{nonlinear: } k_2 = (1-n^{-2})/2 \end{cases} \tag{42}$$

And the other parameters are valued by formula (41), the convergence conditions (formula (26)) were met, populations would converge in probability.

3) Breeding pool selection

According to formula (26), the lower strength coefficient bound of breeding pool selection is same to roulette selection, so they can take the same convergence strategies.

4) Boltzmann selection

When the simulated annealing temperature T meets:

$$\begin{cases} M \le (1-n^{-2})N: T = \{\ln[(N-M)n^2 - N] - \ln[(n^2-1)N]\}^{-1} \\ M > (1-n^{-2})N: T = \{\ln[(M-N)n^2 + M] - \ln(n^2 N)\}^{-1} \end{cases} \quad (43)$$

And the other parameters are valued by formula (41), the convergence conditions (formula (26)) were met, populations would converge in probability.

5) Competition selection

For the cellular competition selection and the probability competition selection that the sampling number $m \ge ln(1\text{-}M^{-1})/ln(1\text{-}N^{-1})$, their strength coefficient $\tau_C=1$. At this time, the competition selection is actually becomes an elitist selection, it inevitably convergent according to the findings of Rudolph[11].

As for the probability competition selection that sampling number m<ln(1-M-1) /ln(1-N-1), unless its population size N=1, or the lower bound of its strength coefficient must be greater than 0, it is failed to find its effective convergence strategies under the convergence conditions in this paper.

VI. THE PAUSE TIME OF GENETIC PROGRAMMING

Pause time is a part of the convergence mechanism, it reflects the convergence speed of algorithm. In the section 5, whether genetic programming can convergence and how to convergence were discussed, and the convergence strategies under different evolution modes were given as well. But those strategies were all derived from the limit characteristics of population evolution, can only guarantee that the evolution process can converge to a satisfactory solution, has no further descripted population converge in what time or before what time. The evolution algebra in practice obviously can not be infinite, so it is necessary to estimate the evolution algebra of population reach any satisfactory solution in algorithm design.

Since the evolution is a random process, the generation that it reaches any satisfactory solution is also random, so our concerned is the relationship between the evolution algebra of population reaches any satisfactory solution for the first time and the operators. According to formula (25), there exists a positive integer φ that respects to any positive number ε to make the following formula true:

$$a(n) < \varepsilon[1-b(n)] - b(n) + 1; \ \forall n > \varphi \quad (44)$$

Took a genetic programming that adopt roulette selection as example, when the crossover rate P_C, mutation rate P_m and the simulated annealing temperature T are valued by Table.1, the integer φ that make formula (44) come into existence were derived as shown in Table.2.

TABLE II. THE GENERATION THAT MAKE FORMULA (44) COME TRUE

Operator	Generations
Only point mutation	$\varphi > [(\varepsilon^{-1}-1)^L L^N]^{L/(N-L)}(1-n_S^{-1})^L$
Only sub-tree mutation	$\varphi > [(\varepsilon^{-1}-1)n_S^L N^{-1}]^{1/(N-1)}(1-n_S^{-L})^{N/(N-1)}$
Only arrangement mutation	$\varphi > [(\varepsilon^{-1}-1)(L!-1)N^{-1}]^{1/(N-1)}(1-L^{-1})/2$
Non-random same point crossover and point mutation	$\varphi > [(\varepsilon^{-1}-1)^L L^{2N}]^{L/(N-L)}(1-n_S^{-1})^L$
Non-random same point crossover and sub-tree mutation	$\varphi > [(\varepsilon^{-1}-1)n_S^L L^N N^{-1}]^{1/(N-1)}(1-n_S^{-L})^{N/(N-1)}$
Non-random same point crossover and arrangement mutation	$\varphi > [(\varepsilon^{-1}-1)(L!-1)N^{-1}L]^{1/(N-1)}(L-1)/2$
Random same point crossover and point mutation	$\varphi > [(\varepsilon^{-1}-1)^L L^N N^{2N}]^{L/(N-L)}(1-n_S^{-1})^L$
Random same point crossover and sub-tree mutation	$\varphi > [(\varepsilon^{-1}-1)n_S^L N^{2N-1}]^{1/(N-1)}(1-n_S^{-L})^{N/(N-1)}$
Random same point crossover and arrangement mutation	$\varphi > [(\varepsilon^{-1}-1)(L!-1)N]^{1/(N-1)} N^2(1-L^{-1})/2$
Non-random different point crossover and point mutation	$\varphi > [(\varepsilon^{-1}-1)^L L^{3N}]^{L/(N-L)}(1-n_S^{-1})^L$
Non-random different point crossover and sub-tree mutation	$\varphi > [(\varepsilon^{-1}-1)n_S^L L^{2N} N^{-1}]^{1/(N-1)}(1-n_S^{-L})^{N/(N-1)}$
Non-random different point crossover and arrangement mutation	$\varphi > [(\varepsilon^{-1}-1)(L!-1)N^{-1}L^2]^{1/(N-1)} L(L-1)/2$
Random different point crossover and point mutation	$\varphi > [(\varepsilon^{-1}-1)^L L^{2N} N^{2N}]^{L/(N-L)}(1-n_S^{-1})^L$
Random different point crossover and sub-tree mutation	$\varphi > [(\varepsilon^{-1}-1)n_S^L L^N N^{2N-1}]^{1/(N-1)}(1-n_S^{-L})^{N/(N-1)}$
Random different point crossover and arrangement mutation	$\varphi > [(\varepsilon^{-1}-1)(L!-1)LN]^{1/(N-1)} N^2(L-1)/2$
Multi-parental crossover and point mutation	$\varphi > [(\varepsilon^{-1}-1)n_S^L L^N m^{2N-1}]^{1/(N-1)}(1-n_S^{-L})^{N/(N-1)}$
Multi-parental crossover and sub-tree mutation	$\varphi > [(\varepsilon^{-1}-1)n_S^L m^{N(L-1)} N^{-1}]^{1/(N-1)}(1-n_S^{-L})^{N/(N-1)}$
Multi-parental crossover and arrangement mutation	$\varphi > [(\varepsilon^{-1}-1)(L!-1)m^{N(L-1)} N^{-1}]^{1/(N-1)}(1-L^{-1})/2$

Substituting the formula (44) to formula (24), there is:

$$1-P(n+1) < \varepsilon + [1-P(\varphi)-\varepsilon]\prod_{i=\varphi}^{n} b(i) \quad (45)$$

Since $b(n) \le 1\text{-}\alpha_E(n)$, and $\alpha_E(n) << 1$ when $n > \varphi$, so there is:

$$\prod_{i=\varphi}^{n} b(i) \le [1-\alpha_E(i)]^{n-\varphi} \approx 1-(n-\varphi)\alpha_E(n) \quad (46)$$

Deriving from formula (46), if a positive integer R can be found to make $(R\text{-}\varphi)\alpha_E(R) \ge 1\text{-}\varepsilon$ come into existence, then for any $n > R$, there is:

$$P(n+1) > 1-2\varepsilon \quad (47)$$

It means that the probability of population reaches any satisfactory solution within R+1 generation should be greater than 1-2ε, the conditions that make formula (47) come into existence were derived as shown in Table.3.

TABLE III. THE CONDITIONS THAT MAKE FORMULA (47) COME TRUE

Operator	Conditions
Only point mutation	$\begin{cases} N \le (1-\varepsilon)(1-n_S^{-1})^{-L}: \ R = \varphi + 1 \\ N > (1-\varepsilon)(1-n_S^{-1})^{-L}: \\ R > N\varphi[N-(1-\varepsilon)(1-n_S^{-1})^{-L}]^{-1} \end{cases}$

Any crossover and point mutation	$\begin{cases} N \le (1-\varepsilon)(1-n_S^{-1})^{-L}(1-N\varphi^{2/N})^{-1}: \\ R = \varphi + 1 \\ N > (1-\varepsilon)(1-n_S^{-1})^{-L}(1-N\varphi^{2/N})^{-1}: \\ R > \dfrac{N\varphi(1-n_S^{-1})^{L}(1-N\varphi^{2/N})}{N(1-n_S^{-1})^{L}(1-N\varphi^{2/N})+\varepsilon-1} \end{cases}$
Only sub-tree mutation	$\begin{cases} N \le (1-\varepsilon)n_S^L:\ R = \varphi + 1 \\ N > (1-\varepsilon)n_S^L:\ R > N\varphi[N-(1-\varepsilon)n_S^L]^{-1} \end{cases}$
Any crossover and sub-tree mutation	$\begin{cases} N > (1-\varepsilon)(1-N\varphi^{2/N})^{-1}n_S^L: \\ R > \dfrac{N\varphi(1-N\varphi^{2/N})}{N(1-N\varphi^{2/N})-(1-\varepsilon)n_S^L} \\ N \le (1-\varepsilon)(1-N\varphi^{2/N})^{-1}n_S^L:\ R = \varphi + 1 \end{cases}$
Only arrangement mutation	$\begin{cases} N \le 2(1-\varepsilon)(L-1)^{-1}LL!:\ R = \varphi + 1 \\ N > 2(1-\varepsilon)(L-1)^{-1}LL!: \\ R > N\varphi[N-2(1-\varepsilon)LL!(L-1)^{-1}]^{-1} \end{cases}$
Any crossover and arrangement mutation	$\begin{cases} N \le 2(1-\varepsilon)(L-1)^{-1}(1-N\varphi^{2/N})^{-1}LL!: \\ R = \varphi + 1 \\ N > 2(1-\varepsilon)(L-1)^{-1}(1-N\varphi^{2/N})^{-1}LL!: \\ R > \dfrac{N\varphi(L-1)(1-N\varphi^{2/N})}{N(L-1)(1-N\varphi^{2/N})-2(1-\varepsilon)LL!} \end{cases}$

Through Table.1 and Table.2, the generation of genetic programming reaches any satisfactory solution under various evolution modes can be estimated. Without less of generality, we take ε=1/8, then for the genetic programming that adopts roulette selection, non-random same point crossover and sub-tree mutation, when its population size N=60, maximum tree depth h=6, the larger element number of algorithm and parameter collection n_S=6, and the other parameters are valued by Table.1, the R=763. It means that, if this evolution process was independently carried out 10 times, then the probability of find out at least one satisfactory solution within 764 generation should be not less than $1-10^{-6}$. If the evolution algebra was increased to 840, there is $P(840)>0.9991$, then it is only need to independently carry out this evolution twice, at least one satisfactory solution can guarantee to be found out in the probability of not less than $1-10^{-6}$.

Because of the limitation of paper space, this section only derived the pause time of genetic programming under roulette selection. Other cases can also be analyzed by the similar process, and will not go into here.

VII. CONCLUSIONS AND ANALYSIS

The convergence mechanism and strategies of non-elitist genetic programming were studied in this paper, and the conclusions and analysis were summarized as follows:

(1) The convergence study of genetic programming in this paper is based on the operators' analysis and estimates, it is different from the traditional mode analysis that based on gene encoding. This study approach has downplayed the biological background of genetic programming, and thereby provides the basis for a broader application of GP algorithm.

(2) The convergent target of this study is the collection of satisfactory solutions, and the optimal solution is inevitably a satisfactory solution, so it can be found out by the strategies that derived in paper.

(3) Through the analysis of the pause time, it can be found that the mutation operators seems to play a greater impact on the convergence, while the crossover operators do not play a decisive role, but conversely decrease the convergence rate of algorithm. This conclusion has been supported by some other studies[6,16,19,20].

(4) Formula (26) is not the necessary condition of genetic population converges in probability, and there are many groups of parameters that meet formula (26), the convergence strategies derived in this paper is just one simpler group. It means there may exist a great difference in the convergence performance of the same problem under different evolution modes, so it is necessary to study the influence of different modes to the evolution process of certain problems, and thus to determine the optimal evolutionary strategy.

REFERENCES

[1] J.R.Koza. Genetic Programming: On the Programming of Computers by Means of Natural Selection [M]. Cambridge, MA: MIT press, 1994.

[2] J.R.Koza. Genetic Programming Ⅱ: Automatic Discovery of Reusable Programs [M]. Cambridge, MA: MIT press, 1992.

[3] J.R.Koza, M.A.Keane, M.J.Streeter, et al. Genetic programming Ⅳ: Routine human-Competitive machine intelligence [M]. Nerwell, MA: Kluwer academic publishers, 2003.

[4] H.Mohammad. Analysis on the evolution of the discourse on computer software and programming languages in the light of literary genres and power-knowledge [J]. Computers in Human Behavior, 2010, 26(3): 464-473.

[5] J.R.Koza. Automatic creation of human-competitive programs and controllers by means of genetic programming [J]. Genetic Programming and Evolvable Machines, 2000, 1(2): 121-164.

[6] D.Lin, M.Q.Li, J.S.Kou. A theorem on the Convergence of genetic programming [J]. Journal of Xiamen University (Natural Science), 2000, 39(1): 125-127.

[7] Thomas Back. Handbook of evolutionary computation [M]. Oxford University Press, 1997.

[8] J.Qin, L.S.Kang, T.P.Chen. The convergence analysis and algorithm improvement of computation algorithm [J]. Computer Engineering and Application, 2003(19): 91-93.

[9] M.Vose, A.H.Wright. Simple genetic algorithms with linear fitness [J]. Evolutionary Computation, 1995, 2(4): 347-368.

[10] A.H.Wright, M.Vose. Finiteness of the fixed point for Simple genetic algorithms [J]. Evolutionary Computation, 1996, 3(3): 299-309.

[11] G.Rudolph. Convergence analysis of canonical genetic algorithm [J]. IEEE Tran Neural Network, 1994, 5(1): 96-101.

[12] R.Cerf. Asymptotic convergence of genetic algorithms [J]. Advance in applied probability, 1998, 30(2): 521-550.

[13] J.Qin, L.S.Kang. A convergence analysis framework for multi-objective optimization evolutionary algorithm [J]. Computer Application Reasearch, 2005(2): 68-70.

[14] J.Suzuki. A further result on the markov chain model of genetic algorithms and its application to a simulated annealing-like strategy [J]. IEEE Tran on systems, Man and Cybernetics-part B: Cybernetics, 1998(1): 95-102

[15] A.Trouve. The evolutionary computation and simulated annealing [J], SIAM journal on control and optimization, 1997(34): 966-986.

[16] Q.H.Duan. The convergence theory and strategies for evolutionary algorithm [Ph.D]. Xi'an: Xi'an Jiaotong University, 2002.

[17] J.S.Zhang, Z.B.Xu, Y.Liang. The whole annealing genetic algorithm and its convergence necessaryand sufficient condition [J]. Science in China (Series E), 1997, 27(2): 154-164.

[18] W.X.Zhang, Z.B.Xu, Z.K.Nie, et al. Probability convergence theorems of genetic algorithms [J]. Journal of Engineering Mathematics, 2001, 18(4): 1-11.

[19] X.M.Dai, R.Sun, R.M.Zou, et al. Global convergence analysis of non-crossover genetic algorithm and its application to optimization [J]. Journal of systems engineering and electronics, 2002, 13(2): 84-91.

[20] J.Jana, R.Zbynek. Influence of multiple crossover and mutation to the convergence of genetic optimization (Article number 4630326) [C]. Proceedings on the 17th international conference on microwaves, radar and wireless communications (MIKON 2008), Wroclaw, Poland, May 19-May 21, 2008: 419-423.