Parallel performance analysis of nonlinear equations algorithm based on hybrid genetic algorithm

Yunwen Yang*
Hubei College of Chinese Medicine, Jingzhou, Hubei, 434000, China

*Corresponding author: unweny@163.com

Abstract. For a long time, people have done a lot of research on nonlinear equations in theory and numerical calculation. In this paper, the basic theories and methods of genetic algorithm and traditional algorithm are expounded, and the advantages of genetic algorithm and quasi-Newton algorithm are analyzed, thus a new hybrid genetic algorithm for solving nonlinear numerical problems is proposed, and the effectiveness of the algorithm is verified by numerical examples. The hybrid algorithm gives full play to the group search and global convergence of genetic algorithm, and effectively overcomes the initial point sensitivity problem of classical algorithm. At the same time, the classical algorithm is introduced into genetic algorithm for local search, which overcomes the shortcomings of slow convergence speed and poor accuracy of genetic algorithm. The algorithm in this paper provides an effective way to solve nonlinear equations from another angle.

Keywords: Hybrid genetic algorithm: Nonlinear equations; parallel computing

1. Introduction
There are mature and effective algorithms for solving linear equations, but there is no good method for solving nonlinear equations, especially for solving large-scale nonlinear equations [1-2]. Starting from the classical Newton iterative method, although many iterative methods have been proposed, no better method has been found for large-scale engineering application problems, especially the applicable algorithm for parallel computing [3].

Genetic algorithm is an intelligent optimization algorithm that imitates natural selection and genetic mechanism. Implicit parallelism and group global search are its two remarkable features, which have strong robustness and unique superior performance for solving some large and complex nonlinear systems [4-5]. Therefore, this paper attempts to solve nonlinear equations from the perspective of optimization and iteration. Based on the respective characteristics of genetic algorithm and classical algorithm (quasi-Newton method), an efficient and reliable hybrid genetic algorithm is designed for solving highly nonlinear equations.

2. Problem description
The general form of real function nonlinear equations (assuming that there are $n$
variables \( X = [x_1, x_2, \cdots, x_n] \) and \( n \) equations and there are solutions) is

\[
\begin{align*}
  f_1(x_1, x_2, \cdots, x_n) &= 0 \\
  f_2(x_1, x_2, \cdots, x_n) &= 0 \\
  & \vdots \\
  f_n(x_1, x_2, \cdots, x_n) &= 0
\end{align*}
\]  

(1)

Solving this system of equations is equivalent to solving the following extremum optimization problem:

\[
\begin{align*}
  \text{Find} & : X = [x_1, x_2, \cdots, x_n] \in \Phi \\
  \text{Min} & : f(X) = \sum_{i=0}^n f_i^2(X)
\end{align*}
\]  

(2)

In formula (2), \( \Phi \) is the interval of the solution of equations, and when the minimum value of \( f(X) \) is 0, the corresponding \( X \) is the solution of equations.

3. Hybrid genetic algorithm

In this paper, the idea of combining genetic algorithm with quasi-Newton method is to use quasi-Newton method for secondary optimization on the basis of optimization design with genetic algorithm, and take the obtained result as the optimal design result, that is, first run genetic algorithm to find a point near the best point, and use it as the initial point of quasi-Newton method to find the minimum value of the objective function [6].

Quasi-Newton method is a local convergence algorithm in theory, which has strict requirements on initial value. When it is far from the exact solution, it is divergent, and when it is closer to the exact solution, its convergence speed is faster. The numerical algorithm is as follows:

\[
\begin{align*}
  x^{(k+1)} &= x^{(k)} - A_k^{-1}F(x^{(k)}) \\
  S_k &= x^{(k+1)} - x^{(k)}, y_k = F(x^{(k+1)}) - F(x^{(k)}), k = 0, 1, 2 \cdots \\
  A_k+1 &= A_k + (y_k - A_k S_k) S_k^T / \|S_k\|^2
\end{align*}
\]  

(3)

Compared with Newton's method, quasi-Newton's method avoids calculating derivatives, and the calculation amount of each iteration is much smaller. When the dimension is large or derivative calculation is difficult, quasi-Newton's method can obviously improve the calculation efficiency, but its iteration times are more than Newton's method, and like Newton's method, it has strict requirements on the initial value of iteration, so it can only be applied to small-scale convergence. For complex multi-rigid-body nonlinear systems with changeable States, this single algorithm cannot meet the calculation requirements.

(1) Choose coding scheme (such as binary, dynamic parameter coding, multi-value coding, floating-point coding, etc.), encode the search space, and transform the variables into chromosomes;

(2) Select appropriate parameters, including population size \( N \), crossover probability \( P_c \), mutation probability \( P_m \), and termination evolution criterion;

(3) Determine the fitness function \( f(x) > 0 \):

(4) Set \( t = 0 \), and randomly generate initial population \( X(0) = \{x_1(0), x_2(0), \cdots, x_n(0)\} \);

(5) The fitness value \( f_i(X), i = 1, 2, \cdots, N \) of each chromosome is calculated. At the same time, the total fitness value \( F = \sum_i f_i \) of the population is calculated;

(6) Select a number of individuals \( X(t) \) selected from \( X(t) \) by using operators to reproduce, and
the probability of being selected is \( P_i = \frac{f_i}{F} \).

(7) Cross according to a certain probability of crossing \( P_c \). Two new individuals are generated by crossover operation at the determined position. Repeat this process until a new population \( X'(t) \) is formed;

(8) Variation according to a certain probability of variation. Randomly change its value for each individual to form a new generation of population \( X(t+1) = \{x_1(t+1), x_2(t+1), \ldots, x_n(t+1)\} \);

(9) Check whether the termination criteria are met. If it is satisfied, the individual with the largest fitness in \( X(t+1) \) is output as the optimal solution, and the calculation is terminated. Otherwise, let \( t+1 = t \), and go to (5).

The calculation flow chart of genetic algorithm is shown in Figure 1:

3.1 Genetic algorithm design
- Fitness design
  How to design fitness function is the first problem to solve equations successfully by genetic algorithm. Equation (1) is normalized in the form of \( \Phi_i(X)/A_i = 1 \) (if \( A_i = 0 \), no processing is needed), so the fitness function can be designed as:

\[
\text{Fitness}(X) = - \min_{x_j, x_k, \ldots, x_l} \sum_{i=1}^{n} \left( \Phi_i(X)/A_i - 1 \right)^2
\]

The larger the fitness value, the better the approximation degree of the solution.
- Coding design
  The decision variables of mixed integer programming have both integer and real types. Genetic algorithm has higher search efficiency for binary coding of integer variables [10], while real coding of real variables can effectively reduce chromosome coding length and can be better combined with chaos optimization algorithm to improve search accuracy.

Obviously, each individual in the population is an \( n \)-dimensional vector \( x = (x_1, x_2, \ldots, x_n)^T \) on the
domain $D$. A domain $D = \left\{ (x_1, x_2, \cdots, x_n) \mid l_i \leq x_i \leq u_i, i = 1, 2, \cdots, n \right\}$ can be set, but in other cases, we can handle it similarly. A binary string vector $X = (X_1, X_2, \cdots, X_n)$ is constructed, each component $X_i$ of which is a binary string of length $L$. It is easy to know that under the action of mapping $x_i = (u_i - l_i)(x_i / (2^L - 1)) + l_i$, any binary string $X$ uniquely corresponds to an individual $\vec{T} = (x_1, x_2, \cdots, x_n)^T$ on the area $D$. In fact, the set composed of all binary strings $X$ with length $M = n \cdot L$ can be used as a discrete representation of region $D$.

Therefore, the hybrid coding method is adopted. Each chromosome is divided into two gene segments, one gene segment corresponds to integer variables, and all code values are binary characters; the other gene segment corresponds to real type real quantity, and all code values are real numbers [9].

- Genetic operation design
  Genetic operator design includes the design of crossover operator, mutation operator and selection operator.

  (1) Crossover operator

  Assuming that there is an arithmetic crossover between two entities $X_1$ and $X_2$, the two new entities produced after crossover operation are:

  \[
  X_1' = aX_1 + (1 - a)X_2 \quad X_2' = (1 - a)X_1 + aX_2
  \]

  $a$ is a parameter in the interval $[0,1]$, which can be a constant or a variable determined by evolution.

  This paper chooses the random number in the interval $[0,1]$.

  (2) Mutation operator

  Genetic algorithm emphasizes the crossover function, and the evolution of solution is mainly accomplished by selection mechanism and crossover strategy. Mutation operator aims to make the population escape evolution and fall into a local area of search space.

  According to the probability $P_m$, the $t$ generation population was mutated. In this way, about $p_m \cdot N \cdot M$ bits will be mutated in the $t$ generation population. In order to improve the detection ability of process 1 and process 2 in the late evolution stage and provide new excellent individuals for process 0, the value rule of variation rate $P_m$ of the three processes is shown in Table 1.

  \[
  \text{Table 1 Selection of mutation rate}
  \begin{array}{c|c|c}
  \text{Process} & \text{Process 0} & \text{Process 1} & \text{Process 2} \\
  \text{Value} & 0.001 \leq p_m \leq 0.002 & 0.002 \leq p_m \leq 0.004 & 0.003 \leq p_m \leq 0.006 \\
  \text{Value} & 0.004 \leq p_m \leq 0.008 & 0.005 \leq p_m \leq 0.01 & 0.006 \leq p_m \leq 0.008 \\
  \end{array}
  \]

  This chapter draws lessons from the method adopted in reference [11], when mutation operation is needed, a chromosome is randomly selected in the search space to replace the mutated chromosome.

  (3) Selection operator

  Adopt the tournament selection method. The parameter selected for the tournament is the competition scale, and its value range is $[2, N_{\text{ind}}]$. In order to prevent the premature phenomenon by this method, the scale transformation of fitness function can be used to solve it. In this paper, the top-level scale transformation is adopted. The parameter of this transformation is Quantity.
3.2 Complexity of parallel computing

Parallel computation can’t be used only in serial computation of residual $r^{k+1} = F(x^{k+1})$, but in other steps. When residual $r^{k+1}$ is not dominant in the whole computation, the acceleration of the new algorithm should be close to the number of processors $z$.

Assuming that the number of blocks $m$ is a multiple of the number of processors $z$, that is, $m = k z$, so each processor will process $k$ small blocks, the total computational complexity formula of the new algorithm is as follows:

$$U = \sum_{i=1}^{m} \left( L(n_{j}) + 2n_{i}^{2} + 4n_{i} - 1 \right) + R(n)$$

(6)

The parallel execution time of the new algorithm is as follows:

$$T_{b} \approx [k(L(n) + 2n_{i}^{2} + 4n - 1)] + R(n) T_{j} + T_{i}$$

$$= (kS(n) + R(n) T_{j} + T_{i})$$

$$= (k + q)S(n) T_{j} + T_{i}$$

(7)

Where: $T_{j}$ is the data communication time

As for the data communication time, it can be seen from the new algorithm that communication between processors is only needed when calculating the residuals and $(e^{k})^{T} (e^{k})$, and all other solutions $x^{k}$ and all correction quantities $e^{k}$ can be obtained by serial calculation, which is negligible compared with the execution time of the whole calculation part.

4. Examples and analysis

The following nonlinear bilevel mixed integer programming problems are analyzed

**Upper levels**

$$\min_{x} F(x, y) = -x_{1}^{2} - 2x_{2}^{2} - y_{1}^{2} - 2y_{2}^{2}$$

$$2x_{1} - 3x_{2} \geq -10$$

$$st \quad 14x_{1} - 10x_{2} \geq -97.31$$

$$x_{1}, x_{2} \geq 0 \quad and \quad x_{i} \text{is an integer}$$

(8)

**Lower levels**

$$\min_{y} f(x, y) = -2x_{1}^{2} - x_{2}^{2} - 2y_{1}^{2} - y_{2}^{2}$$

$$3x_{1} - 2x_{2} + 10y_{1} - 10y_{2} \geq -80$$

$$st \quad 10x_{1} - 6x_{2} - 18y_{1} - 10y_{2} \geq -120$$

$$y_{1}, y_{2} \geq 0 \quad and \quad y_{i} \text{is an integer}$$

(9)

In which: $x = (x_{1}, x_{2})^{T}$; $y = (y_{1}, y_{2})^{T}$.

Take the upper initial feasible solution $x^{0} = (0,1), N = 25, P_{c} = 0.5, P_{m_{1}} = 0.02, P_{m_{2}} = 0.2, c = 0.5$ in mutation operator design, the number of chaotic initialization points $n = 300$, and the iteration times $K_{c} = 350$ of combination method. The iteration times $k = 150, \mu = 8$ of chaotic local search and the optimization cycle times of upper and lower layers are taken as $\alpha = 0.2, \beta = 3, \theta = 50$.

Programming in Matlab6.5, the global optimal solution and the approximate value can be obtained by
optimizing the upper and lower layers for 4 times. Using hybrid genetic algorithm, genetic algorithm and quasi-Newton method to solve this problem, the corresponding optimal solutions and values can be obtained, as shown in Table 2.

Table 2 Comparison of other hybrid genetic algorithms

| Algorithm              | Upper layer problem | Lower level problem | Objective function value |
|------------------------|---------------------|---------------------|--------------------------|
| Hybrid genetic algorithm | 7.00218             | 5.81027             | -489.30119               |
| Genetic algorithm       | 6.02071             | 5.50228             | -450.0275                |
| Quasi-Newton algorithm  | 6.15066             | 5.53016             | -4566.9301               |

It can be seen from Table 2 that the hybrid genetic algorithm designed in this paper has better global optimal searching ability than genetic algorithm and quasi-Newton algorithm.

5. Conclusion
Combining the characteristics of genetic algorithm and classical algorithm, this paper proposes a hybrid genetic algorithm for solving nonlinear equations, and establishes a hybrid genetic algorithm for training process neural networks. The hybrid genetic algorithm integrates optimization and iteration, which can give full play to the global convergence and group search ability of genetic algorithm and the advantages of strong local convergence and high precision of classical algorithm (quasi-Newton method), and get satisfactory results. The new algorithm is a good algorithm which can perform parallel computing and has low storage requirements, and can be used for large-scale high-performance computing.

References
[1] Wang Hemin, Gu Miao, Wei Zhiqian, et al. Research and Implementation of Job Shop Problem Based on Hybrid Genetic Algorithm. New Technology and New Technology, no. 8, pp. 69-73, 2017.
[2] Zhang Yuzhou, Xu tingzheng, Zheng junshuai. emergency logistics problem with uncertain emergency degree based on hybrid genetic algorithm. systems science and mathematics, vol. 40, no. 04, pp. 152-166, 2020.
[3] Wang Wei, Liu Chun, Li Dongsheng. Helicopter forward/backward trim based on hybrid genetic algorithm. Journal of BeiHang University, vol. 43, no. 010, pp. 1994-2002, 2017.
[4] Pang Bo. Parallel mutation genetic algorithm based on uniform design. Computer and Digital Engineering, no. 1, pp. 39-45, 2020.
[5] Pan Yuling, Luo Yabo. Process route optimization of complex box parts based on hybrid genetic algorithm. Modular Machine Tool and Automatic Machining Technology, vol. 000, no. 002, pp. 103-107, 2019.
[6] Qu Zhijian, Chen Yuhang, Li Panjing, et al. Adaptive parallel quantum genetic algorithm based on multi-operator coevolution. Acta Electronica Sinica, vol. 47, no. 2, pp. 266-273, 2019.
[7] Gong W, Wang Y, Cai Z, et al. A Weighted Biobjective Transformation Technique for Locating Multiple Optimal Solutions of Nonlinear Equation Systems. IEEE Transactions on Evolutionary Computation, 2017:697-713.
[8] Zheng Shixiong, Zhu Jinbo, Tang Yu, et al. Two-dimensional coupled flutter method based on genetic hybrid algorithm. Journal of Southwest Jiaotong University, no. 1, pp. 64-71, 2018.
[9] Kelley, C. T. Numerical methods for nonlinear equations. Acta Numerica, 2018, 27:207-287.
[10] Zhao Hongchen, Wang Gang, Liu Xiaoming. Application of Multi-population Genetic Algorithm and Truncated Singular Value Decomposition in Switching Arc Inversion. Electrical Manufacturing, vol. 013, no. 006, pp. 14-19,48, 2018.
[11] Bi W, Ren H, Wu Q. Three-step iterative methods with eighth-order convergence for solving nonlinear equations. Journal of Computational & Applied Mathematics, vol. 225, no. 1, pp. 105-112, 2017.