1. Introduction

An important area in current research is the development and application of search techniques based upon the principles of natural evolution. Evolution can be viewed as a change in the genetic composition of a population of individuals over time. In a simplified form, evolution is result of the successive processes of reproduction and genetic variation followed by natural selection, which allows the fittest individuals to survive and reproduce, thus propagating their genetic material to future generations. We shall use this as a starting point in introducing evolutionary computation. The theory of natural selection proposes that the plants and animals that exist today are the result of millions of years of adaptation to the demands of the environment. At any given time, a number of different organisms may co-exist and compete for the same resources in an ecosystem. The organisms that are most capable of acquiring resources and successfully procreating are the ones whose descendants will tend to be numerous in the future. Organisms that are less capable, for whatever reason, will tend to have few or no descendants in the future. The former are said to be more fit than the latter, and the distinguishing characteristics that caused the former to be more fitness are said to be selected for over the characteristics of the latter. Over time, the entire population of the ecosystem is said to evolve to contain organisms that, on average, are more fit than those of previous generations of the population because they exhibit more of those characteristics that tend to promote survival.

Evolutionary computation techniques abstract these evolutionary principles into algorithms that may be used to search for optimal solutions to a problem. In a search algorithm, a number of possible solutions to a problem are available and the task is to find the best solution possible in a fixed amount of time. For a search space with only a small number of possible solutions, all the solutions can be examined in a reasonable amount of time and the optimal one found. This exhaustive search, however, quickly becomes impractical as the search space grows in size. Traditional search algorithms randomly sample or heuristically sample the search space one solution at a time in the hopes of finding the optimal solution. The key aspect distinguishing an evolutionary search algorithm from such traditional algorithms is that it is population-based. Through the adaptation of successive generations of a large number of individuals, an evolutionary algorithm performs an efficient directed search. Evolutionary search is generally better than random search and is not susceptible to the hill-climbing behaviors of gradient based search [1-3].

By mimicking the process of natural evolution, researchers developed the evolutionary algorithms (EA), which are based on the collective adaptability within a population of individuals,
each of which represents a search point in the space of potential solutions to a given problem. In order to an evolutionary algorithms to work, a population of candidate solution is initialized, and it evolves towards increasingly better regions of the search space by means of selection, reproduction and genetic variation mechanisms. The environment in which the population evolves is defined by the aim of the search, and delivers the information, termed fitness, that quantifies how good an individual is. The selection process favors the reproduction of individuals of higher fitness, and a recombination mechanism allows the mixing of parental information while passing it to their descendants. Finally, mutation introduces novelties in the population.

The most popular evolutionary model used in the current research is Genetic Algorithms (GA), originally developed by John Holland [4]. The GA reproduction operators, such as recombination and mutation, are considered analogous to the biological process of mutation and crossover respectively in population genetics. The recombination operator is traditionally used as the primary search operator in GA while the mutation operator is considered to be a background operator, which is applied with a small probability.

Traditionally, GA uses a binary string representation of chromosomes with concentration on the notion of 'schemata'. A schema is a template that allows exploring the similarity among chromosomes. Genetic Algorithms model evolution as a search for structures or building blocks that perform well in a given environment. Therefore, the recombination and mutation operators focus on an individual's structure, not the structure's interpretation. The results of applying reproduction operation in GA generate solutions that share structural similarities with their parents but may have significantly different interpretations. However, many recent applications of GA have used other representation such as graphs, Lisp expressions, ordered list, and red-valued vectors.

The above code gives the basic algorithmic steps for GA. After the initial population of individuals is generated (usually randomly) and individuals' structures are evaluated, the loop is entered. Then a selection buffer C(t) is created to accommodate the selected copies from P(t-1), "select-reproduction". In the Holland original GA, individuals are selected probabilistically by assigning each individual a probability proportional to its structural fitness. Thus, better individuals are given more opportunity to produce offspring. Next the variation operators (mutation and crossover) are applied to the individuals in C(t) buffer producing offspring C(t). After evaluating the structural fitness of C(t), the selection method is applied to select replacement for P(t) from C'(t) and P(t+1).

In general, genetic algorithms are usually used to solve problems with little or no domain knowledge, NP-complete problems, and problems for which near optimum solution is sufficient. The GA methods can be applied only if there exist a reasonable time and space for evolution to take place.

2. Improved Genetic Algorithm

In our algorithm, the code representation we use the real coding, the real coding method has advantages compared with the binary coding in the function optimization problems, because the real coding can solve the "Hamming cliff" problem which the binary coding has no idea to solve it, and then the encoding, the decoding and the calculation of the fitness function is more convenient when we use the real coding.

2.1. Initialize Population

The traditional method of genetic algorithm is randomly initialized population, that is, generate a series of random numbers in the solution space of the question. Design the new
algorithm, we using the orthogonal initialization [5] in the initialization phase. For the general condition, before seeking out the optimal solution the location of the global optimal solution is impossible to know, for some high-dimensional and multi-mode functions to optimize, the function itself has a lot of poles, and the global optimum location of the function is unknown. If the initial population of chromosomes can be evenly distributed in the feasible solution space, the algorithm can evenly search in the solution space for the global optimum. Orthogonal initialization is to use the orthogonal table has the dispersion and uniformity comparable; the individual will be initialized uniformly dispersed into the search space, so the orthogonal design method can be used to generate uniformly distributed initial population.

2.2. Intergenerational Elite Mechanism

Genetic algorithm is usually complete the selection operation based on the individual's fitness value, in the mechanism of intergenerational elite, the population of the front generation mixed with the new population which generate through crossover and mutation operations, in the mixed population select the optimum individuals according to a certain probability. The specific procedure is as follows:

Step1: using crossover and mutation operations for population P1 which size is N then generating the next generation of sub-populations P2;
Step2: The current population P1 and the next generation of sub-populations P2 mixed together form a temporary population;
Step3: Temporary population according to fitness values in descending order, to retain the best N individuals to form new populations P1.

The characteristic of this mechanism is mainly in the following aspects. First is robust, because of using this selection strategy, even when the crossover and mutation operations to produce more inferior individuals, as the results of the majority of individual residues of the original population, does not cause lower the fitness value of the individual. The second is in genetic diversity maintaining, the operation of large populations, you can better maintain the genetic diversity of the population evolution process. Third is in the sorting method, it is good to overcome proportional to adapt to the calculation of scale.

2.3. Adaptive Local Search Operator

Local search operator has a strong local search ability, and then can solve the shortcomings of genetic algorithm has the weak ability for the local search. And the population according to the current state of adaptive evolution of the local search space adaptive local search operator will undoubtedly greatly enhance the ability of local search. In the initial stage of the evolution, the current optimal solution from the global optimum region is still relatively far away, this time the adaptive local search operator to require search a large neighborhood space to find more optimal solution, it can maintain the population diversity. When the population has evolved to the region containing the global optimum, the adaptive local search operator to require a relatively small area to search in order to improve the accuracy of the global optimal solution.

In our algorithm, the adaptive local search operator is the adaptive orthogonal local search operator. Adaptive orthogonal local search operator is aimed at the neighborhood of a point to search, so the key point is to identify a point as the center of the hypercube, the hypercube in the orthogonal test, expect to be better solution.

2.4. Framework of algorithm

In this paper, we design the improved genetic algorithm with the real encoding, orthogonal initialization, combined with the roulette selection and the adaptive local search operator, using the intergenerational elitist mechanism. The framework of the algorithm as follows:

- Step1: Initialize a population P using the orthogonal initialization method, the scale is N, and evaluate the individuals in this population;
- Step2: Using the roulette selection method, select two parent individuals from the population P respectively then using the crossover operation to get an optimal individual, then using the intergenerational elitist mechanism for this individual generate a sub-individual, and bring this individual into the new population pool P', repeat the step2 until the new population pool P' size is N;

An Improved Genetic Algorithm and its Application (Xuesong Yan)
Step3: Evaluate the new population $P'$, then combined the new population $P'$ with the populations $P$ and form a temporary population;
Step4: Sort the temporary population according to the fitness value of the individuals in descending order; retain the best $N$ individuals to form the next generation population $P$;
Step5: If not satisfy the termination condition, switch to step2, otherwise, end of the algorithm.

3. Experiment and Results
In order to verify the validity of the new algorithm, we use benchmarks function to verify the effectiveness of our improvement. We use our algorithm to optimization the following two functions. For each of the two functions our algorithm is run 50 times and our algorithm can get the global optimal solution.

F1: Schaffer function
\[
\min f(x_i) = 0.5 - \frac{(\sin^2 \sqrt{x_i^2 + x_i^2} - 0.5)}{[1 + 0.001(x_i^2 + x_i^2)]^2}, -100 \leq x_i \leq 100
\]

In this function the biggest point is that the overall situation (0,0), the largest in the overall points for the center, to 3.14 for the radius of a circle on the overall situation from numerous major points of the uplift, and, This function has a strong shock, therefore, it is difficult to find a general method of its global optimal solution.

F2: Shubert function
\[
\min f(x_i, y_i) = \left(\sum_{i=1}^{n} \cos[(i+1) x_i + i]\right) \times \left(\sum_{i=1}^{n} \cos[(i+1) y_i + i]\right), x_i, y_i \in [-10, 10]
\]

This function has 760 local minimum and 18 global minimum, the global minimum value is -186.7309.

In order to verify the improvement of the new algorithm, we also use other benchmarks function to test the algorithm’s performance and compare the results with traditional genetic algorithm. Specific details of the test function see table 1. In the table 1, $n$ behalf of the dimension number of the function, $S$ behalf of the range of variables, $f_{\text{min}}$ behalf of the minimization of the function.

The two algorithms of the same experimental parameters set. Each function in Table 1 is run 50 times with the two algorithms, their experimental results such as Table 2. By analyzing the experimental results we know, in solving function $f_1$, $f_4$ and $f_7$, use the genetic algorithm is easily into local optimum, but use the new algorithm, convergence soon, and find a better
solution, the average fitness and the best fitness was both superior to genetic algorithm. For the function f2, the new algorithm and GA all can find the global optimal, these two algorithm for this test function is very effective. For function f5, the two algorithms can find the best solutions are the same (see Table 2), and the new algorithm to get the best value of the average is better than GA algorithm. In sum, we can see that in solving function f1, f4, f5 and f7, the new algorithm more efficient than GA algorithm, in solving other function, the performance almost same of the two algorithms. In short, this new algorithm has the following value: better global search capability.

Table 1. Test function

<table>
<thead>
<tr>
<th>Test Function</th>
<th>n</th>
<th>S</th>
<th>f_{min}</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1 (x) = \sum_{i=1}^{n} x_i^2</td>
<td>30</td>
<td>(-100,100)</td>
<td>0</td>
</tr>
<tr>
<td>f_2 (x) = 6 \cdot \sum_{i=1}^{5}</td>
<td>30</td>
<td>(-5.12, 5.12)</td>
<td>0</td>
</tr>
<tr>
<td>f_3(x) = \sum_{i=1}^{n} i \cdot x_i^4 + U(0,1)</td>
<td>30</td>
<td>(-1.28,1.28)</td>
<td>0</td>
</tr>
<tr>
<td>f_4(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2</td>
<td>30</td>
<td>(-300.0,300.0)</td>
<td>0</td>
</tr>
<tr>
<td>+ \prod_{i=1}^{n} \cos\left(\frac{x_i - 100}{\sqrt{n}}\right) + 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_5(x) = -20 \cdot \exp\left(-0.2 \cdot \sum_{i=1}^{n} x_i^2\right) - \exp\left(-\frac{1}{n} \sum_{i=1}^{n} \cos(2 \pi x_i)\right) + 20 + e</td>
<td>30</td>
<td>(-32,0,32)</td>
<td>0</td>
</tr>
<tr>
<td>f_6(x) = \sum_{i=1}^{n} 100((x_{i+1} - x_i)^2 + (x_i - 1)^2)</td>
<td>30</td>
<td>(-2.048,2.048)</td>
<td>0</td>
</tr>
<tr>
<td>f_7(x) = \sum_{i=1}^{n} x_i \sin(\sqrt{</td>
<td>x_i</td>
<td>})</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 2. New algorithm and GA experiment

<table>
<thead>
<tr>
<th>Function</th>
<th>Algorithm</th>
<th>Best Value</th>
<th>Worst Value</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1 (x)</td>
<td>GA</td>
<td>1495.71</td>
<td>7032.89</td>
<td>201.038</td>
</tr>
<tr>
<td></td>
<td>Improved GA</td>
<td>4.13731E-29</td>
<td>1.0882E-24</td>
<td>2.28015E-26</td>
</tr>
<tr>
<td>f_2 (x)</td>
<td>GA</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Improved GA</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_3(x)</td>
<td>GA</td>
<td>0.00177094</td>
<td>0.00833963</td>
<td>0.000210055</td>
</tr>
<tr>
<td></td>
<td>Improved GA</td>
<td>0.00193565</td>
<td>0.0103595</td>
<td>0.000259903</td>
</tr>
<tr>
<td>f_4(x)</td>
<td>GA</td>
<td>72.5069</td>
<td>123.954</td>
<td>1.5228</td>
</tr>
<tr>
<td></td>
<td>Improved GA</td>
<td>2.18559E-12</td>
<td>8.63194E-25</td>
<td>0.00177512</td>
</tr>
<tr>
<td>f_5(x)</td>
<td>GA</td>
<td>-3.19744E-14</td>
<td>4.4299</td>
<td>0.148418</td>
</tr>
<tr>
<td></td>
<td>Improved GA</td>
<td>-3.19744E-14</td>
<td>1.50229</td>
<td>0.0749509</td>
</tr>
<tr>
<td>f_6(x)</td>
<td>GA</td>
<td>1.84889E-28</td>
<td>8.63194E-25</td>
<td>1.83991E-26</td>
</tr>
<tr>
<td></td>
<td>Improved GA</td>
<td>2.55147E-28</td>
<td>1.20401E-23</td>
<td>2.41678E-25</td>
</tr>
<tr>
<td>f_7(x)</td>
<td>GA</td>
<td>-5038.62</td>
<td>-3233.13</td>
<td>54.0123</td>
</tr>
<tr>
<td></td>
<td>Improved GA</td>
<td>-9535.19</td>
<td>-8203.56</td>
<td>45.1661</td>
</tr>
</tbody>
</table>

4. Conclusion

This paper introduces a new algorithm based on the traditional genetic algorithm, for the traditional GA algorithm the new algorithm has done some improvements: By introducing genetic selection strategy, decreased the possibility of being trapped into a local optimum. Compared the traditional genetic algorithm, the new algorithm enlarges the searching space and the complexity is not high. By analyzing the testing results of nine benchmarks functions optimization, we reach the conclusion: in the optimization precision and the optimization speed, the new algorithm is efficiency than the traditional genetic algorithm and the new algorithm is more efficient than traditional genetic algorithm in coping with the function optimization problems.
Acknowledgments
This paper is supported by the Provincial Natural Science Foundation of Hubei (No. 2011CDB334 and 2011CDB346), National Civil Aerospace Pre-research Project of China and the Fundamental Research Founds for National University, China University of Geosciences(Wuhan).

References