International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Multi-Agent Self-Adaptive Multi-Objective Genetic Algorithm

Shi LianShuan[1]*,Wang HuaHui[1]

[1] *Shool of Information Technology Engineering*
Tianjin University of Technology and Education
Tianjin，China
* Corresponding author's Email: Shilianshuan@sina.com

Abstract: The agent technology and genetic algorithms are integrated and is applied to solve multi-objective optimization problem. An agent in this algorithm represents a candidate solution to the multi-objective optimization problem. Agent lives in the grid environment and it possesses own local space called the neighborhood. In the neighborhood, an agent can compete and collaborate with other agents to achieve the purpose of gene exchange and evolution. Agent also possesses some knowledge of the environment and can learn itself while evolving, in order to adapt itself to the environment better and enhance its viability. A new multi-objective genetic algorithm based on Multi-Agent Self-Adaptive Genetic Algorithm(MASAGA) is proposed, in which the evolution parameters are adjusted adaptively in the evolutionary process and a new selection operator is used to select individual. By adjusting the crossover and mutation parameters in the evolutionary process it can improve the accuracy and convergence speed of the algorithm. Several benchmark functions are used to test the performance of the algorithm and the simulation results indicate that the multi-objective evolutionary algorithm based on MASAGA has a better performance. The algorithm can converge to the Pareto solutions quickly, and has a good diversity compared with NSGA-II.

Keywords: Multi-objective optimization; Genetic algorithm; Agent technology; Self-adaptive operator

1. Introduction

Multi-objective optimization problem (MOP) exists widely for many complex engineering optimization problems in practice and these problems usually contain more than one objective, and objectives are not unified and conflicting with others. MOP is different from Single-objective optimization problem. Optimizing a particular solution with respect to a single objective can result in unacceptable results with respect to the other objectives. Normally, its solution is not the only one, but a compromised solution set called non-dominated solution set or Pareto optimal solution set, each of which satisfies the objectives at an acceptable level without being dominated by any other solution. The traditional method of solving MOP is the weighting method. Weight values corresponded to each optimization objective are used to transform the MOP to a single objective problem. But the weighting method exists some problems. For instance, it is very sensitive with the Pareto frontier (convex or non convex). Weight is not easy to determine in advance. Only a solution can be got at each running and so on.

Genetic algorithm is a representative of a class of methods based on heuristic random search technique. It was proposed by John H. Holland in early seventies and has found application in a number of practical problems since then. The genetic algorithm may be viewed as an evolutionary process wherein a population of solutions evolve over a sequence of generations. The basic idea of GA comes from the mechanics of natural selection.

Each optimization parameter is coded into a gene by some methods such as a real number or string of bits. The corresponding genes for all parameters form a chromosome, which describes each individual. A chromosome could be an array of real numbers or a binary string component depends on the specific problem. The chromosomes are evaluated by fitness function in each generation. After evaluation, solutions are selected for reproduction based on their fitness. The good individuals are selected for reproduction and the bad individuals are eliminated. The selected solutions then undergo recombination under the action of genetic operators, crossover and mutation. Crossover causes exchange of genetic material between chromosomes and crossed chromosomes can produce ones with better fitness value. It occurs only with some the crossover probability. Mutation is done by modifying a solution with the mutation probability. The purpose of mutation is to obtain new genetic material. After performing genetic operator, a termination condition is checked in order to determine whether loop ends.

Based on population searching, genetic algorithm can search in multi-directions and globally, so that it is very suitable for solving multi-objective optimization problem. The vector evaluation genetic algorithm (VEGA) is the first algorithm which applies GA to solve MOP, and then researchers proposed many different multi-objective evolutionary algorithm, such as NSGA[1], NPGA[2], SPEA[3], and the improved algorithm NSGA-II[4] and SPEA2[5], etc.

Although the underlying principles are simple, the genetic algorithm has proven them as a general robust and powerful search mechanism. Genetic algorithm has been established as one of the most widely used technique for multi-objective optimizations. This is because the parallel search nature of genetic algorithm makes the task of approximating Pareto front of optimal solutions in one optimization run becomes possible.

Agent technologies are a natural extension of current component-based approaches, and have the potential to greatly impact our lives and work. Accordingly, this area is one of the most dynamic and exciting in computer science today. Any physical or virtual entity which can perceive and react to the environment can be called agent. Because it has good autonomy, collaboration and self-organizing and self-learning ability, the agent technology has a strong reliability and high efficiency in solving optimization problems. The literature [6] used multi-agent evolutionary thinking to solve function optimization problem of ultra-high dimension, and the advantage of the agent to solve optimization problems is shown.

Adaptive genetic strategy is an improvement of evolutionary algorithm. By adjusting the genetic parameters in the evolutionary process, it can improve the convergence accuracy and speed of the algorithm. Adapting genetic algorithm does not define its parameter values, which is in most cases left to the user. Those values are known to significantly affect the algorithm's performance; poorly chosen parameters can cause the algorithm which will not produce any relevant solutions at all. Moreover, the optimal parameter configuration is often problem dependent, which can make an inexperienced user's utilization of genetic algorithm very difficult.

In this paper, with agent technology agent can adjust the evolution parameters adaptively to adapt the evolution environment better. The simulation results indicate that the multi-objective evolutionary algorithm based on Multi-Agent Self-Adaptive has a better performance result.

2. Multi-objective Optimization Problem

The Multi-objective Optimization Problem (MOP) or the vector optimization problem can be defined as the problem of finding a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. The optimization problem is to maximize or minimize many objective functions with a set of constraints. Because maximization and minimization problem can be transformed into each other, a minimization problem is described for the multi-objective optimization problem without loss of generality.

The mathematical model of general Multi-objective Optimization Problem (MOP) can be described as follows:

$$\textit{Find the vector} \quad x = (x_1, x_2, \cdots, x_n) \in X$$
$$\textit{Minimize} \quad y = f(x) = (f_1(x), f_2(x), \cdots, f_k(x)) \quad (1)$$
$$\textit{Subject to} \quad g(x) = (g_1(x), g_2(x), \cdots, g_m(x)) \le 0 \quad (2)$$

Wheres $x = (x_1, x_2, \cdots, x_n) \in X$ represents the decision vector, $(f_1(x), f_2(x), \cdots, f_k(x)) \in Y$ represents the objective vector, X is the decision space, Y is the objective space, $g(x) = (g_1(x), g_2(x), \cdots, g_m(x)) \le 0$ represents constraint condition, which determines the feasible range of the decision vector.

$G = \{x \in X | g(x) = (g_1(x), g_2(x), \cdots, g_m(x)) \leq 0\}$ is constraint set.

Multi-objective optimization seeks to optimize the components of a vector-valued cost function. Unlike single objective optimization, the solution to this problem is not a single point but a family of points known as the Pareto-optimal set. The aim of solving multi-objective optimization problem is to get a compromise solution and make multiple objectives optimal in a certain sense.

Definition 1 Pareto solution *If not exist any feasible solution x such that* $f_i(x) \leq f_i(x^*)$ *for all* $i = 1,2,\cdots,m$, *and* $f_j(x) < f_j(x^*)$ *for at least one j, then* x^* *is called Pareto solution.*

In a word, this definition says that x^* is Pareto optimal if there exists no feasible vector of decision variables *x* which would decrease some objectives without causing a simultaneous increase in at least one other objective.

Definition 2 Pareto dominate *With any decision variable a and b, if* $f_i(a) < f(b)$ *for all* $i = 1,2,\cdots,m$, *and* $f_j(a) < f_j(b)$ *for at least one j, then* $f(a)$ *is better than* $f(b)$, *b is dominated by a, namely* $a \succ b$; *if* $f(a)$ *can not be compared to* $f(b)$, *then a is non-dominated b, namely* $a \sim b$.

The set of the Pareto optimal solutions is called the Pareto optimal set. The vectors *x* corresponding to the solutions included in the Pareto optimal set are called non-dominated. The plot of the objective functions whose non-dominated vectors are in the Pareto optimal set is called the Pareto front.

Definition 3 Pareto Optimal Set *For a given MOP f(x), the Pareto optimal set P is defined as:* $P^* = \{x^* \in X \mid$ *If not exist any feasible solution x such that* $f(x) \leq f(x^*)\}$.

Definition 4 Pareto Front *For a given MOP f(x), the Pareto front is defined as:*

$PF = \{y = (f_1(x), f_2(x), \cdots, f_k(x)) \mid x \in P^*\}$.

Although the solution of MOP consists of a set of solutions, from a practical point the user needs only one solution. How to select this one optimal solution needs some other preference factor of the objectives function or some other higher-level professional information in advance.

The convergence to the Pareto optimal set and the maintenance of the diversity of the current population on it becomes more difficult as the number of variables increases.

3. Multi-objective Genetic Algorithm based on Agent Self-adaptive

Genetic algorithm is particularly suitable to solve multi-objective optimization problems because they deal simultaneously with a set of possible solutions. A lot of Pareto optimal solutions can be obtained by a single simulation run of genetic algorithm instead of having to perform a series of separate runs in the case of the traditional mathematical programming techniques. Additionally, genetic algorithm does not need derivative and continuity of the objective functions, so that it is easy to apply to practice problem.

Multi-objective genetic algorithm is to use genetic algorithm to solve the multi-objective optimization problem. The idea is that as the evolutionary process individual in the population is converging to the Pareto optimal frontier gradually. The multi-objective genetic algorithm (MOGA) was first introduced by Fonseca and Fleming (1993). The idea is to assign to each individual a rank based on the number of individuals that dominate it plus one. According to this scheme, non-dominated individuals are assigned a rank. The fitness value of each individual is computed according to its rank values. Then GA is used to solve.

Because of the intelligent features of evolutionary algorithm, this paper forms all individuals in the population into agents. All agents live in the grid environment [7] whose size is n*n, and each agent occupies one of the grid nodes.

Agent lives in the grid environment and it possesses own space called the neighborhood. How to define the neighborhood of agent is a very important issue. In some agent evolutionary systems, the idea of position adjacent is often used. So in this paper eight individuals around an agent are taken as its neighborhood. In neighborhood, an agent can compete and collaborate with other agents to achieve the purpose of gene exchange and evolve. Agent also possesses some knowledge of the environment and can learn itself while evolving, in order to adapt itself to the environment better and enhance its viability.

Define 5 Agent energy *Each agent possesses a certain energy, and their survival status is determined according to the energy. Energy for every agent is taken in reverse of the average of agent's objective functions. The formula as follows:*

$$Energy(x) = -(f_1(x) + \cdots + f_k(x))/k \quad (3)$$

Due to genetic algorithm inherent parallelism, the algorithm has the potential of finding multiple Pareto-optimal solutions in a single simulation run. However with many complex applications it is not possible to generate the Pareto optimal solutions, much less the entire Pareto optimal set. Therefore the optimization goal for an MOP may be reformulated in the more general fashion based on three objectives: the distance of the resulting non-dominated front to the Pareto-optimal front should be minimized; A good distribution of the solutions found is desirable; The spread of the obtained non-dominated front should be maximized, i.e., for each objective a wide range of values should be covered by the non-dominated solutions [12].

3.1 Algorithm idea

Due to the unpredictability of the evolutionary process in the multi-objective evolutionary algorithm there is no a suitable crossover and mutation probability to keep the optimum evolutionary state of algorithm. Therefores combined with the intelligence of the agent, it can change the crossover and mutation rate adaptively to adapt the evolution. The algorithm uses Gaussian distribution to generate the initial population, and then label those who are non-dominated. All dominated solutions are removed from the population and then the remaining non-dominated solutions are retained for reproduction. In selection, a new method is used. Three parents are selected at random. A child is generated from the three parents. If the child dominates the optimal parent, then it is placed into the population. Otherwise, a new selection process takes place. This process continues until the population meets the requirements.

3.2 Selection operator

In order to obtain limited resources in the environment agents compete against each other. Only the individuals with good adaptability can be selected to survive and others will be eliminated.

The first step is to label the non-dominated individuals by definition 2, namely, according to the definition of the Pareto domination to select non-dominated solutions from the population.

If the number of non-dominated is less than three, a non-dominated solution is obtained by search the dominated population, then the solution is labeled as non-dominated and placed into evolution population.

The process continues until the number of population is not less than three.

Then all dominated solutions that non-labeled are removed from population, and then three agents are selected at random in the remaining solutions, in which the agent with the largest energy is called reference solution defined as a_1, and two others are called support solutions defined as a_2, a_3.

3.3 Crossover operator

The crossover operation reflects the collaborative behavior of agents. The agent who lives in the environment will collaborate with others in the same neighborhood, to improve their own energy.

In order to make the agent adapt to the evolution environment better and improve the ability of evolution, the crossover rate is changed adaptively. The child crossover rate is generated by its parents, the formula as follows:

$$P_c^{child} = P_c^{a_1} + U(0,1) \times (P_c^{a_2} - P_c^{a_3}) \quad (4)$$

If the crossover rate is not in [0, 1], repair the crossover rate in the [0, 1] according to the repair rule.

After got a new crossover rate, generate the child who is $child = (c_1, c_2, \cdots, c_n)$ according to the formula as follows:

$$c_i = \begin{cases} a_i^1 + P_c^{child} \times (a_i^2 - a_i^3), U(0,1) < P_c^{child} \\ a_i^1, otherwise \end{cases} \quad (5)$$
$$i = (1,2,\cdots n)$$

Where, $U(0,1)$ is a random number between [0, 1].

3.4 Mutation operator

Agent possesses some environment knowledge, and it can make use of the knowledge to learn to improve energy itself. If the child generated by crossover operator dominates the reference parents it is mutated. The mutation rate is also changed adaptively by the formula as follows:

$$P_m^{child} = P_m^{a_1} + U(0,0.1) \times (P_m^{a_2} - P_m^{a_3}) \quad (6)$$

If the mutation rate is not in [0, 1], the mutation rate is repaired into the [0, 1] according to the repair rule.

A new $child' = (c_1', c_2', \cdots, c_n')$ is generated by as follows mutation formula:

$$c_i' = \begin{cases} c_i + U(0,0.1) \times range, U(0,1) < P_m^{child} \\ c_i, otherwise \end{cases} \quad (7)$$
$$i = 1,2,\cdots,n$$

Here range is the difference between the maximum and minimum value in which the variable c_i can take. $U(0,0.1)$ is a random number between [0,1]. That means a small random perturbation to variables of child is generated, which is equivalent to find a better solution in the vicinity, and then place the child into the population.

The repaired rule [8] is simply to truncate the constant part of the value when crossover and mutation rate is not in [0, 1]. If greater than 1, the P_c is set to 0.9, P_m is set to 0.1. If less than 0, the P_c is set to 0.5, P_m is set to 0.01.

4. Algorithm Descriptions

The initial population is N, crossover rate is P_c, mutation rate is P_m, the maximum number of generation is 100. The algorithm steps are as follows:

Step 1 *The initial population is produced at random according to a Gaussian distribution N(0.5, 0.15);*

Step 2 *The population is sorted by the Pareto dominated method and then all dominated individuals are removed from the population.*

If the number of non-dominated individuals is less than three, find a non-dominated solution from dominated population and place the solution into evolution population until the number of population is not less than three.

Step 3 *Select three individuals from the population at random and the best as a reference solution and the other two as supporting solution;*

Step 4 *Use adaptive crossover rate to generate the child;*

Step 5 *If the child is non-dominated to the reference, mutate the child adaptively and place the individual into population; otherwise turn to Step 4;*

Step 6 *If the population size is less than N, turn to Step 3; otherwise turn to 7;*

Step 7 *gen = gen + 1, if gen is less than 100, turn to Step 2; otherwise stop evolving and output the Pareto solutions.*

5. Experiment Simulation and Analysis

5.1 Test problems

In order to test the performance of the algorithm six benchmark functions are used, and performances are compared with NSGA-II. Test functions are shown as follows[9]. An important purpose of an algorithm is to achieve a well distributed non-dominated front. However certain characteristics of the Pareto optimal front may prevent an algorithm from finding diverse Pareto-optimal solutions, such as convexity or non-convexity, discreteness, and non-uniformity.

These test functions have complicate features and are difficult to find good the Pareto fronts. All these problems contain two objectives, and the convexities of the feasible regions in objective space are different. The dimensions are also different. So the performance of the algorithm can be tested comprehensively.

ZDT2 contain 30 variables, where the optimal frontier of ZDT2 is non-convex and continuous, ZDT4 contains 10 variables, and the optimal frontier is convex.

1) ZDT2：The non-convex Pareto optimal solution set

$$\min \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(1-(x_1/g)^2) \end{cases}, g = 1 + 9(\sum_{i=2}^{n} x_i/(n-1))$$

$$x_i \in [0,1],$$

$$i = 1,\cdots,30$$

2) ZDT4：The convex Pareto optimal solution set.

The test function ZDT4 contains a lot of local Pareto-optimal sets and therefore tests the algorithm ability to deal with multimodality.

$$\min \begin{cases} f_1(x) = x_1 \\ f_2(x) = g \times (1-\sqrt{x_1/g}) \end{cases}$$

$$g = 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i))$$

$$x_1 \in [0,1]$$

$$x_i \in [-5,5]$$

$$i = 2,\cdots,10$$

5.2 Experiment results and analysis

There are two main purposes of Multi-objective optimization problems: one is to converge to the Pareto solutions fast and the other is to maintain the diversity of the Pareto solutions.

Set the Population size of the algorithm to 100, initial crossover rate 0.8, mutation rate 0.1; the crossover and mutation rate of NSGA-II are set to 0.8, 1/n, where n is the number of variables. All the max generation is 100.

The simulation graph of the test problems of two algorithms as follows:

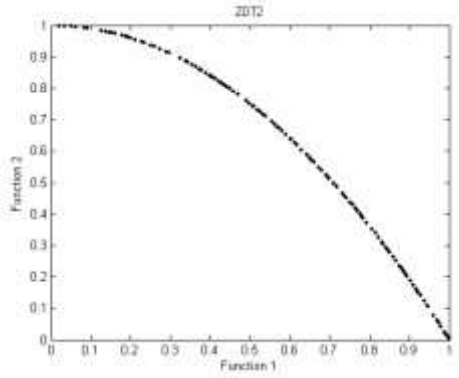

Figure1 Pareto frontier of ZDT2 by MASAGA SAMOGA

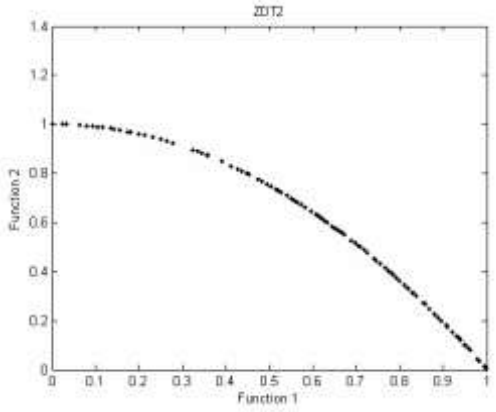

Figure2 Pareto frontier of ZDT2 by NSGA-II

The Pareto frontier of ZDT2, ZDT4 are shown in Figure 1-7.

It can be seen from the above simulation figures that the algorithm is good at solving ZDT2, ZDT4 as well as NSGA-II, which indicates the algorithm has a good solving performance.

Some results of MASAGA for ZDT2 in the running process are given at initial population, generation 20 and generation 100. At initial population the individuals are distributed at random. At generation 20, the solutions begin closing to the Pareto front. At generation 100, the solutions convege to the Pareto front.

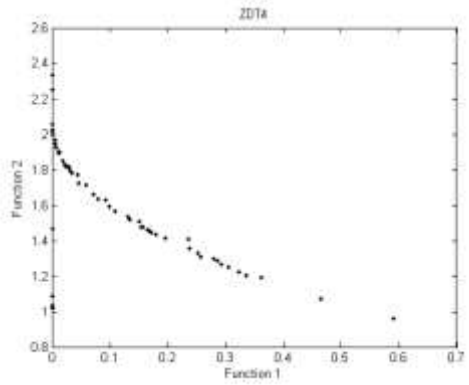

Figure3 Pareto frontier of ZDT4 by NSGA-II SAMOGA

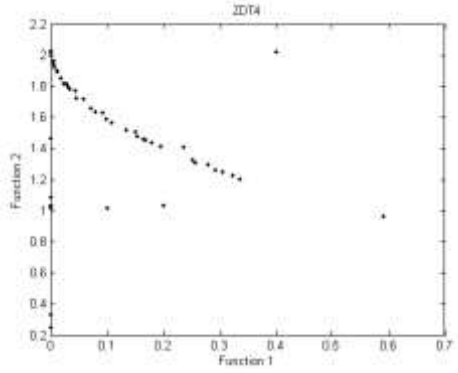

Figure4 Pareto frontier of ZDT4 by SAMOGA

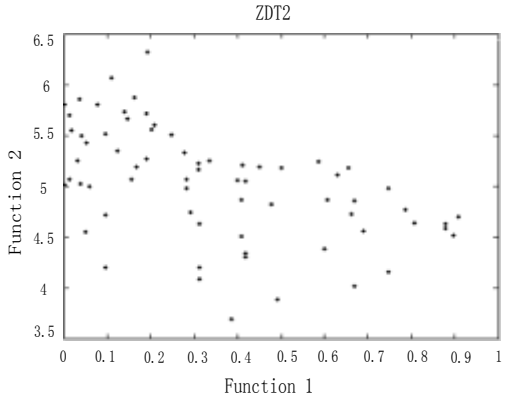

Figure5 Initial population for ZDT2 by MASAGA

6. Conclusions

In this paper, an evolution algorithm based on self-adaptive agent was introduced for multi-objective optimization problems. The approach use the crossover and mutation rates by self-adapts. The six benchmark problems were used to test the algorithm,

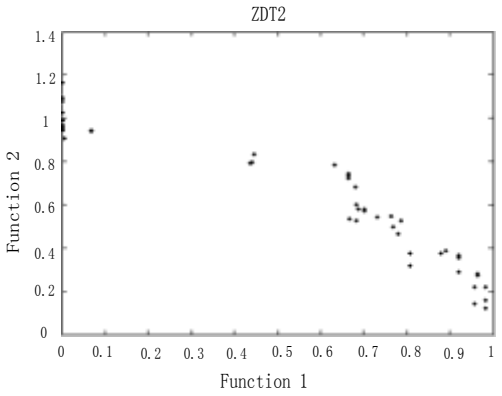

Figure 6 The population at generation 20 for ZDT2 by MASAGA

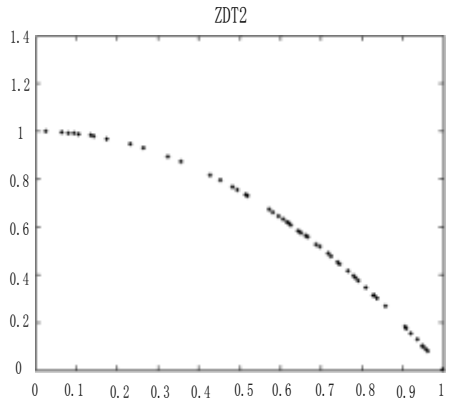

Figure7 The population at the generation 100 for ZDT2 by MASAGA

and the results show it has a good performance. The adaptive techniques can be implemented independently by a more experienced user, which can combined with specific problem dependent knowledge. The human factor remains the most responsible element for successful GA utilization in real-life applications.

7. Acknowledgment

This work was supported by Tianjin Research Program application Foundation and Advanced Technology (14JCYBC15400).

References

[1] Srinivas N, Deb K, Multi-objective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 1994, 2(3):221-248.

[2] Horn J, Nafpliotis N, Goldberg DE. A niched Pareto genetic algorithm for multi-objective optimization. In: *Proc. Of the 1st IEEE Conf. On Evolutionary Computation.* Piscataway: IEEE World Congress on Computational Computation, 1994, 1:82-87.

[3] Zitzler E, Thiele L. Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation*, 1999, 3(9):257-271.

[4] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. On Evolutionary Computation*, 2002, 6(2): 182-197.

[5] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm. In: Giannakoglou K, Tsahalis D, Periaux J. *Proc of the EUROGEN 2001, Evolutionary Methods for Design, Optimization can Control with Applications to Industrial Problems.* Athens, 2002. 95-100.

[6] Zhong WC, Liu J, Xue MZ, Jiao LC. A multi-agent genetic algorithm for global numerical optimization. *IEEE Trans. On Systems, Man and Cybernetics*, 2004, 34(2): 1128-1141.

[7] PanXY, Liu F, Jiao LC. Multi-objective social evolutionary algorithm based on multi-agent. Journal of Software, 2009, 20(7): 1703-1713.

[8] Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. *Proceedings of the 2002 Congress on Evolutionary Computation.* 2002. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC2002), volume 1, pages 831–836, Piscataway, NJ, 2002. IEEE Press.

[9] Lei DM, Wu ZM. Crowding-Measure based multi-objective evolutionary algorithm. *Chinese Journal of Computers*, 2005, 28(8): 1320-1326.

[10] Abdullah Konak,David W. Coit,Alice E. Smith. Multi-objective optimization using genetic algorithms: *A tutorial. Reliability Engineering & System Safety*. 2006,91(Issue 9):992 – 1007

[11] Nattavut Keerativuttitumrong, Nachol Chaiyaratana, and Vara Varavithya. Multi-objective Co-operative Co-evolutionary Genetic Algorithm. *PPSN VII, LNCS 2439*, pp. 288 – 297, 2002.

[12] Eckart Zitzler. Evolutionary Algorithms for Multi-objective Optimization: Methods and Applications[Doctor dissertation]. *Institut für Technische Informatik und Kommunikationsnetze Computer Engineering and Networks Laboratory*. 1999