International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

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

Tauqir Wahab

School of Information Technology and Engineering, Tianjin University of Technology and Education, Tianjin, China

Abstract: ***Now a days multi-objective optimization is one of the biggest problem, to solve such type of problem, the genetic algorithms and an agent technology are integrated and is applied to solve such type of multi-objective optimization problems. In this algorithm an agent represents an applicant answer to the multi-objective optimization problem. An agent lives in the grid environment and it owns confined space called the neighborhood. An agent can compete and cooperate with other agents with other agents, to achieve the purpose of inheritable factor replaced and developed. Agents also retains some knowledge of the environment and can learn by itself while developing, in order to adapt itself to the environment better and enhance its possibility. A new multi-objective genetic algorithm based on Multi-Agent Self-Adaptive Genetic Algorithm (MASAGA) is proposed, in this algorithm evolution parameters are accustomed adaptively in the evolutionary process and a new selection operator is used to select individual. By accustoming the mutation and crossover parameters in the evolutionary process, it can improve the precision and convergence speed of the algorithm. Several benchmark functions are run to test the execution of the algorithm, the simulation results indicate that the multi-objective evolutionary algorithm based on MASAGA has better performance. The algorithm can converge to the Pareto solutions quickly, and some intelligent algorithms are embedded in NSGA-II to improve the algorithm in order to expect better results.***

Keywords: Multi-objective optimization; Genetic algorithm; Agent technology

1. Introduction

The complexity of information systems is increased a lot recent years, leading to increase effort for maintenance and configuration. Self-adaptive systems (SAS) addresses this issue. Due to new computing trends, such as pervasive computing, miniaturization of IT leads to mobile devices with the emerging need for context adaptation. Self-adaptive multi agent system is one on the paradigm now a days. The system is a combination of multi-agent system and genetic paradigm, and guided by a hyper-heuristic dynamically adapted by a collective learning process. In every generation, the population is divided into two parts randomly and one of the parts will be done by the dividing operator which will enhance the diversity of the population and avoid falling into the local optimal.

A genetic algorithm represents a class of methods and these class of methods is based on heuristic random search technique. This technique was proposed by John H. Holland in early 70's and has found application in a number of practical problems since. The genetic algorithm maybe observed as an evolutionary progression wherein a population of solutions progresses over a system of generation. The resultant genes for all parameters form a chromosome, which describes each individual. A chromosome could be an array of real numbers, a binary string component, and these all are dependent on the explicit problem. The chromosomes are estimated by fitness function in each generation. After valuation, different solutions are derived for reproduction based on their fitness. The good entities are selected for reproduction and the bad entities are rejected. The selected results then undergo recombination under the action of genetic operatives, crossover and mutation. Crossover causes give-and-take of genetic material between chromosomes; crossed chromosomes can produce ones with better fitness value. It occurs only with some of the crossover probability. Mutation is done by transforming a result with the mutation probability. The purpose of mutation is to gain new genetic material. After performance of genetic operators, a cessation condition is monitored in order to conclude whether loop ends.

In this paper, I delivered a designed overview of self-adaptation and approaches, which analyze future research directions, and stimulate the need for a new perception on self-adaptation in prevalent computing systems.

A multi-agent system (MAS) can succeed near optimum efficiency, as agents have to rely their result on a subset of the production data, and then it has to find local optimums. Though, finding the best conceivable schedule is also perplexing and time-consuming for an integrated systems, both from a development point of view. In recent years, agent-based computation has been broadly applied in distributed problem solving. An agent is a self-contained problem solving entity which shows the properties of autonomy, social ability, sensitivity, and pro activeness [13]. In a multi agent optimization system (MAOS).

On basis of searching population, a genetic algorithm can search through different ways and globally, and this makes it suitable for resolving multi-objective optimization problems. The first algorithm that applied genetic algorithm to solve MOP was vector evaluation genetic algorithm (VEGA), after saving it researchers started proposing different kinds of multi-objective evolutionary algorithms, such as NSGA[1], NPGA[2] SPEA[3], and then NSGA[1] and SPEA[2] were improved to NSGA-II[4] and SPEA2[5] etc.

In spite of the fact that the basic standards are basic, the genetic algorithm has demonstrated them as a common, vigorous and capable search mechanism. Genetic algorithm has been characterized as world's most used technique for multi-objective optimizations. the task of approximating

Pareto front of optimal solutions in one optimization it becomes possible, and all this is because of parallel search nature of genetic algorithm.

These agent technologies are like natural extensions of current component-based approaches, and it has such potential that our lives and work are impacted by it. Accordingly, in computer science field this area is one of the most exciting and dynamic, it is because any virtual or physical entity that can react to this environment or perceive can be called agent, because it has good autonomy, self-organizing and collaboration. Self-learning ability, the agent technology encompasses a solid unwavering quality and tall proficiency in fathoming optimization problems. In literature [6] the researcher used multi-agent evolutionary idea to solve function optimization problems of high dimensional and researcher has shown problems solved by agent.

There are many improvements in evolutionary algorithm like adaptive genetic strategy. Convergence accuracy and speed of algorithm can be improved in evolutionary process by adjusting the genetic parameters. Adapting genetic algorithm does not define parameter values by itself, in most cases an user has to give these parameters. The purpose of these values are to affect algorithm's performance, significantly. Algorithm cannot produce relevant solutions for all poorly chosen parameters. Moreover, the optimal parameter configuration is often problem dependent.

And this can make difficulties for new users' utilization of genetic algorithm.

In my pare, with help of agent technology, the evolution parameters can be adjusted adapively by agent, to habituate the evolution environment better.

The simulation result shows that the multi-objective evolutionary algorithm based on Multi-Agent Self-Adaptive has a good performance result.

2. Multi-objective Optimization Problem

Multi-objective optimization is fundamental part of optimization activities and has an enormous practical importance, since practically all real-world optimization problems are preferably suited to be modeled using multiple inconsistent objectives. The classical means of fathom such problems were fundamentally focused on scalarizing multiple objectives into a single objective, whereas the evolutionary means to solve a multi-objective optimization problem as it is. The Multi-objective Optimization Problem (MOP) or the vector optimization problem can be described as the complication of finding a vector of conclusion variables which satisfies limitation and optimizes a vector function whose elements represent the unbiased functions. These functions form a mathematical interpretation of production criteria which are usually in dispute with each other. The purpose of optimization problem is to minimize or maximize different objective functions with a set of limitation. Because of maximizing or minimizing a problem can be alter into each other, without loss of extensively, a minimized problem is described for the multi-objective optimization problem.

Mathematically a general Multi-objective Optimization Problem (MOP) can be described as:
Find the vector $x = (x_1, x_2, \cdots, x_n) \in X$

$$\text{Minimize} \quad y = f(x) = (f_1(x), f_2(x), \cdots, f_k(x)) \qquad (1)$$

$$\text{Subject to} \quad g(x) = (g_1(x), g_2(x), \cdots, g_m(x)) \le 0 \qquad (2)$$

Where $x = (x_1, x_2, \cdots, x_n) \in X$ is represented as vector decision, $(f_1(x), f_2(x), \cdots, f_k(x)) \in Y$ is represented as objective vector, Here Y is objective space and X is the decision space, $g(x) = (g_1(x), g_2(x), \cdots, g_m(x)) \le 0$ shows limitation of condition. Which decides the attainable range of the decision vector. $G = \{x \in X | g(x) = (g_1(x), g_2(x), \cdots, g_m(x)) \le 0\}$ Is a constraint set.

Multi-objective optimization attempts to find optimize the components of a vector-valued cost function. In comparison to single-objective optimization. There isn't a single solution to this dilemma. But a group of points known as the Pareto-optimal set. The point of solving multi-objective optimization problem is to get an understanding answer, make multiple objectives optimal in a definite sense.

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

The first idea from above definition shows that, if there is no feasible vector of decision variables, x^* is Pareto optimal.
X will reduce certain targets while simultaneously increasing at least one other objective.

The second idea from above definition shows that, $f_i(a) < f(b)$ for all $i = 1, 2, \cdots, m$ and for at least one j, then $f(a)$ is better than $f(b)$, b is dominated by a, namely $a \succ b$; if $f(a)$ cannot be compared to $f(b)$ then a is non-dominated b, namely $a \sim b$

The Pareto optimal set is the collection of Pareto optimal solutions. Non-dominated vectors are the vectors x that correspond to the solutions in the Pareto optimal range. The Pareto front is a plot of objective functions whose non-dominated vectors are in the Pareto optimal range.

The third definition of Pareto Optimal Set For a given MOP f(x), the Pareto optimal set P can be defined as: P*= {x*∈X|If not exist any feasible solution x such that $f(x) \le f(x^*)$ }.

The fourth definition of Pareto FrontFor a given MOP f(x), the Pareto front can be defined as:

PF= {y= $(f_1(x), f_2(x), \cdots, f_k(x))$ |x∈P*}.

While MOP's solution consists of a number of choices, the user only requires one. Any other preference element of the objectives features or some other higher-level technical knowledge is needed in advance to choose this one optimal solution. As the number of variables increases, achieving the Pareto optimal set and maintaining the variability of the current population on it becomes more difficult.

3. Multi-objective genetic algorithm based on agent self-adaptive

Generally genetic algorithm is particularly fit to find and answer for multi-objective optimization problems, because they give out concurrently with a set of possible solutions. By a single simulation run of a genetic algorithm we can get a lot of Pareto optimal solutions, and because of this we don't have to perform a series of different separate running of algorithms as in traditional mathematical programming techniques cases. Additionally, Since the genetic algorithm does not include the derivative or continuation of the objective functions, it is simple to apply to real-world problems.

To solve the multi-objective optimization problem a Multi-objective genetic algorithm is to use genetic algorithm, the theory is that as the evolutionary process progresses, each entity in the community eventually approaches the Pareto optimal frontier. In 1993 Fonseca and Fleming introduced multi-objective genetic algorithm (MOGA) for the first time. The idea is to assign each entity a rank based on the number of individuals who increment it. Non-dominant entities are given a rating in this system. Each individual's fitness value is calculated based on their rank values. Then, GA is used to solve the problem. Because of the evolutionary algorithm's intelligent characteristics, this paper transforms all individuals of population into an agent. All agents live in the grid environment [7] that the size is n*n, each agent occupies one of the grid nodes. The grid environmental shown in Figure 1.

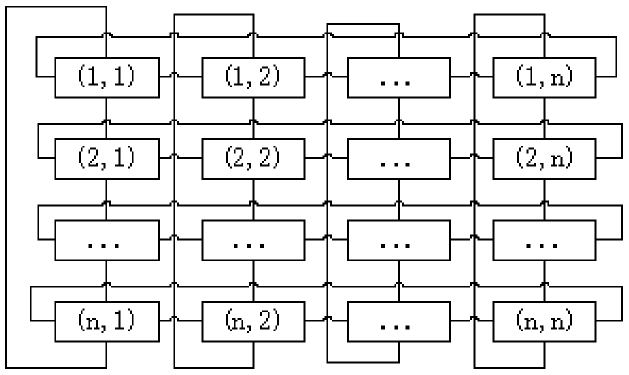

Figure 1: Survival grid of agent

Agents use to lives in the grid environment and it has its own space which is called the neighborhood. It is very important to define agent's neighborhood. The idea of area adjacent is often used, in someagent evolutionary systems, eight entities around an agent are taken as its neighborhood, inthis paper. Inside neighborhood, an agent can collaborate as well as compete with other agents. The reason of all this is to achieve the motive of gene evolved and exchanged. Agent also hold some knowledge of the environment around itself and it can also learn itself while developing, in order to adapt itself to the given environment better and increase its viability.

Definition of 5 agent energy, each agent holds a specific energy, and their survival position is intent on according to the energy. Any agent's energy is determined by reversing the average of the agent's objective functions.

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

Parallel processing is implicit in genetic algorithms. In a single simulation run the algorithm is probable of finding multiple Pareto optimal solutions. Though it's impossible to result in the Pareto-optimal solutions with many complex applications, much lessthe entire Paretooptimal set. Therefore, MOP's optimization goal may be reformulated in vast general fashion on three objectives.

The distance of the resulting non-dominated front to the Pareto-optimal frontshould be minimized: The gap between the non-dominated front and the Pareto-optimal front should be as minimal as possible; It's preferable to have a good distribution of the solutions found; The spread of the non-dominated front that has been obtained should be maximized, i.e., foreach objective a wide range of values should be covered by the non-dominated solutions [6].

4. Algorithm idea

Since the multi-objective evolutionary algorithm's evolutionary mechanism is unpredictable, there is no appropriate crossover and mutation likelihood to preserve the algorithm's optimal evolutionary state. As a result, when combined with the agent's intellect, to adapt the evolution, alter the crossover and mutation rates adaptively. The algorithm produces the initial population using a Gaussian distribution and then labels those who are non-dominated. The population is cleansed of all dominated solutions, and the non-dominated solutions that remain are then held for reproduction. A new approach is used in the selection process. At random, three parents are chosen. From the three parents, a child is born. If the child has the dominant position over the ideal parent, after that, it's released into the population. If this is not the case, then a new selection process is initiated. This method is repeated until the population satisfies the criteria.

4.1 Crossover Operator

Agent cooperation is reflected in the crossover process. In order to boost their own resources, the agent who lives in the area will cooperate with others in the same community. The crossover rate is modified adaptively in order to improve the agent's ability to adapt to the evolution environment and improve the agent's ability to evolve. The offspring crossover rate is determined by the parents of the child. he formula as follows:

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

use the repair rule to bring it into the range [0, 1], if incase the crossover rate is not in the range [0, 1].

After you've obtained a new crossover pace, create a child $child = (c_1, c_2, \cdots, c_n)$ using the following formula:

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

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

4.2 Selection Operator

Agents compete against one another in order to acquire scarce resources in the environment. Only those with high adaptability will be chosen to survive; the rest will be dismissed.

The first step is to classify non dominated individuals using definition 2, which is to pick non-dominated solutions from the population using the definition of Pareto domination.

If the number of non-dominated solutions are less than three, the dominated population is searched for a non-dominated solution, which is then classified as non-dominated and put into the evolution population.

The method continues until there are at least three entries in the community.Then all non-labeled dominated solutions are eliminated from the population, and three agents are chosen at random from the remaining solutions, with the agent with the highest energy being designated as the reference solution, and the other two being designated as support solutions, defined as a_2, a_3.

4.3 Mutation operator

The agent has some environmental awareness, and it can use that knowledge to learn how to improve its own resources. It is mutated if the child formed by the crossover operator dominates the allusion to parents. The following formula is used to adjust the mutation rate adaptively:

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

The mutation pace is repaired into the [0,1] if in case the mutation rate is not in [0,1] according to the repair rule.

A new $child' = (c_1', c_2', \cdots, c_n')$ is generated with the help of following mutation formula:

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

The variable's range is defined as the difference between its maximum and minimum value c_i can take. $U(0, 0.1)$ is a number that falls between [0,1]. That is, a small random perturbation to the child's variables is created, which is equivalent in finding a better solution nearby, and then the child is placed into the population.

When crossover and mutation rates are not in [0, 1], the repaired rule [7] simply truncates the constant part of the value. If less than 0, the P_c is set to 0.5, P_m is set to 0.01. and if greater than 1 then P_c is set to 0.9, and P_m is set to 0.1.

5. Algorithm Descriptions

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

Step 1: A Gaussian distribution N (0.5, 0.15) is used to create the initial population.
Step 2: The Pareto dominated approach is used to sort the population, and then all dominated individuals are excluded. If non dominated individual's number is less than three, then Find a non-dominated alternative from the dominated population and put it in the evolution population before the overall population reaches three.
Step 3: Choose three individuals at random from the population, with the better functioning as a reference solution and the other two as supporting solutions.
Step 4: To build the child, use an adaptive crossover rate.
Step 5: Incase if the child is non dominated according to the reference, then the child is mutated adaptively and population is filled with individuals; otherwise, repeat step 4.
Step 6: after doing the above step if the population size came up with less than N, so then we will jump to step3; otherwise start step 7.
Step 7: gen = gen + 1, turn to Step 2 if gen is less than 100, otherwise avoid evolution and produce the Pareto solutions.

6. Experiment Simulation and Analysis

6.1 Test Problems

In sequence of testing the performance of the algorithm, I have used three benchmark functions, and then the performances are compared with NSGA-II. An algorithm's main goal is to produce a well-distributed, non-dominated front. However, certain features of the Pareto optimal front can make it difficult for an algorithm to find multiple Pareto-optimal solutions. Such as, such as, discreteness, and non-uniformity, convexity or non-convexity, the test functions have complicated features that make finding good Pareto fronts difficult. The problem has two purposes, and the convexities of the feasible regions in objective space vary, as do the dimensions. So, we will test the problem comprehensively.

SCH：Theconvex Pareto optimal solution set

$$\min \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$$
$$x_i \in [-10^3, 10^3]$$
$$i = 1, \cdots, n$$

ZDT1：Theconvex Pareto optimal solution set

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

$x_i \in [0,1]$,

$i=1,\cdots,30$

ZDT2: The convex and non-continuousPareto optimal solution set.

ZDT3 shows the discreteness features: its Pareto-optimal frontconsists of several non-contiguous convex parts.

$$\min\begin{cases} f_1(x)=x_1 \\ f_2(x)=g[1-\sqrt{x_1/g}-x_1\sin(10\pi x_1)/g] \end{cases}$$

$$g=1+9(\sum_{i=2}^{n} x_i/(n-1))$$

$x_i \in [0,1]$,

$i=1,\cdots,30$

The sine function in $f_2(x)$ causes incoherence in the Paretooptimalfront. However, there is no incoherence in the objective space.

ZDT3: The convex Pareto optimal solution set.

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

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

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

$x_1 \in [0,1]$

$x_i \in [-5,5]$

$i=2,\cdots,10$

ZDT4: The non-convex Pareto optimal solution set

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

$x_i \in [0,1]$,

$i=1,\cdots,30$

ZDT6 : The Non-convex and non-uniform Pareto optimal solution set

The test function ZDT6 embraces 2 types of problems caused by non-uniformity of the object space: The density of the solution is highest away from the front and least near the Pareto-optimal front and secondly, the Pareto-optimal solutions are non-uniformly distributed along the global Pareto front.

$$\min\begin{cases} f_1(x)=1-\exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(x)=g\times(1-(f_1/g)^2) \end{cases}$$

$$g=1+9\times(\sum_{i=2}^{n} x_i/(n-1))^{0.25}$$

$x_i \in [0,1]$,

$i=1,\cdots,10$

7. Experiment Results and Analysis

There are two main purposes of multi-objective optimization problems. First one is to converge to the Pareto solutions fast; and second 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 are as follows:

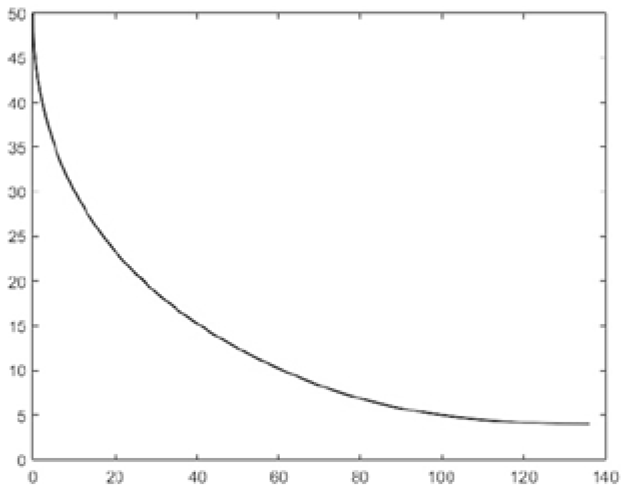

Figure 2: Pareto frontier of SCH byMASAGA

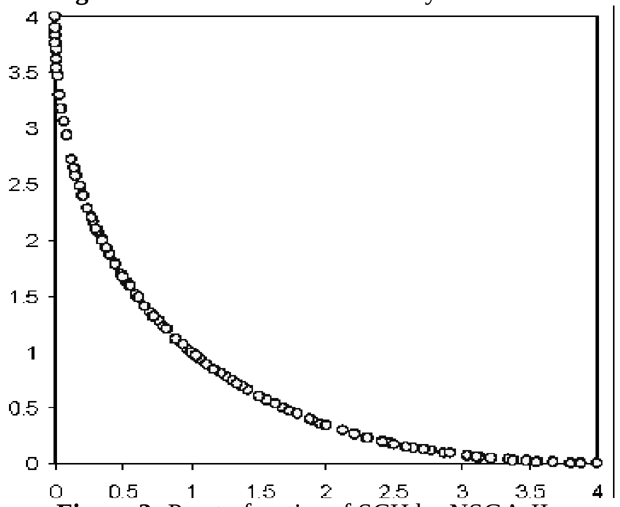

Figure 3: Pareto frontier of SCH by NSGA-II

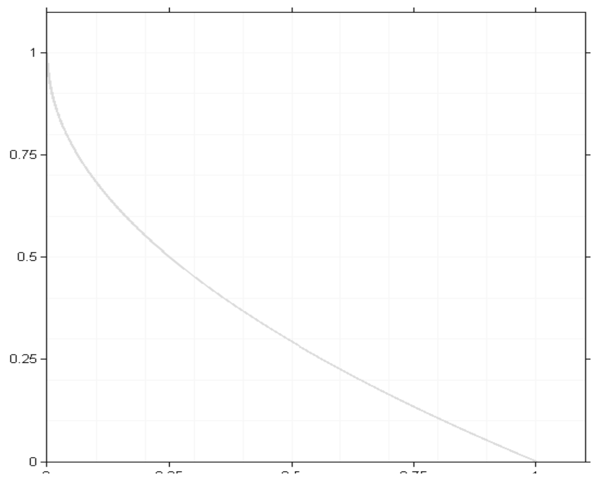

Figure 4: Pareto frontier of ZDT1 by MASAGA MOGA

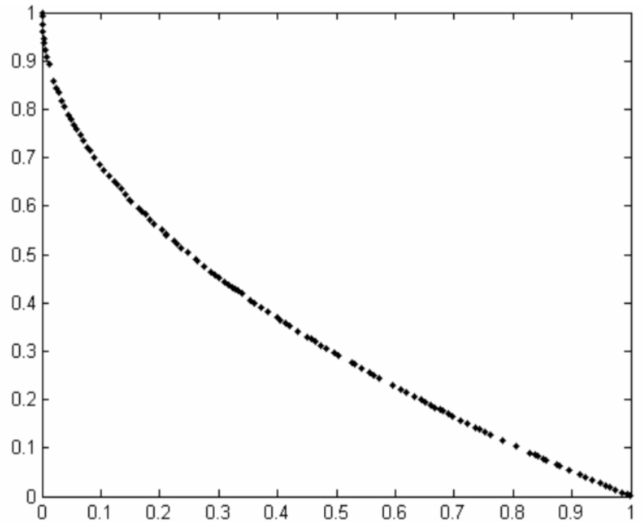
Figure 5: Pareto frontier of ZDT1 by NSGA-II

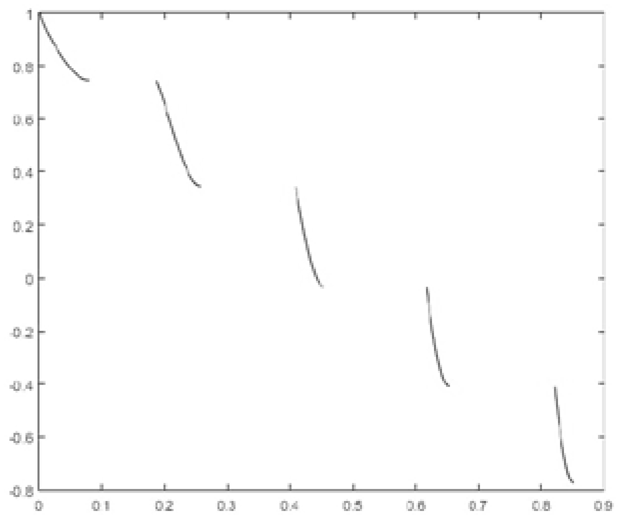
Figure 8: Pareto frontier of ZDT3 by SAMOGA

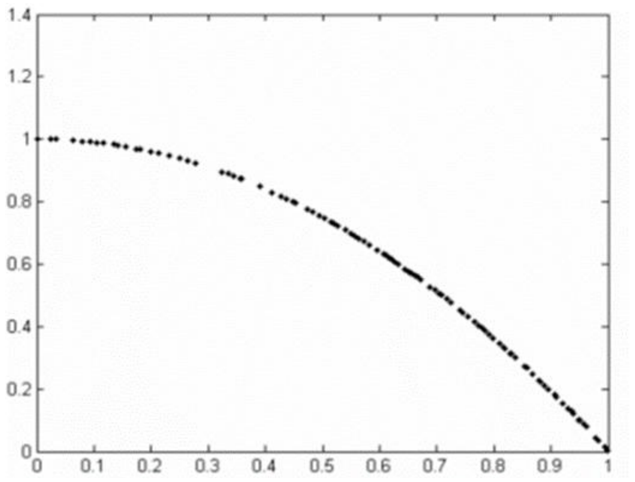
Figure 6: Pareto frontier of ZDT2 by NSGA-II

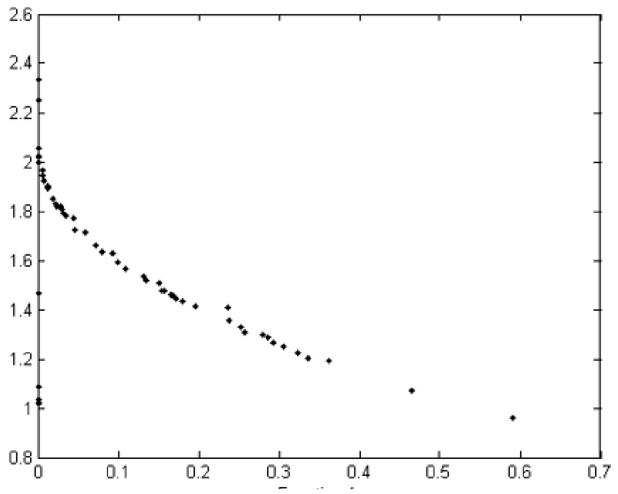
Figure 9: Pareto frontier of ZDT4 by NSGA-II

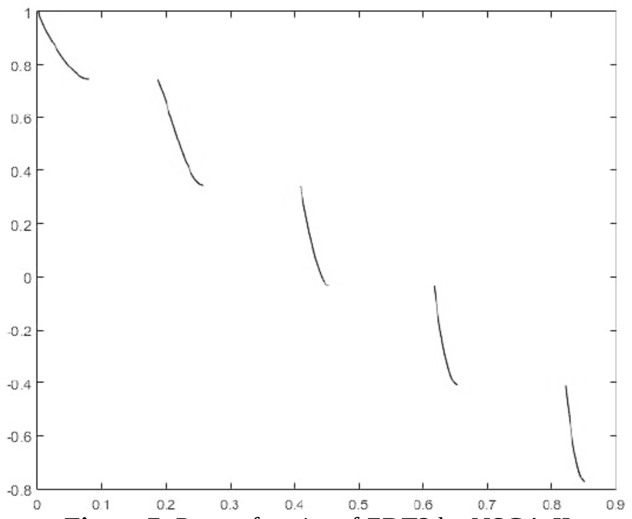
Figure 7: Pareto frontier of ZDT3 by NSGA-II

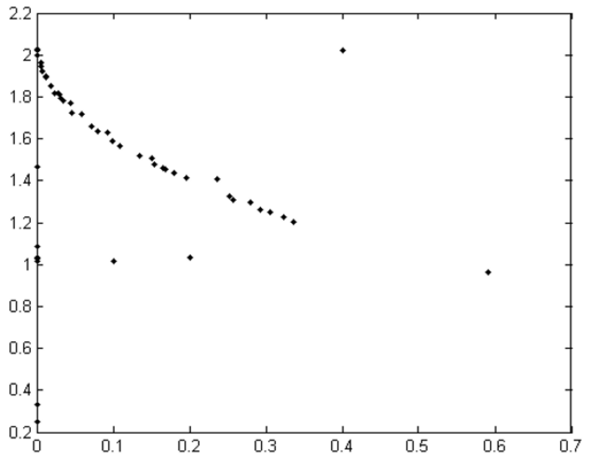
Figure 10: Pareto frontier of ZDT4 by SAMOGA

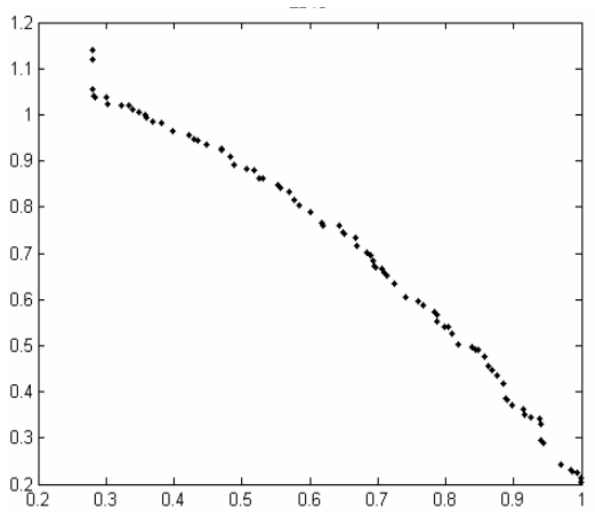

Figure 11: Pareto frontier of ZDT6 by NSGA-II

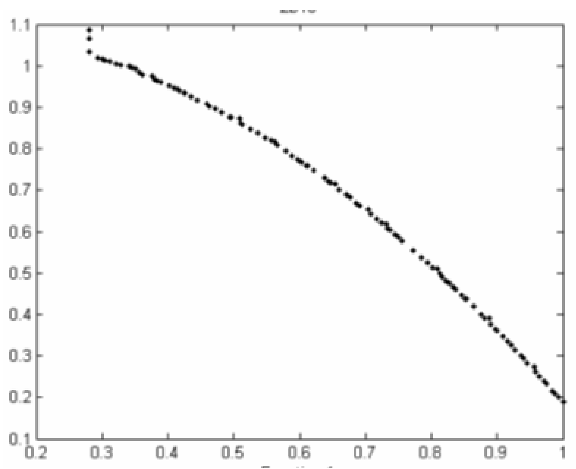

Figure 12: Pareto frontier of ZDT6 by MASAGA SAMOGA

In above simulation figures you can see that SCH, ZDT1, ZDT2, ZDT3, ZDT4 and NSGA II. For The algorithm's convergence results and diversity are better than NSGA-II for the ZDT6 problem, indicating that it has good solving efficiency.

8. Conclusions

In this research paper, a self-adaptive agent which is an evolutionary algorithm is introduced for multiple optimization Problems. A more advanced user can independently apply adaptive strategies, which can be paired with unique problem-dependent skills.

The approach uses the mutation and crossover rates by self-adapts. The six level problems are used to test the algorithm, and from the results we can see that the performance is good. The human factor continues to be the most important factor in the efficient implementation of GA in real-world applications.

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

[7] 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.

[8] Leitao P. (2013) Multi-agent Systems in Industry: Current Trends & Future Challenges. In: Kelemen J., Romportl J., Zackova E. (eds) Beyond Artificial Intelligence. Topics in Intelligent Engineering and Informatics, vol 4. Springer, Berlin, Heidelberg

[9] 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

[10] Y. Zhang, C. Qian, J. Lv and Y. Liu, "Agent and Cyber-Physical System Based Self-Organizing and Self-Adaptive Intelligent Shopfloor," in IEEE Transactions on Industrial Informatics, vol. 13, no. 2, pp. 737-747, April 2017, doi: 10.1109/TII.2016.2618892.

[11] Deb, Kalyan. (2008). Scope of stationary multi-objective evolutionary optimization: A case study on a hydro-thermal power dispatch problem. Journal of Global Optimization. 41. 479-515. 10.1007/s10898-007-9261-y.

[12] Bandyopadhyay, S. and Bhattacharya, R., 2014. Solving a tri-objective supply chain problem with modified NSGA-II algorithm. *Journal of Manufacturing Systems*, 33(1), pp.41-50

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

[14] CHEN Hua-ping,GUFeng,LU Bing-yuan,GU Chun-sheng(Department of Information Management and Decision Science,University of Science and Technology of China,Hefei 230026,China**)**

[15] LianShuan, S. and HuaHui, W., 2015. A Multi-Agent Self-Adaptive Multi-Objective Genetic Algorithm. *International Journal of Intelligent Engineering and Systems*, 8(2), pp.7-13.

[16] Guo, Y., Goncalves, G. and Hsu, T., 2012. RETRACTED ARTICLE: A Multi-agent Based Self-adaptive Genetic Algorithm for the Long-term Car Pooling Problem. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 12(1), pp.45-66.

[17] 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.

[18] YuelinGao ,Yingzhen Chen, and Qiaoyong Jiang Institute of Information and System Science, Beifang University of Nationalities, Yinchuan Ningxia, 750021, China