A Framework for Multi-objective, Biogeography Based Optimization of Complex System Families

Jeffrey Abell [1]
General Motors, Warren, Michigan, 48090

Dawei Du[2]
Cleveland State University, Cleveland, Ohio, 44115

Multi-objectives and multi-constraints problems are a big challenge in the industry, and they are also difficult to be conquered by traditional methods. In this situation, heuristic algorithms become an executable choice when facing this kind of problems. In this paper, we introduce the complex system which is more complicated than normal multi-objectives and multi-constraints problems. For each family of systems, the objective functions and constraints are normally shared among families. Since keeping the certain number of objective functions and constraints for the entire population is no longer a realistic situation in today's industry, it is time to find a new algorithm to deal with complex system. Biogeography-based optimization (BBO) is a newly developed heuristic algorithm, and with the promising and consistent performance in [1], [2] and [3], it becomes a strong competitor in this area. In this paper, BBO will be applied to the complex system families for the system optimization.

Nomenclature

DI	=	Cluster distance between islands
i	=	SIV number for islands with missing data
K	=	Missing data indicator
m	=	Parent number
n	=	SIV number
N	=	Ranking information vector
Tr	=	Area of triangle

I. Introduction

Biogeography-based optimization (BBO) was first introduced in 2008 [1]. Although it is a newly developed heuristic algorithm, its performance is better than many classical algorithms on both standard benchmarks and practical optimization problems. The population of BBO is analogous to an archipelago, and each island in this archipelago is a possible solution to the optimization problem. BBO introduces four important terms in its theory --- habitat suitability index (HSI), suitability index variable (SIV), immigration rate, and emigration rate. The HSI represents the goodness of the island, where a high HSI means good performance and a low HSI means bad

[1] Staff Researcher, Manufacturing Systems Research Lab, 30500 Mound Road, Warren, MI 48090.
[2] Ph.D candidate, Electrical and Computer Engineering, 2121 Euclid Avenue, Cleveland, OH 44115, and AIAA Member Grade.

performance. An SIV represents a solution feature in an island. The immigration rate and emigration rate are important solution characteristics for migration. A high performing island has a high emigration rate and low immigration rate. Conversely, a low performing island has a low emigration rate and high immigration rate. The emigration rate indicates how likely a solution is to share its features with other solutions. The immigration rate indicates how likely a solution is to accept features from other solutions.

One challenge faced when optimizing families of systems is when some objectives and constraints are shared among the families and others are unique to particular families but must still be satisfied. This is the case when developing product families that might have their own unique performance targets and constraints, but could share common objectives, requirements, or components. Bringing these product families to market in an optimal, robust fashion is quite challenging and requires new methods for representation and optimization. This paper presents the current state of our research in this area.

II. Biogeography-based Optimization

In [1], BBO was tested against seven well developed evolutionary algorithms on 14 standard benchmarks. With the best performance on 4 of the benchmarks and the second best performance on 9 of the benchmarks, BBO shows its potential to be a viable alternative as an evolutionary algorithm.

The basic procedure of BBO is as follows [2]:

1) Define the island modification probability, mutation probability, and elitism parameter. Island modification probability is similar to crossover probability in genetic algorithms (GAs). Mutation probability and elitism parameter are the same as in GAs.
2) Initialize the population (*n* islands).
3) Calculate the immigration rate and emigration rate for each island. Good solutions have high emigration rates and low immigration rates. Bad solutions have low emigration rates and high immigration rates.
4) Probabilistically choose the immigrating islands based on the immigration rates. Use roulette wheel selection based on the emigration rates to select the emigrating islands.
5) Migrate randomly selected SIVs based on the selected islands in the previous step.
6) Probabilistically perform mutation based on the mutation probability for each island.
7) Calculate the fitness of each individual island.
8) If the termination criterion is not met, go to step 3; otherwise, terminate.

The Genetic Algorithm (GA) is a widely used heuristic that provides many quite successful solutions for the problems which are hard to solve by traditional methods. Like most of the evolutionary algorithms, BBO and GA are the algorithms which solve the problem by sharing the individual information in the population and updating the population generation by generations using the shared individual information. But compared with GA, the main difference of BBO is that it introduces three different concepts in its information sharing step – the immigration rate, the emigration rate and the immigration.

Here, the immigration rate and emigration rate are used to determine the immigrating islands and emigrating islands (similar with the parents in GA). We provide the islands with high fitness performance a high probability to share their information, and the islands with low fitness performance a high probability to receive information from other islands. In BBO, both immigrating island and emigrating island are always chosen based on the island performance.

Generating a child in BBO is different from GA, because a child in BBO may have more than two parents. In the immigration, each SIV is changeable. So for each SIV position in the immigrating island, we can probabilistically choose an emigrating island based on the immigration rate and emigration rate. This is very different from GA. For problem with n SIVs in each island, a child may has *m* parents ($m \in [2,n+1]$). As we know, a child is usually generated based on two parents in GA. Compared to the crossovers in GA, the immigration is a more flexible information sharing technique.

In [3], we can obtain the Markov models of BBO and GA. Base on both the theoretic results and experiment results, BBO is proved to be a more aggressive algorithm with much better results than GA. That is the main reason we choose the BBO for the multi-objective optimization problem.

III. Complex System Families

For the complex system, it is not the system with only one input, one output, one objective function and sometimes, no constraints. In this modern world, the complex system faces the problems with multi-inputs, multi-

outputs, multi-objections and multi-constraints. The system we talk about in this paper consists of different families, here, and each of them is designed as a multi-objective and multi-constraint sub-system.

In this situation, this system cannot be treated as typical multi-objective system any long. Since each family is defined as a sub-system, so they can be considered as the independent systems. In this case, each family can have different objective functions and constraints. So the complex system is not just a single multi-objective system but a combination of multi-objective systems. Also, the constraints of each family in the system may not be the same as well. At this point, the complex system we use is much more complicated than the ordinary multi-objective problems.

Although complexity of this complex system is higher than most multi-objective problems in the academic research field, it is the system which can be widely found in today's industry. Like in the automobile assembly processing, the factory can manufacture n models of cars. For each model, we now consider it as a family, the family members are the parts of the car. For some auto parts, they are shared by several models. Here is the question – what is the best combination of auto parts the company should purchase, and which purchase strategy can bring the company the largest profit? This is a system with more than one set of objectives, but this kind of problems is so obvious now. So this comes to a question – how can we optimize this complex system?

IV. Optimization of Complex System Families based on BBO

The multi-objective problem has been discussed for years, and it becomes the new trend for the entire heuristic computation area. In recent years, many new heuristic algorithms have been invented as the optimization methods aiming to dealing with multi-objective problems like [4, 5]. Pareto optimality is a commonly used optimization method for multi-objective problems, but its limitation is severe – we can only observe the Pareto front up to 3 dimensions [6]. For problems with higher dimensions, the pair comparison becomes a compromise solution, but it also sacrifices the primary reason for us to use it – the straightforward solution view.

This is the motivation for people to find an alternate method to search for the multi-Pareto front solutions in a single run. For the multi-objective problem, it is not common to find the optimal solutions for all objectives which can dominate all the other solutions. When the researches go further, it is inconvenient to compare more than one result for each solution. In [4], a new method is introduced to be the alternative choice for high dimensional problems. It provides us a new comparison mechanism other than the Pareto front – the non-dominated sorting system. The primary advantage of this sorting system is that the optimization process is back to scalar level other than the vector level in Pareto optimality.

In this paper, we propose a new framework algorithm for multi-objective optimization problems for complex system families. In multi-objective optimization, the algorithms need to calculate the tradeoff and search for the optimization candidate with the constraints of each objective. This is much different from the problem with a single objective. For multi-objective problem, each objective has its own objective function. In this case, there is no unique criterion to evaluate the combination of all candidate solutions. In general, non-dominated sorting [4] and Pareto sorting [5] are the two primary techniques used in the multi-objective optimization problem. The primary contribution of these two algorithms is that these strategies provide us an evaluation method for multi-objective solutions. In order to apply BBO to the multi-objective optimization problems, we plan to create the hybrid BBO (BBO/mdo) with borrowing of non-dominated sorting.

Before we discuss the details of BBO/mdo, there is a major fact we need to mention at first. Most of the heuristic algorithms designed for multi-objective problems are not planned for realistic problems [4, 7]. For these algorithms, each candidate shares the same objective functions and constraints with others. But in complex system, this setup is overturned. Each family in the system is relatively independent, although the family may share the objectives or constraints with each other, but it is not necessary, the family can share the partial setup with others, all the families might not have to implement the identical setup.

The system BBO/mdo designed for complex system which has a more complex situation than traditional multi-objective algorithms. In this case, BBO/mdo cannot identically borrow the multi-objective components from others. In the following part of this section, we will mainly focus on the internal mechanism of BBO/mdo.

The BBO/mdo consists of four basic components including the algorithm setup, immigration probability calculation, non-dominated sorting and immigration. The following of this section aims to introduce each of the components.

B. Algorithm Setup

The setup of BBO/mdo is still based on the basic BBO algorithm, and the difference is we add the new multi-objective components for the complex system families.

1) PO = {AR1, AR2, AR3, ...}, the initialized population consisted of archipelagos.
2) AR = {IS1, IS2, IS3, ...; OBJ1, OBJ2, OBJ3, ...; CO1, CO2, CO3, ... }, an archipelago consisted of islands, objective functions and constraints. In BBO/mdo, each archipelago is considered as a subsystem, which may have different setups to each other. In this case, the archipelago is the family in the complex system.
3) IS = {SIV1, SIV2, SIV3, ...; }, an island consisted of SIVs objective functions and constraints.
4) SIV: the suitability index variable which characterizes the habitability in different aspects.
5) SIVi ∈ P, here P is the SIV domain.
6) PM: the mutation probability used in the algorithm which is predefined by user
7) EP: the elitism parameter which decides how many elitism individuals should be stored for the next generation, and it is predefined by user.

C. Immigration Probability Calculation

For each heuristic algorithm, there two difficulties which should be faced, one is to converge to the optimal solution which is also the ultimate goal for the optimization method. The other is to keep the diversity of population since the duplicated individual can only provide same useful information for population improvement. In BBO/mdo, each family in complex system is considered as a unique archipelago, and it's not necessary to share the same objective functions and constraints. In this case, there is nearly no way to compare the performance of islands from different archipelagos. Since the diversity is the main motivation which keeps improving the population, we come out a theory that the emigration island with a high difference with the immigration island can provide more information than the emigration island with a low difference. At this point, we introduce the fuzzy cluster distance as the major component in BBO/mdo, and this component focuses on the calculation of the difference level for each pair of islands.

Another two factors which need to be considered about are the similarity level of objective function and constraints during the immigration step. For two islands with high similarity level of objective function or constraints, it means that the problems which are faced by these two islands are more similar to each other. This also means the features which are important in one island may have the same importance level in the other. So the immigration between islands with similar objective functions and constraints should be more helpful to each other.

With the discussion above, there are three factors which have major influence on the immigration between islands, so the immigration probability should also be calculated based on these three components: the fuzzy cluster distance between islands, the similarity level of objective functions between islands and the similarity level of constraints between islands.

The distance calculation between islands is not straightforward, since the archipelago may not have the same objectives, so the structure for two islands may not be the same. With the different island structures, the SIV types, and the length these islands should not be the same. When calculate the distance between islands, it may cause the problem that there is no corresponding SIV in emigration islands or immigration islands. In this situation, we refer to [8] for the solution. In this paper, partial distance strategy (PDS) is introduced for the calculation of the distance between two matrices with data missing. With the capability of handling the missing data, PDS becomes a perfectly suitable tool to calculate the distance between islands.

The general formula of the PDS is as follows [8],

1) $$DI_{mn} = \frac{i}{K_n}\sum_{j=1}^{i}(SIV_{nj} - SIV_{mj})^2 K_{nj} \qquad (1)$$

DI is the distance between two islands based on the norm. $DI_{mn} \in D$, where *D* is the distance domain.

Where,

$$K_{nj} = \begin{cases} 0, & if\ SIV_{nj} = N/A \ \ or \ \ SIV_{mj} = N/A \\ 1, & if\ SIV_{nj} \neq N/A \ \ and\ SIV_{mj} \neq N/A \end{cases} \qquad (2)$$

$$K_n = \sum_{j=1}^{i} K_{nj} \qquad (3)$$

The following example is given to illustrate how to perform PDS:

Here we have two islands with different structures: *island1* and *island2*.

Island1 = [2, 3, N/A, 5, 5, 7, N/A]

Island2 = [N/A, 5, 7, N/A, 6, 6, 7]

According to the algorithm, the parameters are setup as follows,

n = 1

m = 2

$i = 7$

$K_n = \sum_{j=1}^{i} K_{nj} = 0 + 1 + 0 + 0 + 1 + 1 + 0 = 3$

$DI_{mn} = \frac{i}{K_n} \sum_{j=1}^{i} \left(SIV_{nj} - SIV_{mj}\right)^2 K_{nj} = 9.33$

So the distance between island1 and island2 is 9.33.

As the second and third components of the immigration probability, the similarity level of objective functions between islands and the similarity level of constraints between islands are relatively easy to calculate. The following algorithm named fast similarity level calculation (FSLC) is implemented in this paper. The algorithm is like follows,

The function has two inputs, X and Y. Each of the inputs is a vector: X = [x1, x2, x3, ...], Y = [y1, y2, y3, ...]. The ultimate goal of FSLC is to find the same elements in both two inputs, and gives out the similarity level (SL) between these inputs. Here, a bigger SL means a higher similarity level.

```
SL = FSLC(X, Y)
for each x ∈ X
for each y ∈ Y
    if (x = y) then
        SL = SL + 1;
    end
end
end
```

1) SOB_{mn} = FSLC (OBJm, OBJn), SOB is the similarity level of objective functions between two archipelagos. All the islands in these two archipelagoes share the same SOB. $SOB_{mn} \in$ So, where So is the objective function similarity domain.
2) SCO_{mn} = FSLC (COm, COn), SCO is the similarity level of constraints between two archipelagos. All the islands in these two archipelagoes share the same SCO. $SCO_{mn} \in$ Sc, where Sc is the constraint similarity domain.
3) After the introduction for all three components of probability, we can calculate the immigration probability correspondingly. For the immigration probability between two islands, the calculation is based on the spider graph showed in the Fig. 1.

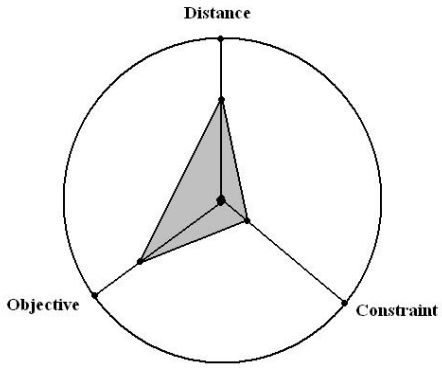

Figure 1. Immigration probability calculation

The circle dot is defined as the origin for fuzzy cluster distance, objective function similarity level and constraints similarity level. The lines between circle dot and dots around the circle represent the domain of DI, SOB and SCO, the small angle between each two lines are 120 degrees. Also, the lengths of all three lines are 100 in this graph. In order to allocate the DI, SOB and SCO value in the line, we should normalize the values of DI, SOB and SCO with minimum value 0 and maximum 100.

The calculation of immigration probability is based on four steps:

1. Calculate the fuzzy distance between two islands, and allocate the DI point in the line which represents the domain DI.
2. Calculate the similarity level of the objective functions between two archipelagos, and allocate the SOB point in the line which represents the domain of SOB.
3. Calculate the similarity level of the constraints between two archipelagos, and allocate the SCO point in the line which represents the domain of SCO.
4. Construct a triangle based on the three points created in step 1, 2 and 3. The immigration probability is calculated as follows,

$Pi = Tr/Tr^*$,

Pi is the immigration probability. *Tr* is the area of the triangle (the grey area in Figure 1.) we created in step four, and *Tr** indicates the maximum area of the triangle.

D. Ranking System for Multi-objective Problems

The idea of non-dominated sorting is first brought out by Kalyanmoy Deb in [4]. With the test of different benchmarks, it is proved to be an efficient algorithm for individual sorting for multi-objective problems.

For the multi-objective system, the performance of each individual is not as straightforward as the single objective problem. For the single problem, the only criterion to determine the performance is the fitness value on the scalar level. For multi-objective problem, the fitness value become on the vector level other than the scalar, and for the individuals which cannot dominate or be dominated by others, we cannot tell the goodness of these individuals only based on the vector level fitness. The aim of the non-dominated sorting is to create a ranking system for multi-objective problems, then regress the fitness back to scalar level. In this paper, we partially borrow the idea of the non-dominated sorting algorithm, and create the ranking system for BBO/mdo. This system is called fast non-dominated sorting system (FNSS).

The basic procedure of FNSS is like follows,

```
for t1= 1 : n
for t2 = t1 : n
    if (is_t1 > is_t2)              if island t1 dominates island t2
        N_t2 = N_t2 + 1             increment N_t2
    else                            if island t2 dominates island t1
        N_t1 = N_t1 + 1             increment N_t1
    end
end
end
return N
```

n: the number of islands in the population

N: the vector contains the ranking information for each island

In FNSS, islands without being dominated by any other islands have the rank 0, and they are also the islands at Pareto front. So the performance of each island depends on the ranking system, the best performance should have rank 0. With bigger rank, the performance should be worse.

E. BBO/mdo Algorithm

Gather all information from previous sections, we already have all the components of the BBO/mdo. In this section, we mainly focus on building BBO/mdo algorithm based on these components.

The basic procedure of BBO/mdo is as follows:

1) Define mutation probability, and elitism parameter. Mutation probability and elitism parameter are the same as in GAs.
2) Initialize the population (n islands).
3) Calculate the immigration probability based on DI_{mn}, SOB_{mn}, SCO_{mn}, which is also called the immigration rate to determine the probability of immigration between all pairs of islands.
4) Calculate the ranks of the islands based on non-dominated sorting algorithm. The rank of each island is also called the emigration rate, it determine the immigration direction during the immigration process.
5) Use Roulette wheel with the immigration probability to choose an immigration/emigration island for each island in the population (Since the immigration direction has not been decided, the island selected by Roulette Wheel can be either immigration island or emigration island.)
6) The immigration direction is determined by the emigration rate. The island with higher emigration rate is chosen as the emigration island which specializes in sending features to the other island. The island with lower emigration rate is chose as the immigration island which specializes in receiving features from other island and upgrades itself based on the received features.
7) After the determination of the immigration direction, operate the immigration and migrate randomly selected features from emigration island to immigration island.
8) Probabilistically perform mutation based on the mutation probability for each island.
9) Calculate the fitness of each individual island.
10) If the termination criterion is not met, go to step 3; otherwise, terminate.

V. Experiment

There is a major difference between BBO/mdo and other heuristic algorithms designed for multi-objective problems. For regular algorithms, the objective functions are fixed for each individual, but in BBO/mdo, the objective functions and constraints for each individual are not necessary to be the same, and this feature provides BBO/mdo a much more flexible way to deal with multi-objective problems. In most of the industry problems, like what is discussed in the complex system families section, with multi-inputs, multi-outputs, multi-objectives and multi-constrains, this is no longer typical multi-objective problems depending on the root setup of this system. For each family, it may have a totally different setup, and it is more like a totally different system than others.

When facing the complex system, the traditional multi-objective algorithms have no capability to deal with it, since their setup is only capable with the fixed setting for all individuals. But with the system in the industry growing more and more complex, it can rarely find a system which perfectly fits the operation requirement of traditional algorithm. In this case, BBO/mdo which provides a more realistic solution method for complex multi-objective problems becomes a solution method which cannot be passed by.

In this section, a case study is proposed to test the performance of BBO/mdo. In [9, 10], they conclude several features which cause difficulty for a multi-objective problem to converge to the Pareto-optimal front or maintain the diversity of the population and test them accordingly. Several benchmarks are created according to these features. Here, we borrow two of them in this section as a case study for BBO/mdo. The objective functions are as follows,

$$
\begin{aligned}
a(x_1) &= x_1 \\
b(x_2, \cdots, x_n) &= 1 + 9 \cdot \sum_{i=2}^{n} x_i/(n-1) \\
c(a,b) &= 1 - \sqrt{a/b} \\
f_1 &= \min(a, f_2) \\
f_2 &= b(x_2, \cdots, x_n)c(a,b)
\end{aligned} \qquad (4)
$$

$$
\begin{aligned}
a(x_1) &= x_1 \\
b(x_2, \cdots, x_n) &= 1 + 9 \cdot \sum_{i=2}^{n} x_i/(n-1) \\
c(a,b) &= 1 - (a/b)^2 \\
f_1 &= \min(a, f_2) \\
f_2 &= b(x_2, \cdots, x_n)c(a,b)
\end{aligned} \qquad (5)
$$

For these two objective functions, f_1 and f_2 are the two objective outputs in the multi-objective problems. For the first objective function, it has a convex Pareto front. In this paper, we need a different style objective function as the second one to demonstrate the performance of BBO/mdo when dealing with different families in the complex system, so the second objective function used in the paper is a function with a nonconvex Pareto front which is opposite to the first one. Also, the boundaries of the inputs in these two objective functions are not necessary to be the same, so it can be considered at the different constraints situation which has detailed description in section IV.

Since the highlight of the BBO/mdo is its capability of dealing with complex system. In this section, we will use both objective functions in the BBO/mdo all at once. So there are two archipelagoes totally in our system, and these two archipelagoes will process two different objective functions simultaneously. Also, these two archipelagoes are not processed independently, otherwise it would lose the reason for our BBO/mdo work.

A. System Setup

In this testing system, we design three different configurations. For each of them, we will test the performance BBO/mdo.

Basically, we have two objective functions – objective function (4) and objective function (5). Also, we have two boundary settings for the inputs of each family. These boundary settings are treated as different constraints for each archipelago.

Boundary setting 1: $x_i \in [0,1]$

Boundary setting 2: $x_i \in [0,2]$

With two objective functions and two constraints, we define three different configurations. The detailed configurations are as follows,

1) Configuration 1

This configuration is designed to test the family performance with different constraints,

Archipelago 1: objective function (4), boundary setting 1

Archipelago 2: objective function (4), boundary setting 2

2) Configuration 2

This configuration is designed to test the family performance with different objective functions

Archipelago 1: objective function (4), boundary setting 1

Archipelago 2: objective function (5), boundary setting 1

3) Configuration 3

This configuration is designed to test the family performance with both different objective functions and different boundary settings.

Archipelago 1: objective function (4), boundary setting 1

Archipelago 2: objective function (5), boundary setting 2

The parameter setup of the complex system is like follows, and the generation number and population size is the same as in [10]

Family number: 2

Generations: 250

Population size: 100 per archipelago

Elitism: 3 per generation

B. Experiment result

In the previous section, there are three configurations designed for the complex system. Depending on different configurations, we will provide three sets of results for analysis. In the following part of this section, the Pareto fronts are created for each archipelago, the Pareto front is a good criterion which can visually determine the performance of the archipelagoes.

The result based on configuration 1 is as follows, and the objective function (4) is called fitness1 and the objective function (5) is called fitness2.

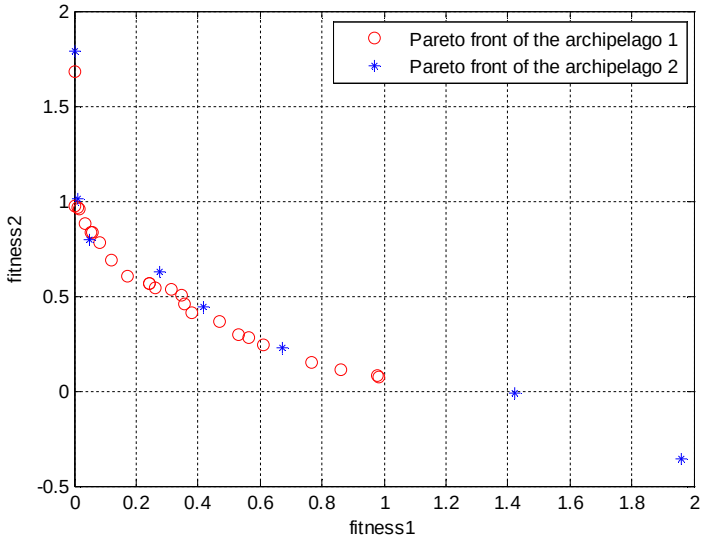

Figure 2. Pareto fronts based on configuration 1

The result based on configuration 2 are shown in Figure 3 below.

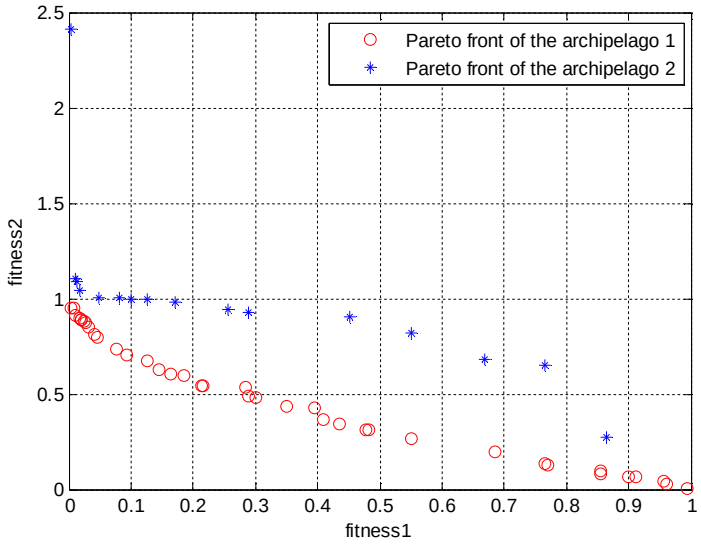

Figure 3. Pareto fronts based on configuration

The results based on configuration 3 are shown in Figure 4 below.

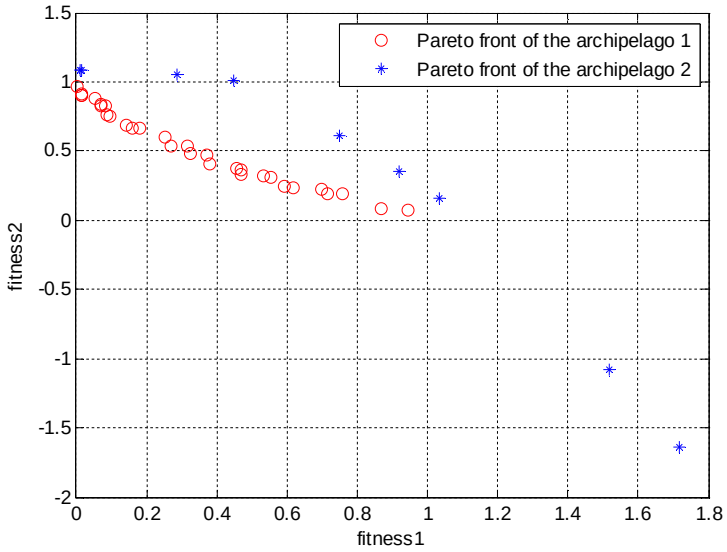

Figure 4. Pareto fronts based on configuration 3

C. Experiment Result Analysis

Since the objective functions used here are based on [10], compared to the results with [10] is a good way to test the performance of BBO/mdo. For the performance of archipelago 1 in three of the configurations, they all keep the high consistence of the performance – the Pareto front starts from the coordinate [0, 1] and end at coordinate [1, 0]. Depend on Fig. 2, 3, and 4, archipelago 1 achieves good and consistent results. Then, we need check the results from archipelago 2. Based on the three configurations of the archipelago 2, its setup is different from archipelago, and because of the different setup from [10], we cannot directly compare these two sets results. But it is clear to see, the trends of the archipelago 2 based on different configurations are very similar to the result from [10]. At this point, the archipelago also achieves the desired performance along with the archipelago 1.

Even though the performance of our work is similar to [10], the point is that we have a much more complicated system which internally faces the different objective function and constraint setups than the traditional multi-

objective problems, and it has much higher tolerance to the flexibility of the system. So, for the real world problem, the advantage of our system compared to others is very obvious – BBO/mdo is a promising system for the complex system.

VI. Conclusion

In this paper, the complex system families are first introduced to the heuristic algorithm area. Compared to the traditional multi-objective problems, the complex system are much more complicated. In traditional problems, the number of the objectives is fixed to each of the individual in the population. Also, the constraint of each individual is also the same to each other. But for the complex system, each family is a subsystem which might have totally setups both for objectives and constraints. With the growth of the industry need, the complex system is widely used. But for the complex system, the solution method used for traditional multi-objective problems cannot fulfill its need by only providing the solution for the identical objectives to each individual. At this point, the urgent need of the solution to the complex system becomes the motivation of our work.

In this paper, BBO/mdo which is based on the basic work of biogeography-based optimization, and its design is specialized for the complex system. Even with different setup of multi-objective functions and constraints, its performance is still consistent and promising. With the results from section V, the system consists of two archipelagos. For each of the archipelago, the setup is different from each other. But according to our experimental result, even for the complex system, the Pareto front in Fig. 2, 3 and 4 still gains the similar trend like in [10]. Remarkable capability of BBO/mdo for complex system is proved with these results.

In this paper, our focus is to build the frame work of BBO/mdo for the complex system. For future work, basically there are two directions. First we plan to extend our work from two dimensions to higher dimensions, and extend capability of BBO/mdo to deal with more realistic situation. Second, a real world case study will be applied to the BBO/mdo study, and it is an opportunity to prove the performance of BBO/mdo in a real industry problem.

Acknowledgments

This work was partially supported by NSF Grant 0826124 in the CMMI Division of the Engineering Directorate.

References

[1] Simon, D., "Biogeography-Based Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 6, Dec. 2008, pp. 702–713.

[2]Du, D., Simon, D., and Ergezer, M., "Biogeography-Based Optimization Combined with Evolutionary Strategy and Immigration Refusal," *IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2009, pp. 1023–1028.

[3] Simon, D., Ergezer, M., Du, D., and Rarick, R., "Analytical and Numerical Comparisons of Biogeography-Based Optimization and Genetic Algorithms," submitted for publication.

[4] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, Apr 2002, pp. 182–197.

[5] Knowles, J. and Corne, D., "The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization," in Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press, 1999, pp. 98–105.

[6] Das, I. and Dennis, J. E., "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems," *SIAM Journal on Optimization*, Vol. 8, No.3, 1998, pp. 631–657.

[7] Deb, K., "Multi-Objective Optimization Using Evolutionary Algorithms," 1st ed., Wiley, 2009.

[8] Hathaway, R. J. and Bezdek, J. C., "Fuzzy c-means clustering of incomplete data," *IEEE transaction on Systems, Man, and Cybernetics*, Vol. 31, No.5, Sep 2001, pp. 735–744.

[9] Deb, K., "Multi-objective genetic algorithms: problem difficulties and construction of test problems," *Evolutionary Computation*, Vol. 7, No.3, 1999, pp. 205–230.

[10] Zitzler, E., Deb, K. and Thiele, L., "Comparison of multiobjective evolution algorithms: empirical results," *Evolutionary Computation*, Vol. 8, No.2, 2000, pp. 173–195.