Dalhousie University
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Technical Report

Optimal Fuzzy Input Set of Fuzzy Relation Equations Using Biogeography-Based Optimization Algorithm

Ali R. Al-Roomi

No. ECED-2016-Jul-27

July 2016

1459 Oxford Street, Halifax, NS B3H 4R2 Canada

How to cite this technical report:

1. APA Referencing Style:

Al-Roomi, A. R. (2016, Jul.). Optimal Fuzzy Input Set of Fuzzy Relation Equations Using Biogeography-Based Optimization Algorithm (Tech. Rep. No. ECED-2016-Jul-27). 1459 Oxford Street, Halifax, NS B3H 4R2 Canada: Dalhousie University, Department of Electrical and Computer Engineering. Retrieved August xx, 20yy, from https://dalspace.library.dal.ca/

2. MLA Referencing Style:

Al-Roomi, Ali R. Optimal Fuzzy Input Set of Fuzzy Relation Equations Using Biogeography-Based Optimization Algorithm. Dalhousie University, Department of Electrical and Computer Engineering, 1459 Oxford Street, Halifax, NS B3H 4R2 Canada, Jul. 2016, https://dalspace.library.dal.ca/. ECED-2016-Jul-27. Accessed xx August 20yy.

3. Chicago Referencing Style:

Al-Roomi, Ali R. Optimal Fuzzy Input Set of Fuzzy Relation Equations Using Biogeography-Based Optimization Algorithm. Report Numbers ECED-2016-Jul-27. 1459 Oxford Street, Halifax, NS B3H 4R2 Canada: Dalhousie University, Department of Electrical and Computer Engineering, Jul. 2001. Accessed August dd, 20yy. https://dalspace.library.dal.ca/.

4. IEEE Referencing Style:

A. R. Al-Roomi, "Optimal Fuzzy Input Set of Fuzzy Relation Equations Using Biogeography-Based Optimization Algorithm," Dalhousie University, Department of Electrical and Computer Engineering, 1459 Oxford Street, Halifax, NS B3H 4R2 Canada, Tech. Report. ECED-2016-Jul-27, 27 Jul. 2016. Accessed on: August xx, 20yy. [Online]. Available: https://dalspace.library.dal.ca/

5. BibTex File:

```
@techreport{Al-Roomi2016,
     title = {{Optimal Fuzzy Input Set of Fuzzy Relation Equations
              Using Biogeography-Based Optimization Algorithm}},
     author = {Ali R. Al-Roomi},
     number = {ECED-2016-Jul-27},
     year = {2016},
     institution = {Dalhousie University, Department of Electrical
                    and Computer Engineering},
     address = {1459 Oxford Street, Halifax, NS B3H 4R2 Canada},
     month = {Jul.},
     url = {https://dalspace.library.dal.ca/},
     note = {Accessed on: August xx, 20yy}
}
```

Summary

This study presents an approach to optimally find a fuzzy input set $\hat{A}$ that can be used with the fuzzy relation R to producing a fuzzy output $\hat{B} \approx B$. In this study, the fuzzy relation is translated into a mathematical optimization model. Although this model can be optimally solved in any meta-heuristic optimization algorithm, the biogeography-based optimization (BBO) algorithm is implemented just as a practical example and evidence of our success. In this study, it has been observed that the quality of the results depends on two criteria; the resolution of λ-cuts and the constraint (whether the estimated fuzzy output $\hat{B}$ is a subset of the actual fuzzy output B or not). If this constraint is enabled in the design function, then obtaining the optimal value becomes a hard task, especially if the λ-cuts are provided with limited discrete elements of the membership function of A. Also, it has been found that the optimal value obtained by Sanchez's operator can be met by many alternatives of $\hat{A}$ vectors obtained by BBO.

Contents

List of Figures

List of Tables

1 Introduction

In fuzzy systems, the term "**fuzzy relation**" is commonly seen in many references, such as the books and technical reports given in (Dubois & Prade, 1980; Zimmermann, 1996; Babuška, 1999; Dubois & Prade, 2000; Lee, 2005; Ross, 2010), and much more in journals and conferences. It is a logical reflection of its importance in many applied fields of fuzzy systems, such as: fuzzy system analysis, design of fuzzy controllers, decision-making process, and fuzzy pattern recognition (El-Hawary, 2016).

Suppose A is a fuzzy set in X, and R is a binary fuzzy relation in $X \times Y \rightarrow$ i.e., $A(X)$ and $R(X,Y)$:

$$A(X) = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \cdots + \frac{\mu_A(x_n)}{x_n} \right\} \tag{1}$$

$$R(X,Y) = \begin{bmatrix} \mu_R(x_1,y_1) & \mu_R(x_1,y_2) & \cdots & \mu_R(x_1,y_m) \\ \mu_R(x_2,y_1) & \mu_R(x_2,y_2) & \cdots & \mu_R(x_2,y_m) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_R(x_n,y_1) & \mu_R(x_n,y_2) & \cdots & \mu_R(x_n,y_m) \end{bmatrix} \tag{2}$$

where $\mu_A(x_i)$ and $\mu_R(x_i,y_j)$ are respectively the membership functions of the ith x of the fuzzy set $A(X)$ and the ith x/jth y of the fuzzy relation $R(X,Y)$.

Then, the fuzzy relation equation of A and R is represented as follows:

$$A \star R = B \tag{3}$$

where B is called the fuzzy output set in Y and its jth membership function is denoted as $\mu_B(y_j)$. Thus, the vector of B can be expressed as follows:

$$B(Y) = \left\{ \frac{\mu_B(y_1)}{y_1} + \frac{\mu_B(y_2)}{y_2} + \cdots + \frac{\mu_B(y_m)}{y_m} \right\} \tag{4}$$

As can be obviously observed in Eq.(3), B depends on the composition operator $\star$[1]. One of the most popular composition operators is known as the ***max-min composition operator***:

$$A \circ R = B \tag{5}$$

If this composition operator is applied here, then the membership function μ_B of the jth y element is defined as follows:

$$\mu_B(y_j) = \mu_{A \circ R}(y_j) \equiv \max_{x \in X} \min \left\{ \mu_A(x), \mu_R(x,y_j) \right\} \tag{6}$$

Instead of using the min-operator, someone could use the product-operator (or any other t-norm):

$$A \bullet R = B \tag{7}$$

So that, Eq.(6) becomes:

$$\mu_B(y_j) = \mu_{A \bullet R}(y_j) \equiv \max_{x \in X} \left\{ \mu_A(x) \cdot \mu_R(x,y_j) \right\} \tag{8}$$

[1]The star symbol $\star$ is a general or universal symbol; it could be $\circ$ for the max-min composition, $\bullet$ for the max-product composition, etc.

Moreover, the max-operator itself can be replaced by any other s-norm operator. Furthermore, both operators can be swapped between each other to have what is called the min-max composition operator where its fuzzy relation and membership function for B are respectively defined as follows:

$$A \diamond R = B \tag{9}$$

$$\mu_B(y_j) = \mu_{A \diamond R}(y_j) \equiv \min_{x \in X} \max \{\mu_A(x), \mu_R(x, y_j)\} \tag{10}$$

1.1 Inverse Fuzzy Relation Equations

For all these composition operators, sometimes the inverse of fuzzy relation equations is required. For example, imagine that A and B are given and R needs to be determined, or R and B are given and A needs to be determined. Because of the nonlinearity of the fuzzy relation equation, so many solutions could be obtained from that inversion process (El-Hawary, 2016).

This study focuses on the second situation where both R and B are given and A is unknown. To find A, there are many operators presented in the literature as helping tools. One of the most popular operators is called the α***-operator***[2]. This operator states that the solution to the fuzzy relation equation $A \circ R = B$ exists if the following inequality constraint holds for all the elements of the vector Y:

$$\max_{x \in X} \{\mu_R(x, y)\} \geqslant \mu_B(y) \ \forall \ y \in Y \tag{11}$$

Thus, the largest fuzzy set $\overset{*}{A}(X)$ that satisfies the fuzzy relation equation $A \circ R = B$ is (El-Hawary, 2016):

$$\overset{*}{A} = R \overset{\alpha}{\longleftrightarrow} B \tag{12}$$

where its membership function for the ith element of x is:

$$\mu_{\overset{*}{A}}(x_i) = \mu_{R \overset{\alpha}{\longleftrightarrow} B}(x_i) \equiv \min \{\mu_R(x_i, y_1) \ \alpha \ \mu_B(y_1), \cdots, \mu_R(x_i, y_m) \ \alpha \ \mu_B(y_m)\} \tag{13}$$

The calculated fuzzy set $\hat{B}$ becomes a subset of the actual fuzzy set B:

$$\left(R \overset{\alpha}{\longleftrightarrow} B\right) \circ R = \overset{*}{A} \circ R = \hat{B} \subseteq B \tag{14}$$

Because the α-operator is applied to the relation between $R(X, Y)$ and $B(Y)$, so it is defined as follows:

$$\mu_R(x_i, y_j) \ \alpha \ \mu_B(y_j) = \begin{cases} 1, & \text{if } \mu_R(x_i, y_j) \leqslant \mu_B(y_j) \\ \mu_B(y_j), & \text{otherwise} \end{cases} \tag{15}$$

Based on that, Eq.(13) becomes:

$$\mu_{\overset{*}{A}}(x_i) = \mu_{R \overset{\alpha}{\longleftrightarrow} B}(x_i) \equiv \min \left\{ \begin{cases} 1, & \text{if } \mu_R(x_i, y_1) \leqslant \mu_B(y_1) \\ \mu_B(y_1), & \text{otherwise} \end{cases}, \cdots, \begin{cases} 1, & \text{if } \mu_R(x_i, y_m) \leqslant \mu_B(y_m) \\ \mu_B(y_m), & \text{otherwise} \end{cases} \right\} \tag{16}$$

Besides the α-operator, there are many other operators presented in the literature, such as (Ahmed & Saqib, 2010):

[2]It is sometimes called ***Sanchez-operator*** because it was introduced by Elie Sanchez who was one of the pioneers in this filed (Sanchez, 1976).

- γ-operator (also known as the ***equality operator***):

$$\mu_R(x_i, y_j) \; \gamma \; \mu_B(y_j) = \begin{cases} 1, & \text{if } \mu_R(x_i, y_j) = \mu_B(y_j) \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

- σ-operator:

$$\mu_R(x_i, y_j) \; \sigma \; \mu_B(y_j) = \begin{cases} 0, & \text{if } \mu_R(x_i, y_j) < \mu_B(y_j) \\ \mu_B(y_j), & \text{otherwise} \end{cases} \tag{18}$$

- ε-operator (it is dual to σ-operator):

$$\mu_R(x_i, y_j) \; \varepsilon \; \mu_B(y_j) = \begin{cases} \mu_B(y_j), & \text{if } \mu_R(x_i, y_j) < \mu_B(y_j) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

1.2 Inverse Fuzzy Relation Equations Using Optimization Approach

We have said that there are many solutions that could be obtained from the inverse fuzzy relation equations due to its nonlinearity nature. Thus, the optimization techniques can be involved here to accomplish this purpose.

But, because A, B, and R are supposed to be in discrete values[3], so someone might ask the following question: ***Can we obtain all the best solutions (that the calculated fuzzy set*** $\hat{B} \rightarrow B$***), without using any optimization method, by checking all the possible sets of*** A***?***

Initially, we can say: YES, we can! However, it is totally unpractical because its memory usage steeply increases with any slight increase in the problem dimension and/or resolution!

To clarify this point, let's take the following problem:

$$B = \left\{ \frac{0.5}{y_1} + \frac{0.3}{y_2} + \frac{0.6}{y_3} + \frac{0.5}{y_4} \right\}$$

$$R = \begin{bmatrix} 0.4 & 0.7 & 0 & 0.2 \\ 0 & 0.6 & 0 & 0.7 \\ 0.2 & 0.8 & 0.5 & 0.6 \\ 0.9 & 0.7 & 1 & 0 \end{bmatrix}$$

Using MATLAB code, given in Appendix II, with step-size of 0.1 for A, we get 40 optimal solutions out of 14641 sets of A within 0.304406 seconds:

```
Solution No.1:
Fitness = 0.7
Fuzzy Set A =   0           0           0           0
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.2:
Fitness = 0.7
Fuzzy Set A =   0         0.1         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.3:
```

[3] i.e., their membership functions are sliced based on the resolutions or step-sizes of their λ-cuts.

```
Fitness = 0.7
Fuzzy Set A =   0         0.2         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.4:
Fitness = 0.7
Fuzzy Set A =   0         0.3           0         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.5:
Fitness = 0.7
Fuzzy Set A =   0         0.3         0.1         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.6:
Fitness = 0.7
Fuzzy Set A =   0         0.3         0.2         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.7:
Fitness = 0.7
Fuzzy Set A =   0         0.3         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.8:
Fitness = 0.7
Fuzzy Set A = 0.1           0         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.9:
Fitness = 0.7
Fuzzy Set A = 0.1         0.1         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.10:
Fitness = 0.7
Fuzzy Set A = 0.1         0.2         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.11:
Fitness = 0.7
Fuzzy Set A = 0.1         0.3           0         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.12:
Fitness = 0.7
Fuzzy Set A = 0.1         0.3         0.1         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.13:
Fitness = 0.7
Fuzzy Set A = 0.1         0.3         0.2         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.14:
Fitness = 0.7
Fuzzy Set A = 0.1         0.3         0.3         0.3
```

```
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.15:
Fitness = 0.7
Fuzzy Set A = 0.2           0         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.16:
Fitness = 0.7
Fuzzy Set A = 0.2         0.1         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.17:
Fitness = 0.7
Fuzzy Set A = 0.2         0.2         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.18:
Fitness = 0.7
Fuzzy Set A = 0.2         0.3           0         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.19:
Fitness = 0.7
Fuzzy Set A = 0.2         0.3         0.1         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.20:
Fitness = 0.7
Fuzzy Set A = 0.2         0.3         0.2         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.21:
Fitness = 0.7
Fuzzy Set A = 0.2         0.3         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.22:
Fitness = 0.7
Fuzzy Set A = 0.3           0         0.3           0
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.23:
Fitness = 0.7
Fuzzy Set A = 0.3           0         0.3         0.1
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.24:
Fitness = 0.7
Fuzzy Set A = 0.3           0         0.3         0.2
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.25:
Fitness = 0.7
Fuzzy Set A = 0.3           0         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
```

```
Solution No.26:
Fitness = 0.7
Fuzzy Set A = 0.3         0.1         0.3           0
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.27:
Fitness = 0.7
Fuzzy Set A = 0.3         0.1         0.3         0.1
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.28:
Fitness = 0.7
Fuzzy Set A = 0.3         0.1         0.3         0.2
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.29:
Fitness = 0.7
Fuzzy Set A = 0.3         0.1         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.30:
Fitness = 0.7
Fuzzy Set A = 0.3         0.2         0.3           0
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.31:
Fitness = 0.7
Fuzzy Set A = 0.3         0.2         0.3         0.1
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.32:
Fitness = 0.7
Fuzzy Set A = 0.3         0.2         0.3         0.2
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.33:
Fitness = 0.7
Fuzzy Set A = 0.3         0.2         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.34:
Fitness = 0.7
Fuzzy Set A = 0.3         0.3           0         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.35:
Fitness = 0.7
Fuzzy Set A = 0.3         0.3         0.1         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.36:
Fitness = 0.7
Fuzzy Set A = 0.3         0.3         0.2         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.37:
Fitness = 0.7
```

```
Fuzzy Set A = 0.3         0.3         0.3           0
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.38:
Fitness = 0.7
Fuzzy Set A = 0.3         0.3         0.3         0.1
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.39:
Fitness = 0.7
Fuzzy Set A = 0.3         0.3         0.3         0.2
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Solution No.40:
Fitness = 0.7
Fuzzy Set A = 0.3         0.3         0.3         0.3
Fuzzy Set B = 0.3         0.3         0.3         0.3
-----------------------------------------------------
Elapsed time is 0.304406 seconds.
```

If the resolution of A is increased from 0.1 to 0.05, then we get 133 optimal solutions out of 194481 possible sets, but within **37.730767** seconds. Now, imagine a high resolution of λ-cuts with large n and $m \rightarrow$ i.e., a $high$-dimensional problem!

Based on this fact, the optimization techniques are preferable for this kind of problem. However, because we are dealing with discrete variables, so only some special fully discretized optimization techniques can be used. They are called ***combinational optimization algorithms***. By these algorithms, we will not worry anymore about the machine's CPU time as the problem dimension and/or resolution increase.

In the next lines, we are going to use our designed ***combinational biogeography-based optimization*** (***CBBO***) algorithm. The MATLAB source code is given in Appendix III. The mechanism behind this algorithm is fully described in (Simon, 2013; Al-Roomi & El-Hawary, 2016a, 2016b), so it will not be covered here in order to save space.

2 Modeling Fuzzy Relation Equation as a Combinational Optimization Problem

In this section, the solution to the previous problem is modeled as an optimization problem. The following mathematical expression covers both the objective function and design constraints.

Recalling the previous problem:

$$\because \; A \circ R = B \tag{20}$$

where

$$\text{size}\{A\} = [1 \times 4], \;\; \text{size}\{R\} = [4 \times 4], \;\; \text{and size}\{B\} = [1 \times 4]$$

Suppose:

$$A = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \frac{\mu_A(x_3)}{x_3} + \frac{\mu_A(x_4)}{x_4} \right\} = \left\{ \frac{a_1}{x_1} + \frac{a_2}{x_2} + \frac{a_3}{x_3} + \frac{a_4}{x_4} \right\} \tag{21}$$

$$\because\ R = \begin{bmatrix} 0.4 & 0.7 & 0.0 & 0.2 \\ 0.0 & 0.6 & 0.0 & 0.7 \\ 0.2 & 0.8 & 0.5 & 0.6 \\ 0.9 & 0.7 & 1.0 & 0.0 \end{bmatrix}\ ,\ \text{and}\ B = \begin{bmatrix} 0.5 & 0.3 & 0.6 & 0.5 \end{bmatrix} \tag{22}$$

Thus, applying Eq.(20) yields:

$$\therefore\ A \circ R = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \circ \begin{bmatrix} 0.4 & 0.7 & 0.0 & 0.2 \\ 0.0 & 0.6 & 0.0 & 0.7 \\ 0.2 & 0.8 & 0.5 & 0.6 \\ 0.9 & 0.7 & 1.0 & 0.0 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.3 & 0.6 & 0.5 \end{bmatrix} \tag{23}$$

$$\therefore\ \hat{A} \circ R = \begin{bmatrix} \hat{a}_1 & \hat{a}_2 & \hat{a}_3 & \hat{a}_4 \end{bmatrix} \circ \begin{bmatrix} 0.4 & 0.7 & 0.0 & 0.2 \\ 0.0 & 0.6 & 0.0 & 0.7 \\ 0.2 & 0.8 & 0.5 & 0.6 \\ 0.9 & 0.7 & 1.0 & 0.0 \end{bmatrix} = \begin{bmatrix} \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \hat{b}_4 \end{bmatrix} \tag{24}$$

$$\therefore\ \hat{B} = \begin{bmatrix} [(\hat{a}_1 \wedge 0.4) \vee (\hat{a}_2 \wedge 0.0) \vee (\hat{a}_3 \wedge 0.2) \vee (\hat{a}_4 \wedge 0.9)] \\ [(\hat{a}_1 \wedge 0.7) \vee (\hat{a}_2 \wedge 0.6) \vee (\hat{a}_3 \wedge 0.8) \vee (\hat{a}_4 \wedge 0.7)] \\ [(\hat{a}_1 \wedge 0.0) \vee (\hat{a}_2 \wedge 0.0) \vee (\hat{a}_3 \wedge 0.5) \vee (\hat{a}_4 \wedge 1.0)] \\ [(\hat{a}_1 \wedge 0.2) \vee (\hat{a}_2 \wedge 0.7) \vee (\hat{a}_3 \wedge 0.6) \vee (\hat{a}_4 \wedge 0.0)] \end{bmatrix}^T = \begin{bmatrix} \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \hat{b}_4 \end{bmatrix} \tag{25}$$

where $\hat{A}$ and $\hat{B}$ are respectively the optimal and backward calculated fuzzy sets obtained for the fuzzy sets A and B.
Therefore, the ***objective function*** can be modeled as follows:

$$\text{OBJ} = \underset{\hat{a}_1,\hat{a}_2,\hat{a}_3,\hat{a}_4}{\text{minimize}} \sum_{j=1}^{4} \left| b_j - \hat{b}_j \right| \oslash P_j \tag{26}$$

where $\oslash$ is an arithmetic operator, which could be "$+$" or "$\times$" based on the type of P_j used. Also, the term b_j denotes the jth element of the fuzzy set B and the term P_j is called the ***penalty function***[4], and both can be defined as follows:

$$b_j = \mu_B(y_j) \tag{27}$$

$$\begin{aligned} B &= \left\{ \frac{\mu_B(y_1)}{y_1} + \frac{\mu_B(y_2)}{y_2} + \frac{\mu_B(y_3)}{y_3} + \frac{\mu_B(y_4)}{y_4} \right\} = \left\{ \frac{b_1}{y_1} + \frac{b_2}{y_2} + \frac{b_3}{y_3} + \frac{b_4}{y_4} \right\} \\ &= \left\{ \frac{0.5}{y_1} + \frac{0.3}{y_2} + \frac{0.6}{y_3} + \frac{0.5}{y_4} \right\} \end{aligned} \tag{28}$$

$$P_j = \begin{cases} 0, & \text{if } \hat{b}_j \leqslant b_j \\ z_j, & \text{otherwise} \end{cases} \tag{29}$$

where z_j is a user defined penalty factor.

Besides the penalty functions, there are many other constraint-handling techniques available in the literature. We have summarized the most popular types in Fig. 1 (Deb, 2010; Venkataraman, 2009; Simon, 2013; Rao, 2009; Eiben & Smith, 2003; Yeniay, 2005; Kajee-Bagdadi, 2007; Yu & Gen, 2010). For simplicity, the ***exterior penalty function*** (***EPF***) is selected here.

[4]To satisfy the condition $\hat{A} \circ R \subseteq B$, which is equivalent to $\left(R \overset{\alpha}{\longleftrightarrow} B\right) \circ R \subseteq B$.

It has to be said that EPF comes in two common forms known as ***additive*** and ***multiplicative*** forms, which can be respectively expressed as follows (Yeniay, 2005):

$$\min_X \phi(X), \;\; where \;\; \phi(X) = \begin{cases} f(X), & \text{if } X \in \mathcal{F} \\ f(X) + P(X), & \text{if } X \notin \mathcal{F} \end{cases} \tag{30}$$

$$\min_X \phi(X), \;\; where \;\; \phi(X) = \begin{cases} f(X), & \text{if } X \in \mathcal{F} \\ f(X) \times P(X), & \text{if } X \notin \mathcal{F} \end{cases} \tag{31}$$

where $\mathcal{F}$ means having a ***feasible set***. The penalty function $P(X)$ is also called the ***penalty term***, which is equal to zero for feasible individuals, and becomes a positive value in case any one of the constraints is violated. Thus, the ***penalized cost function*** $\phi(X)$ becomes higher than its actual value $f(X)$. This $P(X)$ can be provided in different forms based on the type of penalty functions. The most common form is:

$$P(X) \;\; = \;\; \sum_{j=1}^{m} r_j \; \hat{g}_j(X) \tag{32}$$

$$\text{where: } \hat{g}_j(X) \;\; = \;\; \left[\max\left\{0, g_j(X)\right\} \right]^{\beta} \tag{33}$$

where r_j is called the ***penalty multiplier***, and β is a user-defined positive constant which is commonly set equal to either 1 or 2 (Simon, 2013; Rao, 2009; Eiben & Smith, 2003). $g_j(X)$ is the constraint of the problem, which is defined as follows:

$$g_j(X) = \hat{b}_j - b_j \tag{34}$$

If $\hat{b}_j > b_j$, then it means that the jth constraint of Eq.(14) is violated. Thus, $g_j(X)$ becomes a positive value, which in turn activates the term $\hat{g}_j(X)$ given in Eq.(33).
Therefore, the operator $\textcircled{\scriptsize \Delta}$ given in Eq.(26) means the following:

$$\textcircled{\scriptsize \Delta} = \begin{cases} +, & \text{if } \phi \text{ is an additive EPF} \\ \times, & \text{if } \phi \text{ is a multiplicative EPF} \end{cases} \tag{35}$$

For the sake of simplicity, the additive form is used, so Eq.(26) becomes:

$$\text{OBJ} = \underset{\hat{a}_1,\hat{a}_2,\hat{a}_3,\hat{a}_4}{\text{minimize}} \sum_{j=1}^{4} \left| b_j - \hat{b}_j \right| + P_j \tag{36}$$

This objective function can be expressed in many other ways. For example, the following alternative expression can be used to have a sensitive objective function:

$$\text{OBJ} = \underset{\hat{a}_1,\hat{a}_2,\hat{a}_3,\hat{a}_4}{\text{minimize}} \sum_{j=1}^{4} \left(b_j - \hat{b}_j \right)^2 + P_j \tag{37}$$

From Fig. 1, if EPF is selected to be a ***death penalty*** type, such as the ***infinite barrier penalty***, then z becomes a scalar for all $j \in m$ (here $m = 4$) with a constant value of 10^{20} (Deb, 2010). Although this type of penalties is very simple, it is also very bad type because it kills all the infeasible solutions even if some have very small or ignorable violations, which in

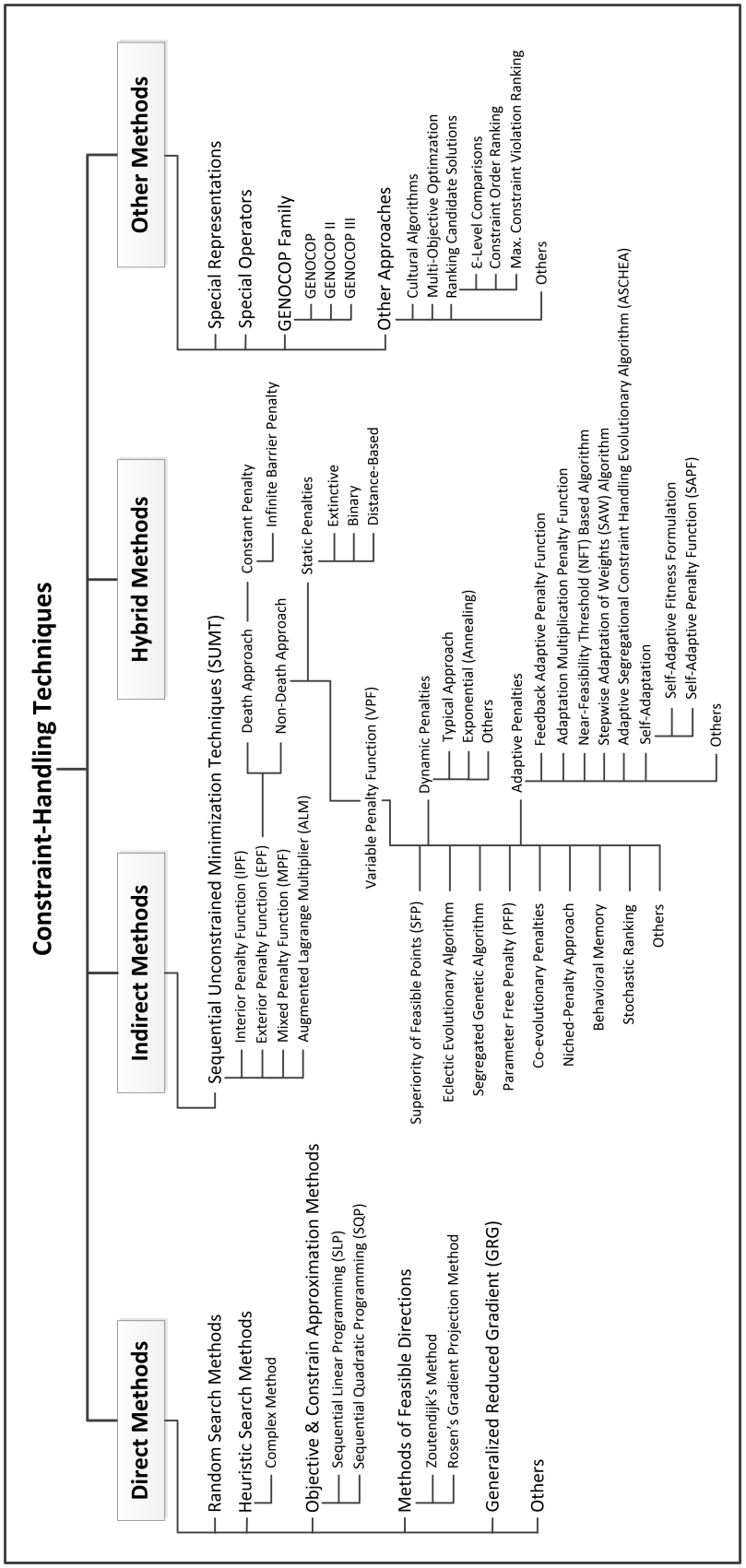


Figure 1. Summary of the Most Popular Constraint-Handling Techniques

turn reduces the exploration level of the optimization algorithm. Thus, the **non-death penalty** type is preferred.

Again, for simplicity, the **static** non-death EPF is selected. The **extinctive** and **binary** subtypes do not help us in solving this problem; *please refer to their properties in* (Simon, 2013; Rao, 2009; Eiben & Smith, 2003). Thus, we will use the **distance-based** static non-death exterior penalty function (with $\beta = 1$) as a constraint-handling technique to solve our fuzzy relation equation presented in Eq.(36) as an optimization problem. Add to that the following penalty factor[5] is used:

$$z_j = 0.5 \; \forall \; j \; \in \; m = 4 \tag{38}$$

3 Experimental Results

In this experiment, we have two cases:

1. Deactivating the penalty function $\rightarrow$ i.e., $\phi = 0 \; \forall \; j \; \in m = 4$
2. Activating the penalty function $\rightarrow$ i.e., $\phi \neq 0 \; \forall \; j \; \in m = 4$

For each case, the following resolutions are considered for the λ-cuts of $\hat{A}$ and $\hat{B}$:

- $\lambda \; \rightarrow \; 0, 0.1, 0.2, \cdots, 1$
- $\lambda \; \rightarrow \; 0, 0.01, 0.02, \cdots, 1$

The CBBO algorithm is initiated using the following settings:

- ***Population Size:*** 20
- ***Elitism:*** 1
- ***No. of Generations:*** 100
- ***Maximum Mutation Probability:*** 0.1

This simulation is coded in MATLAB 2016a using a computing machine with the following specifications: ALIENWARE M14x laptop, 64-bit Windows 10 OS, Intel Core i7-4700MQ CPU 2.4 GHz, and 16 GB RAM. Executing the BBO program gives the results tabulated in Table 1.

As can be clearly observed from the preceding table, the optimal value that matches between the estimated fuzzy output set $\hat{B}$ (obtained by a backward substitution $\rightarrow$ see Eqs.(24)-(25)) and the actual fuzzy output set B cannot be satisfied in this problem (i.e., $B - \hat{B} \neq 0$). We have tried to solve it by using a population size of 60, and 1,000,000 generations without getting any improvement in the convergence! If the constraint $\hat{A} \circ R \subseteq B$ is embedded in the design function, then the best possible value that could be obtained[6] is equal to $\overset{*}{A} = (R \overset{*}{\leftrightarrow} B) \circ R$, which is the solution of Elie Sanchez; i.e., the solution of α-operator (Sanchez, 1976). However, we have found that the BBO algorithm can find many optimal fuzzy input sets $\hat{A}$ that give the

[5] Please note that these factors could give better results if they are well-tuned.

[6] This assumption is based on our quick observation with different initialization settings of BBO.

Table 1: BBO Simulation Results for solving $\hat{A} \circ R = \hat{B}$ Fuzzy Relation Equation

	Unconstrained OBJ ($\phi = 0 \; \forall \; j \; \in \; 4$)		**Constrained OBJ** ($\phi \neq 0 \; \forall \; j \; \in \; 4$)	
	λ in 0.1 steps	λ in 0.01 steps	λ in 0.1 steps	λ in 0.01 steps
$\left[\hat{A}\right]^T$	$\begin{bmatrix} 0.2 \\ 0.2 \\ 0.5 \\ 0.5 \end{bmatrix}^T$	$\begin{bmatrix} 0.42 \\ 0.5 \\ 0.28 \\ 0.5 \end{bmatrix}^T$	$\begin{bmatrix} 0.3 \\ 0.2 \\ 0.3 \\ 0.0 \end{bmatrix}^T$	$\begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.29 \end{bmatrix}^T$
$\left[\hat{B}\right]^T$	$\begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}^T$	$\begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}^T$	$\begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}^T$	$\begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}^T$
CPU (s)	0.202261	0.209260	0.198302	0.164619
OBJ[a]	0.3	0.3	0.7	0.7

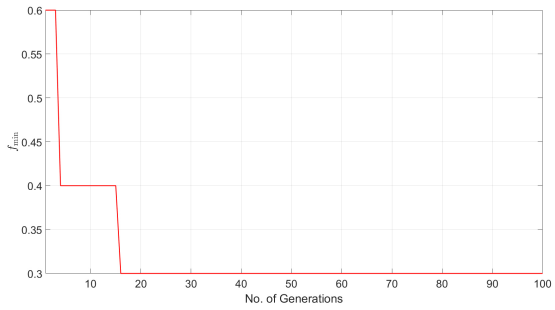
[a]The results obtained by α-operator (also known as **Elie Sanchez's operator**) provides $\overset{*}{A} = \begin{bmatrix} 0.3 & 0.3 & 0.3 & 0.3 \end{bmatrix}$ and $\overset{*}{A} \circ R = \begin{bmatrix} 0.3 & 0.3 & 0.3 & 0.3 \end{bmatrix} \rightarrow \text{OBJ} = 0.7$

same solution of α-operator (i.e., $\hat{A} \circ R = \overset{*}{A} \circ R$). This is a logical phenomenon because we have seen in Sub-Sec. 1.2 that there are 40 optimal solutions if the resolution is 0.1, and 133 optimal solutions if the resolution is 0.01. The benefit of using combinational optimization algorithms rather than the primitive brute-force search approach, given in Sub-Sec. 1.2, can be seen in the total CPU time required to search within the fully discretized domain. In the brute-force search, the algorithm takes 37.730767 to complete its search if the resolution is set to 0.05. In CBBO, changing the resolution from 0.1 to even 0.01 will have an ignorable effect on its total CPU time. Of course, as the resolution and/or dimension increase the algorithm initialization settings should be updated with a higher number of generations and bigger population size. However, the additional increment in CPU time is incomparable with that of the brute-force search. Thus, the combinational optimization approach is superior compared with other approaches. The optimal results tabulated in Table 1 are graphically presented in Fig. 2.

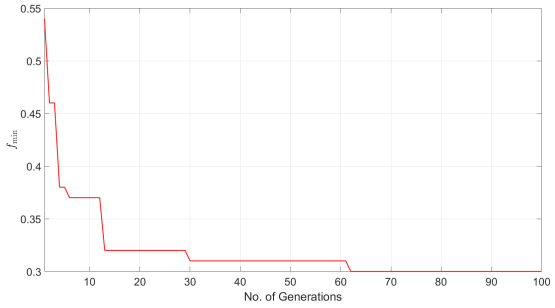
4 Conclusion and Scope of Future Work

This study presents an alternative approach to determine the fuzzy input set A used with the fuzzy relation R to produce the fuzzy output set B through the max-min composition. The modeled objective function proved its ability to solve such kind of fuzzy relation equations. Through the experimental results, it has been found that the solution quality depends on two main factors: the slice resolution of λ-cuts and whether the results extracted from the max-min composition $\hat{B}$ is considered as a subset of the actual fuzzy output set B or not.

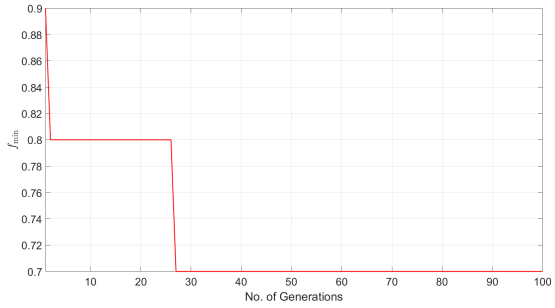
The beauty of this numerical approach is that it can be applied for any type of t-norms/s-norms compositions. With a little care on the constraint-handling technique used in the optimization algorithm, any user-defined problem can be easily entered as a plug-in problem; it will act as a universal optimizer. Thus, it is interesting to test this optimization-based approach with



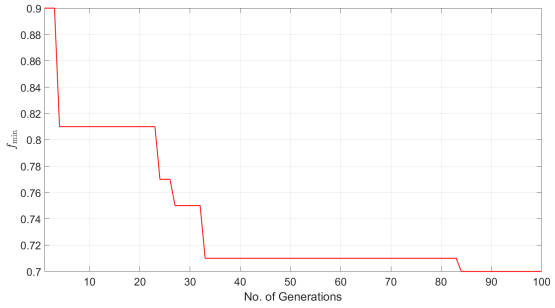
(a) $\phi = 0 \; \forall \; j \; \in 4$ and $\lambda \; \rightarrow \; 0, 0.1, 0.2, \cdots, 1.$



(b) $\phi = 0 \; \forall \; j \; \in 4$ and $\lambda \; \rightarrow \; 0, 0.01, 0.02, \cdots, 1.$



(c) $\phi \neq 0 \; \forall \; j \; \in 4$ and $\lambda \; \rightarrow \; 0, 0.1, 0.2 \cdots, 1.$



(d) $\phi \neq 0 \; \forall \; j \; \in 4$ and $\lambda \; \rightarrow \; 0, 0.01, 0.02 \cdots, 1.$

Figure 2. Fitness Extracted from the BBO Algorithm.

different compositions of s-norms/t-norms. Moreover, it is also interesting to repeat this comparative work with other operators rather than using just α-operator. For instance, γ-operator, σ-operator, and ϵ-operator. Further, the other optimization techniques, including the classical mathematical programming and hybrid algorithms, can be used to enhance the overall performance of the algorithm.

Acknowledgement

I would like to thank Prof. M. E. El-Hawary for his continuous support, advice, motivation, enthusiasm, and immense knowledge. His valuable instructions, ideas, tricks, hints, and updated information helped me in writing this research. Thank you so much Professor.

Appendices

Appendix I: List of Software

Different programs have been used to write this technical report. They are listed as follows:

- Mathworks: *MATLAB R2016a*
- LaTeX: MikTex 2.9, WinEdt 8, JabRef 2.9.2, and LaTable 0.7.2
- Adobe: *Photoshop CS6 (v.13.0)*, and *Acrobat Pro XI*

Appendix II: Source Code of the Primitive Brute-Force Search Method

This appendix provides our MATLAB source code used to find all the possible solutions of the best fuzzy set $\hat{A}$ by the primitive brute-force method:

```
% It is written in the 23rd of July 2016 @ 08:22AM by Ali Ridha Al-Roomi

clear
clc

tic; % start counting the processing speed

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% INITIALIZATION STAGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

B=[0.5 0.3 0.6 0.5]; % Fuzzy set B
R=[0.4 0.7 0 0.2;0 0.6 0 0.7;0.2 0.8 0.5 0.6;0.9 0.7 1 0]; % Fuzzy Relation
P=1000*ones(1,4); % Penalty vector

% Setting the discrete variable bounds (search space) of each element of A
cut=0.05; % possible alpha-cuts of membership function
MSF=0:cut:1; % membership function vector
j=1; % initial value of the jth counter

for i1=1:length(MSF)
    for i2=1:length(MSF)
        for i3=1:length(MSF)
            for i4=1:length(MSF)
                % Finding all the possibilities of the fuzzy input set A
                A(j,:)=[MSF(i1),MSF(i2),MSF(i3),MSF(i4)];
                j=j+1; % update the jth counter
            end
        end
    end
end

% Evaluate the backward fuzzy relation for all the possibilities of A
for i=1:length(A)
    b1(i)=max([min(A(i,1),R(1,1)),min(A(i,2),R(2,1)),min(A(i,3),R(3,1)),min
        (A(i,4),R(4,1))]);
    b2(i)=max([min(A(i,1),R(1,2)),min(A(i,2),R(2,2)),min(A(i,3),R(3,2)),min
        (A(i,4),R(4,2))]);
```

```
    b3(i)=max([min(A(i,1),R(1,3)),min(A(i,2),R(2,3)),min(A(i,3),R(3,3)),min
        (A(i,4),R(4,3))]);
    b4(i)=max([min(A(i,1),R(1,4)),min(A(i,2),R(2,4)),min(A(i,3),R(3,4)),min
        (A(i,4),R(4,4))]);

    % Penalty Factors
    if B(1)==b1(i) || b1(i)-B(1)<cut/10
        P1(i)=0;
    else
        P1(i)=P(1);
    end

    if B(2)==b2(i) || b2(i)-B(2)<cut/10
        P2(i)=0;
    else
        P2(i)=P(2);
    end

    if B(3)==b3(i) || b3(i)-B(3)<cut/10
        P3(i)=0;
    else
        P3(i)=P(3);
    end

    if B(4)==b4(i) || b4(i)-B(4)<cut/10
        P4(i)=0;
    else
        P4(i)=P(4);
    end

    % Objective Function (OBJ)
    F(i)=abs(B(1)-b1(i))+abs(B(2)-b2(i))+abs(B(3)-b3(i))+abs(B(4)-b4(i))+P1
        (i)+P2(i)+P3(i)+P4(i);
end

% Sort and map the fitness
[S,index]=sort(F);

Best(1)=S(1);
B1(1)=b1(index(1));
B2(1)=b2(index(1));
B3(1)=b3(index(1));
B4(1)=b4(index(1));
for i=2:length(S)
    % Accept the other solutions only if they show same fitness
    if S(i)==S(i-1)
        Best(i)=S(i);
        A1(i)=A(index(i),1);
        A2(i)=A(index(i),2);
        A3(i)=A(index(i),3);
        A4(i)=A(index(i),4);
        B1(i)=b1(index(i));
        B2(i)=b2(index(i));
        B3(i)=b3(index(i));
        B4(i)=b4(index(i));
    else
        break;
```

```
    end
end

% Display Results
for i=1:length(Best)
    disp(['Solution No.', num2str(i), ':'])
    disp(['Fitness = ', num2str(Best(i))])
    disp(['Fuzzy Set A = ', num2str([A1(i) A2(i) A3(i) A4(i)])])
    disp(['Fuzzy Set B = ', num2str([B1(i) B2(i) B3(i) B4(i)])])
    disp('------------------------------------------------------------')
end

toc;
```

Appendix III: Source Code of the Combinational BBO Algorithm

This appendix provides our MATLAB source code of the CBBO algorithm used in this study. The below code is adjusted to optimally find the fuzzy input set $\hat{A}$ with ($\phi \neq 0$ and $\lambda \rightarrow 0, 0.01, 0.02 \cdots, 1$) case:

```
% It is my designed BBO Program to get fmin by using Random Function Method
%%%%%%%%%%%%%%%%%%%%%%%%% Partial Migration-Based BBO %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Constrained Problem %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% It is written in the 17th of July 2016 @ 01:15PM by Ali Ridha Al-Roomi

clear
clc

tic; % start counting the processing speed

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% INITIALIZATION STAGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

B=[0.5 0.3 0.6 0.5]; % Fuzzy set B
R=[0.4 0.7 0 0.2;0 0.6 0 0.7;0.2 0.8 0.5 0.6;0.9 0.7 1 0]; % Fuzzy Relation
P=[0.5,0.5,0.5,0.5]; % Penalty vector

% Setting the discrete variable bounds (search space) of each element of A
cut=0.01; % possible alpha-cuts of membership function
MSF=0:cut:1; % membership function vector

Habitat=20; % Total number of islands (solutions)
Elitism=1; % Total number of saved solutions
Species=4; % Total number of species (independent variables for each
    habitat)
Loop=100; % Total number of required generations before getting the final
    solutions
I=1; % Maximum Immigration
E=1; % Maximum Emigration
m_max=0.1; % User-defined probability mutation
Mut_Opt=1; % Mutation Option: if equal to 1, then mutation stage is enabled
MR=round(Habitat/3); % Mutation Range respecting to total number of islands
```

```
CDR=MR; % Clear Duplication Range, it can be set 1, elitism or MR, by
    defaul
% it equal to 1 (Dan Simon), but with this feature, we have more control


%%%%%%%%%%%%%%%%%%%%%%%% Calculating Ps(t) & m(t) %%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:Habitat % Calculating Lambda, Mu & Species count
    Species_count(i)=Habitat-i;
    Lambda(i)=1-(Species_count(i)/(Habitat-1));
    Mu(i)=Species_count(i)/(Habitat-1);
end

for j=1:Habitat
    Prob2_initial(j)=1/Habitat; % Setting the initial probability values
end

% We want to calculate Lambda(S-1) & Mu(S+1), so from the Probaibility
% equation that is described by Dan Simon in his paper can be used here

Lambda_Minus=ones(1,Habitat);
Lambda_Minus(1:Habitat-1)=Lambda(2:Habitat);
Lambda_Minus(Habitat)=0;

Mu_Plus=ones(1,Habitat);
Mu_Plus(2:Habitat)=Mu(1:Habitat-1);
Mu_Plus(1)=0;

dt=1; % Delta t for calculating Ps(t+dt)

for k=1:Habitat

    if k==1
        Prob2=Prob2_initial;
    end

    Prob2_Minus=ones(1,Habitat);
    Prob2_Minus(1:Habitat-1)=Prob2(2:Habitat);
    Prob2_Minus(Habitat)=0;

    Prob2_Plus=ones(1,Habitat);
    Prob2_Plus(2:Habitat)=Prob2(1:Habitat-1);
    Prob2_Plus(1)=0;

    Prob2_Derivative=-(Mu+Lambda).*Prob2+Lambda_Minus.*Prob2_Minus+Mu_Plus
        .*Prob2_Plus;

    Prob2=Prob2+Prob2_Derivative*dt;

end

Prob2(Prob2<0)=0; % replacing any negative value to zero
Prob2=Prob2/sum(Prob2); % sum of the new Ps must be equal 1
Probability=Prob2; % Probability=Ps(t)

% Calculating the Mutation Rate m(t)
Pmax=max(Probability);
```

```
Mutation_Rate=m_max*(1-Probability/Pmax);


SIVindex=randi(length(MSF),Species,Habitat);
for i=1:Habitat
    SIV1(i)=MSF(SIVindex(1,i));
    SIV2(i)=MSF(SIVindex(2,i));
    SIV3(i)=MSF(SIVindex(3,i));
    SIV4(i)=MSF(SIVindex(4,i));
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preparing Function: HSI=f(SIV1,SIV2,SIV3,SIV4
    )

for i=1:Habitat

    b1=max([min(SIV1(i),R(1,1)),min(SIV2(i),R(2,1)),min(SIV3(i),R(3,1)),min
        (SIV4(i),R(4,1))]);
    b2=max([min(SIV1(i),R(1,2)),min(SIV2(i),R(2,2)),min(SIV3(i),R(3,2)),min
        (SIV4(i),R(4,2))]);
    b3=max([min(SIV1(i),R(1,3)),min(SIV2(i),R(2,3)),min(SIV3(i),R(3,3)),min
        (SIV4(i),R(4,3))]);
    b4=max([min(SIV1(i),R(1,4)),min(SIV2(i),R(2,4)),min(SIV3(i),R(3,4)),min
        (SIV4(i),R(4,4))]);

    % Penalty Factors
    if B(1)==b1 || b1-B(1)<cut/10
        P1=0;
    else
        P1=P(1);
    end

    if B(2)==b2 || b2-B(2)<cut/10
        P2=0;
    else
        P2=P(2);
    end

    if B(3)==b3 || b3-B(3)<cut/10
        P3=0;
    else
        P3=P(3);
    end

    if B(4)==b4 || b4-B(4)<cut/10
        P4=0;
    else
        P4=P(4);
    end

    % Objective Function (OBJ)
    HSI(i)=abs(B(1)-b1)+abs(B(2)-b2)+abs(B(3)-b3)+abs(B(4)-b4)+P1+P2+P3+P4;

    end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% SORTING THE FUNCTION STAGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[HSI,index]=sort(HSI);
HSImin=HSI(1);

for i=1:Habitat
    SIV1_sorted(i)=SIV1(index(i));
    SIV2_sorted(i)=SIV2(index(i));
    SIV3_sorted(i)=SIV3(index(i));
    SIV4_sorted(i)=SIV4(index(i));
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Elitism STAGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for elitz=1:Elitism
    SIV1_elitism(elitz)=SIV1_sorted(elitz);
    SIV2_elitism(elitz)=SIV2_sorted(elitz);
    SIV3_elitism(elitz)=SIV3_sorted(elitz);
    SIV4_elitism(elitz)=SIV4_sorted(elitz);
    HSI_elitism(elitz)=HSI(elitz);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% OPTIMIZATION STARTING STAGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for G=1:Loop % Starting # of Generation Loops

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Doing Migration between poor & rich Islands

for i=1:Habitat % Taken as it is written in Dan Simon's Program

    for r2=1:Species % for selecting either SIV1, SIV2, SIV3 or SIV4 of a
        recipient island as an immigrant

        if rand<Lambda(i)

            RandomNum=rand*sum(Mu);
            Select=Mu(1);
            SelectIndex=1;

            while (RandomNum>Select) && (SelectIndex<Habitat)
                SelectIndex=SelectIndex+1;
                Select=Select+Mu(SelectIndex);
            end

            r1=round(1+(Species-1)*rand); % for selecting either SIV1, SIV2
                , SIV3 or SIV4 of a source island as an emigrant
```

```
            if (r1==r2)
                if(r2==1)
                SIV1_sorted(i)=SIV1_sorted(SelectIndex);
                elseif(r2==2)
                SIV2_sorted(i)=SIV2_sorted(SelectIndex);
                elseif(r2==3)
                SIV3_sorted(i)=SIV3_sorted(SelectIndex);
                else
                SIV4_sorted(i)=SIV4_sorted(SelectIndex);
                end

            else
                if(r2==1)
                SIV1_sorted(i)=MSF(randi(length(MSF),1,1));
                elseif(r2==2)
                SIV2_sorted(i)=MSF(randi(length(MSF),1,1));
                elseif(r2==3)
                SIV3_sorted(i)=MSF(randi(length(MSF),1,1));
                else
                SIV4_sorted(i)=MSF(randi(length(MSF),1,1));
                end

            end


        else

            if(r2==1)
                SIV1_sorted(i)=SIV1_sorted(i);
            elseif(r2==2)
                SIV2_sorted(i)=SIV2_sorted(i);
            elseif(r2==3)
                SIV3_sorted(i)=SIV3_sorted(i);
            else
                SIV4_sorted(i)=SIV4_sorted(i);
            end


        end

    end

end


if (Mut_Opt==1) % if (Mut_opt==1), then mutation stage is enabled


for i=1:Habitat

    b1=max([min(SIV1_sorted(i),R(1,1)),min(SIV2_sorted(i),R(2,1)),min(
        SIV3_sorted(i),R(3,1)),min(SIV4_sorted(i),R(4,1))]);
    b2=max([min(SIV1_sorted(i),R(1,2)),min(SIV2_sorted(i),R(2,2)),min(
        SIV3_sorted(i),R(3,2)),min(SIV4_sorted(i),R(4,2))]);
```

```
    b3=max([min(SIV1_sorted(i),R(1,3)),min(SIV2_sorted(i),R(2,3)),min(
        SIV3_sorted(i),R(3,3)),min(SIV4_sorted(i),R(4,3))]);
    b4=max([min(SIV1_sorted(i),R(1,4)),min(SIV2_sorted(i),R(2,4)),min(
        SIV3_sorted(i),R(3,4)),min(SIV4_sorted(i),R(4,4))]);

    % Penalty Factors
    if B(1)==b1 || b1-B(1)<cut/10
        P1=0;
    else
        P1=P(1);
    end

    if B(2)==b2 || b2-B(2)<cut/10
        P2=0;
    else
        P2=P(2);
    end

    if B(3)==b3 || b3-B(3)<cut/10
        P3=0;
    else
        P3=P(3);
    end

    if B(4)==b4 || b4-B(4)<cut/10
        P4=0;
    else
        P4=P(4);
    end

    % Objective Function (OBJ)
    HSI(i)=abs(B(1)-b1)+abs(B(2)-b2)+abs(B(3)-b3)+abs(B(4)-b4)+P1+P2+P3+P4;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Re-Sorting the previous sorted values

[HSI,index]=sort(HSI);
HSImin=HSI(1);

for i=1:Habitat
    SIV1_updating(i)=SIV1_sorted(index(i));
    SIV2_updating(i)=SIV2_sorted(index(i));
    SIV3_updating(i)=SIV3_sorted(index(i));
    SIV4_updating(i)=SIV4_sorted(index(i));
end

SIV1_sorted=SIV1_updating;
SIV2_sorted=SIV2_updating;
SIV3_sorted=SIV3_updating;
SIV4_sorted=SIV4_updating;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Doing Mutation Process

for k=MR:Habitat % Usually, 2 instead of 3; so we keep half islands "
    solutions"
```

```
    if rand<Mutation_Rate(k)

        SIV1_sorted(k)=MSF(randi(length(MSF),1,1));
        SIV2_sorted(k)=MSF(randi(length(MSF),1,1));
        SIV3_sorted(k)=MSF(randi(length(MSF),1,1));
        SIV4_sorted(k)=MSF(randi(length(MSF),1,1));

    end

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Re-Sorting the previous sorted values

[HSI,index]=sort(HSI);
HSImin=HSI(1);

for i=1:Habitat
    SIV1_updating(i)=SIV1_sorted(index(i));
    SIV2_updating(i)=SIV2_sorted(index(i));
    SIV3_updating(i)=SIV3_sorted(index(i));
    SIV4_updating(i)=SIV4_sorted(index(i));
end

SIV1_sorted=SIV1_updating;
SIV2_sorted=SIV2_updating;
SIV3_sorted=SIV3_updating;
SIV4_sorted=SIV4_updating;


for e=1:Elitism % to replace last solutions by the best solutions of the
    previous generation
    SIV1_sorted(Habitat-e+1)=SIV1_elitism(e);
    SIV2_sorted(Habitat-e+1)=SIV2_elitism(e);
    SIV3_sorted(Habitat-e+1)=SIV3_elitism(e);
    SIV4_sorted(Habitat-e+1)=SIV4_elitism(e);
end


for i=1:Habitat

    b1=max([min(SIV1_sorted(i),R(1,1)),min(SIV2_sorted(i),R(2,1)),min(
        SIV3_sorted(i),R(3,1)),min(SIV4_sorted(i),R(4,1))]);
    b2=max([min(SIV1_sorted(i),R(1,2)),min(SIV2_sorted(i),R(2,2)),min(
        SIV3_sorted(i),R(3,2)),min(SIV4_sorted(i),R(4,2))]);
    b3=max([min(SIV1_sorted(i),R(1,3)),min(SIV2_sorted(i),R(2,3)),min(
        SIV3_sorted(i),R(3,3)),min(SIV4_sorted(i),R(4,3))]);
    b4=max([min(SIV1_sorted(i),R(1,4)),min(SIV2_sorted(i),R(2,4)),min(
        SIV3_sorted(i),R(3,4)),min(SIV4_sorted(i),R(4,4))]);

    % Penalty Factors
    if B(1)==b1 || b1-B(1)<cut/10
        P1=0;
    else
```

```
        P1=P(1);
    end

    if B(2)==b2 || b2-B(2)<cut/10
        P2=0;
    else
        P2=P(2);
    end

    if B(3)==b3 || b3-B(3)<cut/10
        P3=0;
    else
        P3=P(3);
    end

    if B(4)==b4 || b4-B(4)<cut/10
        P4=0;
    else
        P4=P(4);
    end

    % Objective Function (OBJ)
    HSI(i)=abs(B(1)-b1)+abs(B(2)-b2)+abs(B(3)-b3)+abs(B(4)-b4)+P1+P2+P3+P4;

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Re-Sorting the previous sorted values

[HSI,index]=sort(HSI);
HSImin=HSI(1);

for i=1:Habitat
    SIV1_updating(i)=SIV1_sorted(index(i));
    SIV2_updating(i)=SIV2_sorted(index(i));
    SIV3_updating(i)=SIV3_sorted(index(i));
    SIV4_updating(i)=SIV4_sorted(index(i));
end

SIV1_sorted=SIV1_updating;
SIV2_sorted=SIV2_updating;
SIV3_sorted=SIV3_updating;
SIV4_sorted=SIV4_updating;

%%%

for elitz=1:Elitism
    SIV1_elitism(elitz)=SIV1_sorted(elitz);
    SIV2_elitism(elitz)=SIV2_sorted(elitz);
    SIV3_elitism(elitz)=SIV3_sorted(elitz);
    SIV4_elitism(elitz)=SIV4_sorted(elitz);
    HSI_elitism(elitz)=HSI(elitz);
end

HSI_saved(G)=HSImin;
SIV1_saved(G)=SIV1_sorted(1);
SIV2_saved(G)=SIV2_sorted(1);
```

```
SIV3_saved(G)=SIV3_sorted(1);
SIV4_saved(G)=SIV4_sorted(1);

disp(['The best Generation # ', num2str(G), ' is ', num2str(HSI_saved(end))
    ])

end % THE END OF THE GENERATION LOOPS

time=1:Loop;
plot(time,HSI_saved,'r','LineWidth',2);grid;set(gca,'FontSize',24);xlabel('
    No. of Generations');ylabel('$$f_{\min}$$','interpreter','latex');xlim
    ([1 G]);

[Minimum,index]=min(HSI_saved);
disp(['The optimal fmin = ' num2str(Minimum)])
disp(['The yield value (a1) = ', num2str(SIV1_saved(index))])
disp(['The yield value (a2) = ', num2str(SIV2_saved(index))])
disp(['The yield value (a3) = ', num2str(SIV3_saved(index))])
disp(['The yield value (a4) = ', num2str(SIV4_saved(index))])

b1=max([min(SIV1_saved(index),R(1,1)),min(SIV2_saved(index),R(2,1)),min(
    SIV3_saved(index),R(3,1)),min(SIV4_saved(index),R(4,1))]);
b2=max([min(SIV1_saved(index),R(1,2)),min(SIV2_saved(index),R(2,2)),min(
    SIV3_saved(index),R(3,2)),min(SIV4_saved(index),R(4,2))]);
b3=max([min(SIV1_saved(index),R(1,3)),min(SIV2_saved(index),R(2,3)),min(
    SIV3_saved(index),R(3,3)),min(SIV4_saved(index),R(4,3))]);
b4=max([min(SIV1_saved(index),R(1,4)),min(SIV2_saved(index),R(2,4)),min(
    SIV3_saved(index),R(3,4)),min(SIV4_saved(index),R(4,4))]);

disp(['The estimated b1 = ', num2str(b1)])
disp(['The estimated b2 = ', num2str(b2)])
disp(['The estimated b3 = ', num2str(b3)])
disp(['The estimated b4 = ', num2str(b4)])

toc;
```

The fitness of the above BBO program converges in the following manner:

```
The best Generation # 1 is 0.9
The best Generation # 2 is 0.9
The best Generation # 3 is 0.9
The best Generation # 4 is 0.81
The best Generation # 5 is 0.81
The best Generation # 6 is 0.81
The best Generation # 7 is 0.81
The best Generation # 8 is 0.81
The best Generation # 9 is 0.81
The best Generation # 10 is 0.81
The best Generation # 11 is 0.81
The best Generation # 12 is 0.81
The best Generation # 13 is 0.81
The best Generation # 14 is 0.81
The best Generation # 15 is 0.81
The best Generation # 16 is 0.81
The best Generation # 17 is 0.81
The best Generation # 18 is 0.81
```

```
The best Generation # 19 is 0.81
The best Generation # 20 is 0.81
The best Generation # 21 is 0.81
The best Generation # 22 is 0.81
The best Generation # 23 is 0.81
The best Generation # 24 is 0.77
The best Generation # 25 is 0.77
The best Generation # 26 is 0.77
The best Generation # 27 is 0.75
The best Generation # 28 is 0.75
The best Generation # 29 is 0.75
The best Generation # 30 is 0.75
The best Generation # 31 is 0.75
The best Generation # 32 is 0.75
The best Generation # 33 is 0.71
The best Generation # 34 is 0.71
The best Generation # 35 is 0.71
The best Generation # 36 is 0.71
The best Generation # 37 is 0.71
The best Generation # 38 is 0.71
The best Generation # 39 is 0.71
The best Generation # 40 is 0.71
The best Generation # 41 is 0.71
The best Generation # 42 is 0.71
The best Generation # 43 is 0.71
The best Generation # 44 is 0.71
The best Generation # 45 is 0.71
The best Generation # 46 is 0.71
The best Generation # 47 is 0.71
The best Generation # 48 is 0.71
The best Generation # 49 is 0.71
The best Generation # 50 is 0.71
The best Generation # 51 is 0.71
The best Generation # 52 is 0.71
The best Generation # 53 is 0.71
The best Generation # 54 is 0.71
The best Generation # 55 is 0.71
The best Generation # 56 is 0.71
The best Generation # 57 is 0.71
The best Generation # 58 is 0.71
The best Generation # 59 is 0.71
The best Generation # 60 is 0.71
The best Generation # 61 is 0.71
The best Generation # 62 is 0.71
The best Generation # 63 is 0.71
The best Generation # 64 is 0.71
The best Generation # 65 is 0.71
The best Generation # 66 is 0.71
The best Generation # 67 is 0.71
The best Generation # 68 is 0.71
The best Generation # 69 is 0.71
The best Generation # 70 is 0.71
The best Generation # 71 is 0.71
The best Generation # 72 is 0.71
The best Generation # 73 is 0.71
The best Generation # 74 is 0.71
The best Generation # 75 is 0.71
```

```
The best Generation # 76 is 0.71
The best Generation # 77 is 0.71
The best Generation # 78 is 0.71
The best Generation # 79 is 0.71
The best Generation # 80 is 0.71
The best Generation # 81 is 0.71
The best Generation # 82 is 0.71
The best Generation # 83 is 0.71
The best Generation # 84 is 0.7
The best Generation # 85 is 0.7
The best Generation # 86 is 0.7
The best Generation # 87 is 0.7
The best Generation # 88 is 0.7
The best Generation # 89 is 0.7
The best Generation # 90 is 0.7
The best Generation # 91 is 0.7
The best Generation # 92 is 0.7
The best Generation # 93 is 0.7
The best Generation # 94 is 0.7
The best Generation # 95 is 0.7
The best Generation # 96 is 0.7
The best Generation # 97 is 0.7
The best Generation # 98 is 0.7
The best Generation # 99 is 0.7
The best Generation # 100 is 0.7
The optimal fmin = 0.7
The yield value (a1) = 0.3
The yield value (a2) = 0.3
The yield value (a3) = 0.3
The yield value (a4) = 0.29
The estimated b1 = 0.3
The estimated b2 = 0.3
The estimated b3 = 0.3
The estimated b4 = 0.3
Elapsed time is 0.164619 seconds.
```

References

Ahmed, U., & Saqib, M. (2010). *Optimal Solution of Fuzzy Relation Equations* (Master's thesis, Blekinge Institute of Technology, Blekinge, Sweden). Retrieved from https://www.diva-portal.org/smash/get/diva2:830458/FULLTEXT01.pdf ([Accessed Jul. 16, 2016])

Al-Roomi, A. R., & El-Hawary, M. E. (2016a). Economic Load Dispatch Using Hybrid MpBBO-SQP Algorithm. In X.-S. Yang (Ed.), *Nature-Inspired Computation in Engineering* (1st ed., Vol. 637, p. 217 - 250). Switzerland: Springer International Publishing. doi: 10.1007/978-3-319-30235-5

Al-Roomi, A. R., & El-Hawary, M. E. (2016b). Metropolis Biogeography-Based Optimization. *Information Sciences*, *360*, 73 - 95. Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025516302274 doi: http://dx.doi.org/10.1016/j.ins.2016.03.051

Babuška, R. (1999). *Computational Intelligence in Modeling and Control* (Tech. Rep.). Delft, the Netherlands: Delft Center for Systems and Control, Delft University of Technology. Retrieved from http://www.dcsc.tudelft.nl/~rbabuska/CTU/transp/lecture_notes_ctu.pdf ([Accessed Jun. 02, 2016])

Deb, K. (2010). *Optimization for engineering design: Algorithms and examples* (11th ed.). Prentice-Hall of India.

Dubois, D., & Prade, H. (1980). *Fuzzy sets and systems: Theory and applications* (Vol. 144). Boston, MA: Academic Press.

Dubois, D., & Prade, H. (2000). *Fundamentals of fuzzy sets*. New York: Springer Science+Business Media. doi: 10.1007/978-1-4615-4429-6

Eiben, A., & Smith, J. (2003). *Introduction to evolutionary computing*. Berlin, Heidelberg: Springer Berlin Heidelberg.

El-Hawary, M. E. (2016). *Fuzzy Systems.* Department of Electrical and Computer Engineering, Dalhousie University. (Lecture Notes for ECED 6660)

Kajee-Bagdadi, Z. (2007). *Differential Evolution Algorithms for Constrained Global Optimization* (Master's thesis, University of the Witwatersrand, Johannesburg, South Africa). Retrieved from http://wiredspace.wits.ac.za/bitstream/handle/10539/4733/Thesis.pdf ([Accessed Aug. 15, 2013])

Lee, K. H. (2005). *First course on fuzzy theory and applications*. Berlin, Germany: Springer-Verlag.

Rao, S. S. (2009). *Engineering optimization: Theory and practice* (4th ed.). Hoboken, New Jersey: John Wiley & Sons.

Ross, T. J. (2010). *Fuzzy Logic - With Engineering Applications* (3rd ed.). Chichester: John Wiley & Sons Inc.

Sanchez, E. (1976). Resolution of Composite Fuzzy Relation Equations. *Information and Control*, *30*(1), 38 - 48. Retrieved from http://www.sciencedirect.com/science/article/pii/S0019995876904460 doi: http://dx.doi.org/10.1016/S0019-9958(76)90446-0

Simon, D. (2013). *Evolutionary optimization algorithms : Biologically-inspired and population-based approaches to computer intelligence*. Hoboken, New Jersey: John Wiley & Sons Inc.

Venkataraman, P. (2009). *Applied optimization: with matlab programming* (2nd ed.). Hoboken, N.J: John Wiley & Sons.

Yeniay, Ö. (2005). Penalty Function Methods for Constrained Optimization with Genetic Algorithms. *Mathematical and Computational Applications*, *10*, 45–56.

Yu, X., & Gen, M. (2010). *Introduction to evolutionary algorithms*. London, UK: Springer Science+Business Media.

Zimmermann, H.-J. (1996). *Fuzzy Set Theory—and Its Applications* (3rd ed.). Norwell, Massachusetts: Kluwer Academic Publishers.