Evaluation and application of Fuzzy Differential Evolution approach for benchmark optimization and reservoir operation problems

Dejan Vucetic and Slobodan P. Simonovic

ABSTRACT

The differential evolution (DE) algorithm is a powerful search technique for solving global optimization problems over continuous space. The search initialization for this algorithm is handled stochastically and therefore does not adequately capture vague preliminary knowledge. This paper proposes a novel Fuzzy Differential Evolution (FDE) algorithm, as an alternative approach, where the vague information on the search space can be represented and used to deliver a more focused search. The proposed FDE algorithm utilizes (a) fuzzy numbers to represent vague knowledge and (b) random alpha-cut levels for the search initialization. The alpha-cut intervals created during the initialization are used for fuzzy interval based mutation in successive search iterations. Four benchmark functions are used to demonstrate performance of the new FDE and its practical value. Additionally, the application of the FDE algorithm is illustrated through a reservoir operation case study problem. The new algorithm shows faster convergence in most of these functions.

Key words | differential evolution, fuzzy numbers, fuzzy random variables, reservoir optimization

Dejan Vucetic (corresponding author)
Slobodan P. Simonovic
Department of Civil and Environmental Engineering,
University of Western Ontario,
London,
Ontario,
Canada
E-mail: *dvucetic2@gmail.com*

NOTATION

$X_{i,G}$	population vector
G	generation
D	number of parameters for optimization
NP	number of members in a population
$m_{i,G+1}$	mutated vector
$x_{i,G}$	target vector
F	$\in$ [0, 2] controls the amplification of difference vector
$u_{i,G+1}$	trail vector
CR	crossover constant $\in$ [0,1]
$r1$,$r2$ and $r3$	$\in$ [1, NP] represent randomly chosen indexes, where $r1$ corresponds to the base vector
rand_t	tth evaluation of a uniform random generator number $\in$ [0, 1]
$f(.)$	indicates the objective function that is being optimized
b_U and b_L	upper and lower bound agents for initialization
$A(x)$	fuzzy number
a_1, a_2 and a_3	fuzzy triplet definition parameters
μ	degree of membership
$A^{\alpha}_{i,0}$	fuzzy population vector defined by the fuzzy degree membership equal to α
$A^{L\alpha}_{i,0}$ and $A^{U\alpha}_{i,0}$	lower and upper bound confidence intervals for fuzzy population vector
$m^{\alpha}_{i,G+1}$	mutation fuzzy vector
$m^{L,\alpha}_{i,G+1}$ and $m^{U,\alpha}_{i,G+1}$	lower and upper fuzzy mutation vector confidence interval bounds
f_{DE}	optimal objective function value using differential evolution
f^{s}_{FDE}	optimal objective function value using fuzzy differential evolution where the superscript represents the initialization focus value
f^{i}_{FDE}	initialization objective function value using fuzzy differential evolution

doi: 10.2166/hydro.2013.118

$f^{i}_{\mathrm{DE-SB}}$	initialization objective function value using classic differential evolution with smaller initialization bounds
f^{F}_{FDE}	optimal objective function value using fuzzy differential evolution
$f^{F}_{\mathrm{DE-SB}}$	optimal objective function value using classic differential evolution with smaller initialization bounds
C	maximum physical capacity of the reservoir
R_t	volume of water released in month t
$R_{\max}$	maximum physical capacity for the reservoir outflow structure
$R_{\mathrm{augmented}}$	minimum target release for low flow augmentation
S_t	reservoir storage in month t
$S_{\min}$	minimum allowable storage in reservoir
i_t	inflow volume into reservoir in month t

INTRODUCTION

Optimization is a procedure defined as the selection of a set of decision variables falling within the feasible region of the system that maximize/minimize the objective function (Simonovic 2009). Optimization is very desirable as it improves efficiency, performance, and revenue which finds application in a broad spectrum of fields, most commonly economics, engineering and operations research.

In the very recent past, most optimization practitioners and researchers have been looking for new approaches that combine efficiency and ability to find the global optimum. One group of such optimization algorithms, known as evolutionary algorithms (EA) has received praise for its efficiency and ability to find the global optimum for complex non-linear systems (Back 1996; Simonovic 2009). EA are based on the biological evolutionary process and are therefore inherently stochastic in nature. In this concept, a population of individuals, each representing a search point in the space of feasible solutions, is exposed to a collective learning process, which proceeds from generation to generation. The population is arbitrarily initialized and subjected to the process of selection, recombination/crossover and mutation through stages known as generations, such that newly created generations evolve towards more favorable regions of the search space. The algorithm resembles the Darwinian concept known as 'the survival of the fittest'. This group of algorithms includes, among others, evolution strategies (ES) (Back 1996), differential evolution (DE) (Storn & Price 1995), evolutionary programming (EP) (Fogel *et al.* 1966; Fogel 2006), genetic algorithms (GA) (Holland 1975), and simulated annealing (Kirkpatrick *et al.* 1983; Lockwood & Moore 1993).

One of the above mentioned EA, the DE (Storn & Price 1995, 1997; Storn *et al.* 2005), is the main focus of this paper. It has gained increasing popularity for solving optimization problems due to its robustness, simplicity, easy implementation and fast convergence. DE has been successfully applied to water resource management, mechanical engineering, sensor networks, scheduling and other domains (Storn 1996; Joshi & Sanderson 1999; Rogalsky *et al.* 1999; Ilonen *et al.* 2003; Arunachalam 2008; Onwubolu 2008; Pan *et al.* 2009). Additional examples of DE application in water resource management include maximizing the hydropower benefits for a multipurpose reservoir in India (Regulwar *et al.* 2010) and the work by Reddy & Kumar (2008) in planning and operation of irrigation reservoir systems.

DE utilizes a parallel direct search method for generating population vectors for each generation G from NP, D-dimensioned parameter vectors, where NP is the number of members in a population which is fixed throughout the optimization process and D is the number of optimization parameters, known as individuals. The population vector is given as:

$$X_{i,G},\ i = 1, 2, \ldots, \mathrm{NP} \tag{1}$$

Initialization of the algorithm occurs once the initial vector population is chosen at random from an assumed parameter range (i.e. a range of integers from –10 to 10). Alternatively, if the preliminary solution is known, the population vector is populated using a normally distributed random deviation to the nominal solution, $X_{\mathrm{nom},0}$. The initially generated population ($X_{i,0}$) is perturbed using

mutation and crossover, leading to the evolution of a new trial population. A selection process takes place to determine the fittest population of the two. The fittest individuals (or population vectors) in the population is selected as the initial population for the subsequent generation. This process continues iteratively until a termination condition is met. Figure 1 summarizes the main components of the algorithm.

The initialization strategies currently used with the DE address two specific scenarios: certainty or uncertainty. When preliminary information is available with certainty, the algorithm may be initialized using the nominal solution as discussed (Storn *et al.* 2005). Otherwise, if preliminary information is not available, the initialization will have to rely on a range of possible solutions (Storn *et al.* 2005).

However, when vague preliminary knowledge of the problem domain is available, neither method for initialization is ideal. Using such vague information to assume a nominal solution incorrectly implies more certainty than available. Alternatively, assuming a range of solutions accounts for the uncertainty but may not utilize all available information to represent it correctly. The more knowledge one includes, the less uncertain will be the initialization and, consequently, the optimization.

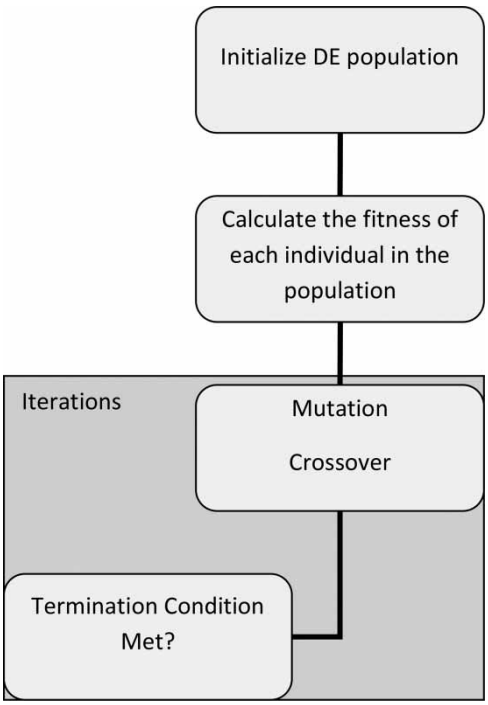


Figure 1 | DE algorithm schematic.

The fuzzy set theory (Zadeh 1965) offers a means to address the quantification of uncertainty from the available vague information. The fuzzy set theory offers unique possibilities for modifications of the traditional fundamentally stochastic DE algorithm. Some fuzzy practitioners have been already involved with evolutionary optimization (Liu & Lampinen 2004). Some have utilized the existing algorithm to develop fuzzy models, like Kisi (2004) who found the parameters of membership functions for daily suspended sediment modeling. Others have joined the ongoing research that has resulted in modifications of the classic DE algorithm, such as Liu & Lampinen (2004). They proposed a fuzzy adaptive parameter control algorithm, based on feedback from the search behavior, to address the sensitivity of the DE to control parameter settings.

The objective of this paper is to create an additional DE initialization strategy that will be able to represent vague knowledge. For example, consider the water resource management problem where allocation decisions are required by a reservoir operator. The operator has certain available information, such as existing operating rules, at his/her disposable to make the appropriate allocation decision but also subjective knowledge (based on past weather, current degree of soil saturation, water level, etc.). This new initialization strategy will therefore allow for a more realistic representation of all available knowledge. The more knowledge that is included, the more likely the optimization will converge, improving algorithm performance. In conjunction with the new initialization technique, the mutation scheme will require modification in order to properly offer valuable guidance to the DE algorithm towards a more focused search.

In this paper, a novel fuzzy differential evolution (FDE) algorithm for initialization and mutation is proposed. This approach conveys priori knowledge for guiding the search towards the optimal solution. A set of standard benchmark functions are used to compare classical DE performance (in terms of convergence speed) with the proposed FDE algorithm. In addition, a reservoir operation case study is used to illustrate the practical application and benefit of the FDE algorithm.

The remainder of this paper is organized as follows. In the following section, we present a methodological overview of the traditional DE algorithm, followed by the methodology of the novel FDE algorithm. Under the section 'FDE algorithm', we introduce four benchmark functions to be used for evaluation of FDE to demonstrate its overall effectiveness as an algorithm. The next section is devoted to the analysis of the benchmark function results. Finally, the last two sections are devoted to the reservoir operation case study, and concluding remarks.

METHODOLOGY

Presented here is the methodological background of the traditional DE and novel FDE algorithm.

DE algorithm overview

The DE algorithm, once initialized, has three main operations before termination: (1) mutation, (2) crossover and (3) selection. The fundamental idea behind DE is a scheme for the generation of trial parameter vectors. Mutation and crossover are used to generate new trial parameter vectors, and then selection determines which of the vectors will survive into the next generation.

The original DE algorithm scheme proposed by Storn & Price (1995, 1997) can be classified using the notation DE/rand/1/bin. Since the original DE/rand/1/bin inception many other variants have been proposed (Storn *et al.* 2005). Reviewed here is the original scheme.

Mutation

The mutation operator is called 'differential mutation' and generates the mutated individual (also known as mutated vector) $m_{i,G+1}$, for the principal parent (also known as target vector) $x_{i,G}$ according to the following equation (Storn & Price 1997):

$$m_{i,G+1} = x_{r1,G} + F\left(x_{r2,G} - x_{r3,G}\right)\ r1 \neq r2 \neq r3 \tag{2}$$

where $F \in [0, 2]$ is a real number that controls the amplification of the difference vector $(x_{r2,G} - x_{r3,G})$, while $r1, r2, r3 \in [1, NP]$ represent randomly chosen indexes, where $r1$ corresponds to the base vector. The indexes have to be different from each other and from the running index i. That way, a parent pool of four individuals is required to breed an offspring.

Crossover

To complement the differential mutation search strategy, DE then uses a crossover operation, in which the mutated individual is mated with the principal parent and generates the offspring or 'trial individual'. This crossover operation for classic DE as reviewed here is known as binomial crossover.

The target vector $x_{i,G}$ is mixed with the mutated vector, $m_{i,G}$, using the following scheme, to yield the trial vector (Storn & Price 1997):

$$u_{i,G+1} = (u_{i,1,G+1}, u_{i,2,G+1}, \ \ldots, \ u_{i,D,G+1}) \tag{3}$$

where

$$u_{i,G+1} = \begin{cases} m_{i,t,G+1} \text{ if } \text{rand}_t < \text{CR or } t = rn_i \\ x_{i,t,G}, \ \text{if } \text{rand}_t > \text{CR and } t \neq rn_i \end{cases} \tag{4}$$

CR is the crossover constant $\in [0, 1]$ (to be specified by the user), $t = 1, 2, \ldots, D$ and rand_t is the tth evaluation of a uniform random generator number $\in [0, 1]$. Lastly, to guarantee that a new altered population vector is produced, a randomly chosen index $rn_i \in [1,2,\ldots,D]$ is used, ensuring that $u_{i,G+1}$ gets at least one element from $m_{i,G+1}$.

Selection

DE uses a selection mechanism to ensure that the individuals promoted to the next generation are strictly those with the best fitness values in the population. A knockout competition is played between each individual (target vector) $x_{i,G}$ and its offspring (trial vector) $u_{i,G+1}$. The survival criteria can be described as follows (Storn & Price 1997):

$$x_{i,G+1} = \begin{cases} u_{i,G+1}, & \text{if } f\left(u_{i,G+1}\right) < f\left(x_{i,G}\right) \\ x_{i,G}, & \text{otherwise} \end{cases} \tag{5}$$

where $f(.)$ indicates the objective function that is being optimized (minimized here). This one-to-one selection mechanism ensures that the selected individuals are strictly those with the best fitness values in the population. That is to say, the trial vector $u_{i,G+1}$ must yield a better fitness value than $x_{i,G}$, for $x_{i,G+1}$ to be set to $u_{i,G+1}$; otherwise, the old value $x_{i,G}$ is retained.

FDE ALGORITHM

The novel FDE algorithm proposed here allows a novel approach for additional problem domain information to be communicated to the DE algorithm for optimization. Doing so results in better overall performance.

DE is fundamentally a stochastic based algorithm. The name FDE may suggest a full deviation to the fuzzy domain. However, this is not the case. The proposed method may be better described as a stochastic and fuzzy hybrid. The (1) initialization and (2) mutation procedures are modified so that they utilize both the fuzzy and the stochastic theory.

Initialization

Initialization is carried out in order to seed the population NP, D-dimensional parameter vector of the algorithm. Traditionally performed through using $\text{rand}_i \in [0, 1]$, a uniform probabilistic distribution to randomly select within upper (b_U) and lower bounds (b_L) agents is to be carried through subsequent algorithm components:

$$x_{i,0} = b_L + \text{rand}_i(b_U - b_L)\ i = 1, 2, \ldots, \text{NP} \tag{6}$$

Instead, in FDE, initialization is carried out by using two fuzzy concepts: (1) a normal continuous-valued fuzzy set characterized by a membership function and (2) alpha-cuts. Membership functions in this case are used to describe the convex single-point normal fuzzy sets defined on the real line, often termed fuzzy numbers (i.e. vague values such as a flow of *about* 5 m^3/s) (Ross 2004). Therefore, the membership functions are used to capture the available knowledge and transfer it to the optimization algorithm. The membership functions and the alpha-cuts are both used to support the initialization step within the optimization algorithm. The use of alpha-cuts allows for the creation of multiple unique population vectors from the singular supplied fuzzy set. Through these fuzzy concepts, the FDE algorithm initialization is able to take advantage of the available domain knowledge, no matter how uncertain.

Membership functions describe the degree of membership or truth in each value corresponding to a parameter. Many shapes of membership functions may be used. In this paper, for illustration and convenience, we are limiting our discussion to the triangular membership function. A fuzzy triangular number $A = (a_1, a_2, a_3)$ can be represented by an ordered triplet or by a triangular membership function:

$$A(x) = \begin{cases} 0, & \text{if } x < a_1 \\ \left(\dfrac{x - a_1}{a_2 - a_1}\right), & \text{if } a_1 \le x \le a_2 \\ \left(\dfrac{a_3 - x}{a_3 - a_2}\right), & \text{if } a_2 \le x \le a_3 \\ 0, & \text{if } x > a_3 \end{cases} \tag{7}$$

Figure 2 shows a triangular membership function defined by Equation (7) where a_2 holds the highest degree of membership in x (membership, $\mu = 1$), comparatively a_1 and a_3 hold no degree of membership ($\mu = 0$). Within the FDE algorithm a_1 and a_3 are called the initial parameter range while a_2 is called the focus or target parameter.

Alpha-cuts are mostly used to extract information from a membership function and are rarely used for defuzzifying the fuzzy sets (converting fuzzy numbers into crisp form). The alpha-cut describes a fuzzy set using a set of sharp sets. The main idea is to fix a certain membership degree α

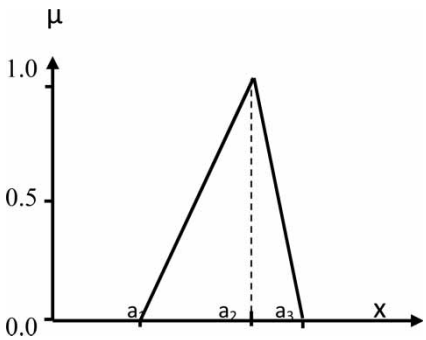


Figure 2 | Triangular fuzzy membership function.

and thus to obtain a crisp set, which is defined as the set of values that have a membership degree μ higher or equal to α. Figure 3 illustrates the concept of alpha-cuts. The membership function is cut horizontally at a finite number of regular α-levels, or cuts, between 0 and 1. This process generates a number of crisp interval sets, as shown in Figure 4.

Taking an arbitrary alpha-cut $\in [0, 1]$ in A (a triangular fuzzy number), a confidence fuzzy interval, A_α is obtained, defined as:

$$A_\alpha = \left[a_1^\alpha, a_3^\alpha\right] = [((a_2 - a_1)\alpha + a_1), (-(a_3 - a_2)\alpha + a_3)] \quad (8)$$

Relating to FDE, parameters are described using triangular fuzzy numbers in the form of inputs for the triangular membership function. To start the algorithm, the initial population vector needs to be generated from these membership functions. This is achieved by using the alpha-cut method NP times at random α-levels to create alpha-cut intervals for each parameter. This allows for a unique individual to be generated NP times from the same parameter membership function input (fuzzy number). The alpha-cut interval is assumed to belong to a unique fuzzy number. In essence, the initial fuzzy number is used to seed NP unique incomplete fuzzy numbers defined only by a single discrete alpha-cut level.

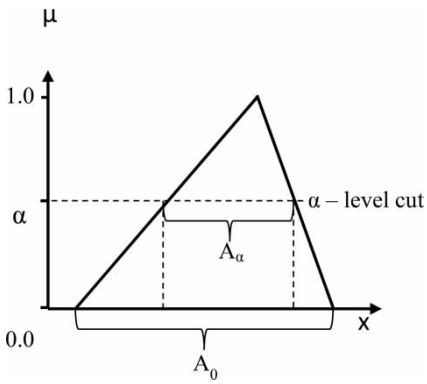


Figure 3 | The alpha-cut method schematic.

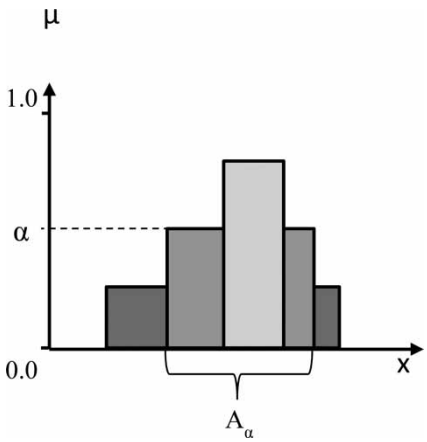


Figure 4 | The alpha-cut intervals schematic.

The alpha-cut interval population vector, $A_{i,0}^\alpha$, is found by modifying Equation (8):

$$\begin{aligned} A_{i,0}^\alpha &= (A_{i,0}^{\mathrm{L}\alpha},\ A_{i,0}^{\mathrm{U}\alpha}) \\ A_{i,0}^{\mathrm{L}\alpha} &= a_1 + \mathrm{rand}_i * (a_2 - a_1) \\ A_{i,0}^{\mathrm{U}\alpha} &= a_3 - \mathrm{rand}_i * (a_3 - a_2) \end{aligned} \quad (9)$$

where $i = 1, 2, \ldots$, NP and α is the alpha-cut level such that it is equal to a uniform random number generated, $\mathrm{rand}_i \in [0,1]_i$. $A_{i,0}^{L\alpha}$ and $A_{i,0}^{U\alpha}$ are the lower and upper interval bounds for each alpha-cut. The parameters a_1, a_2, a_3 are the values representing the fuzzy number triplet for each individual parameter.

In singular value form, the alpha-cut intervals are converted to the familiar population vector where neutral preference is given to the upper and lower intervals

$$x_{i,0} = \frac{1}{2} * (A_{i,0}^{U\alpha} + A_{i,0}^{L\alpha}) \quad (10)$$

In order for a unique singular value to be generated, an asymmetrical triangular membership function must be used.

Mutation

The mutation component of the algorithm allows for new population vectors to be generated in order to investigate the feasible region in search for the optimal solutions. FDE utilizes the alpha-cut intervals from the initialization stage and performs mutation on them by using fuzzy arithmetic. Performing the mutation in the fuzzy domain allows for the algorithm to take advantage of the focused search benefits given by the uncertain or vague available knowledge from the problem domain. The mutation that is

carried out is based on a modification of DE/rand/1/bin, a classical, widely used and successful strategy. Therefore, the full notation for the proposed strategy can be stated as FDE/rand/1/bin. A similar modification to the one presented here could be performed for several other DE variants available, but that is beyond the scope of this paper. DE/rand/1/bin defines the weighted differential of two different randomly chosen vectors and is used to perturb another randomly chosen vector, creating a mutated vector. This process is mathematically expressed in Equation (2).

The mutation vector mathematical expression in Equation (2), transformed using alpha-cut intervals (from initialization and subsequently), has the following form:

$$m_{i,G+1}^{\alpha} = A_{r1,G}^{\alpha_{r1}} + F*\left(A_{r2,G}^{\alpha_{r2}} - A_{r3,G}^{\alpha_{r3}}\right) \; r_1 \neq r_2 \neq r_3 \tag{11}$$

Utilizing fuzzy interval arithmetic properties for addition and subtraction (Bojadziev & Bojadziev 1995):

$$\begin{aligned} A + B &= (a_1^{\alpha}, a_2^{\alpha}) + (b_1^{\alpha}, b_2^{\alpha}) = (a_1^{\alpha} + b_1^{\alpha}), (a_2^{\alpha} + b_2^{\alpha}) \\ A - B &= (a_1^{\alpha}, a_2^{\alpha}) + (b_1^{\alpha}, b_2^{\alpha}) = (a_1^{\alpha} - b_2^{\alpha}), (a_2^{\alpha} - b_1^{\alpha}) \end{aligned} \tag{12}$$

and substituting for Equation (11) yields the lower and upper mutation vector interval bounds:

$$\begin{aligned} m_{i,G+1}^{L,\alpha} &= A_{r1,G}^{\alpha,L} + F*(A_{r2,G}^{\alpha,L} - A_{r3,G}^{\alpha,U}) \; r_1 \neq r_2 \neq r_3 \\ m_{i,G+1}^{U,\alpha} &= A_{r1,G}^{\alpha,U} + F*\left(A_{r2,G}^{\alpha,U} - A_{r3,G}^{\alpha,L}\right) \; r_1 \neq r_2 \neq r_3 \end{aligned} \tag{13}$$

where $i = 1, 2, \ldots, NP$. The alpha-cut population vector interval $A_{i,G}^{\alpha}$, is represented by discrete endpoints $(A_{i,G}^{\alpha,L}, A_{i,G}^{\alpha,U})$ for levels α_{r1}, α_{r2}, α_{r3} These levels may equal to each other or they may be different. However, as seen in Equation (12), the alpha-cut level α must be the same throughout in order to proceed with interval arithmetic. This is likely not the case in the initialization stage where unique alpha-cut intervals are generated.

Each of the alpha-cuts for the purpose of the FDE algorithm represents a unique fuzzy number. These fuzzy numbers are incomplete, because they are defined by a single alpha-cut level (Bojadziev & Bojadziev 1995). In order to perform interval arithmetic at the same alpha-cut level, redefining of incomplete fuzzy numbers is required. Redefining allows incorporating levels not given initially (Bojadziev & Bojadziev 1995).

The mutated alpha-cut intervals vector can be expressed in the traditional singular value form:

$$m_{i,G+1}^{\alpha} = \frac{1}{2} * (m_{i,G}^{L\alpha} + m_{i,G}^{U\alpha}) \tag{14}$$

The above FDE algorithm has been implemented in a convenient decision support system. This decision support system alongside FDE integrates key DE features, such as classical DE variants and the ability to deal with constraints. The decision support system was developed to provide a convenient optimization software package with a graphical user interface for the Windows operating system. Increasing accessibility through this decision support system allows for reaching less technical individuals and allowing them to achieve the practical benefits of optimization.

IMPLEMENTATION OF THE PROPOSED FDE ALGORITHM ON BENCHMARK OPTIMIZATION FUNCTIONS

Benchmark functions, or test functions, are commonly used in order to test optimization procedures (Molga & Smutnicki 2005). The quality of the proposed FDE algorithm is evaluated by comparing it with the original DE algorithm variant – DE/rand/1/bin (simply referred to as DE from this point on) by utilizing well-known benchmark functions from the literature.

The function testbed contains four functions: (1) first De Jong, (2) Rosenbrock's Valley, (3) modified third De Jong, and (4) Rastrigin's function (Back 1996; Molga & Smutnicki 2005). These functions exhibit distinctive difficulties for a global optimization algorithm. For all functions, an initial parameter range, IPR, and focus value were defined. At the beginning of the optimization, initial parameter values are drawn using traditional methodology or FDE initialization.

IPR for FDE and DE is kept consistent for all functions $x \in [-5.12, 5.12]$, while for the case of DE with smaller bounds the IPR is changed to $x \in [-1, 1]$.

The algorithm settings for each test function are given in Table 1; the FDE strategy is compared to DE; user-given controls and computation platform are kept consistent for a fair comparison. Additionally to the results found from keeping the random generator seed constant, 100 independent trials were carried out while keeping the random generator seed value random.

First De Jong function

De Jong is one of the pioneers in evolutionary computation. De Jong's function was originally introduced to evaluate GA and subsequently has been well accepted by the evolutionary optimization community. The First or Sphere De Jong function is one of the simplest problems for optimization algorithms because it does not contain local optima and provides a smooth gradient towards a global optimal solution:

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \tag{15}$$

The global minimum is $f_1(0) = 0$.

Rosenbrock's valley function

Rosenbrock's function is a classical optimization problem used as a performance test for optimization algorithms. The function may be referred to as the second function of De Jong, or Banana function due to its shape.

$$f_2(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \tag{16}$$

Table 1 | Algorithm settings

Method, parameters	Settings for benchmark tests	
Strategy	DE/rand/1/bin	FDE/rand/1/bin
Test problems	Min f(X)	Min f(X)
Generations	2,000	2,000
Mutation factor	0.8	0.8
Crossover factor	0.9	0.9
Number of individuals	10 × D	10 × D
Random generator seed	3	3

Although $f_2(x)$ has just two parameters, it has the reputation of being a difficult minimization problem. The global minimum is $f_2(1) = 0$.

Modified third De Jong function (step)

The step function introduces small plateaus to the topology of an underlying continuous function (Back 1996). Instead of the original linear step function proposed by De Jong is the discretization of a sphere model.

$$f_3(x) = \sum_{i=1}^{D} (x_i + 0.5)^2 \tag{17}$$

The modified step function in Equation (17) exhibits many plateaus which pose a considerable problem for many optimization algorithms as they do not contribute any information on the favorable search direction. The global minimum is $f_3(-0.5) = 0$.

Rastrigin's function

Rastrigin's function is a highly multimodal test function. This function is fairly difficult to optimize due to its large search space and its large number of local minima produced by the cosine modulation. For those reasons, it is frequently selected for testing the performance of various optimization algorithms:

$$f_4(x) = 10D + \sum_{i=1}^{D} [(x_i^2 - 10\cos(2\pi x_i))] \tag{18}$$

The global minimum is $f_4(0) = 0$.

RESULTS AND DISCUSSION

There are several conclusions reached after the comparison of FDE to the original DE strategy using the benchmark functions. First, a comparison of DE and FDE is made by selecting three arbitrary focusing targets. The focus targets are selected as 1, 3 and 5 for the sphere and Rastrigin function while for the step and Rosenbrock function the focus is

selected as –1, 1 and 3. The results are shown in Table 2, for single trials with equal random seeds of 3. In Table 2, the values column lists the dimensions of the problem, D and objective function optimal solution. Where $f(X^*)$ is the known exact solution while f_{DE} and $f_{FDE}^{(focus)}$ are the optimal solutions found through the use of the DE and FDE algorithm, respectively. Complementing the results in Table 2 are the results in Tables 3 and 4 which show the mean and standard deviation of the objective value for 100 independent trials of each scenario. The values in Tables 3 and 4 that are shown in square brackets represent the average generation the objective value was reached if it differed from the standard of 2,000.

Using the same computation platform in all trails; FDE is usually found to be more computationally efficient in arriving at the termination criteria (2,000 generations, or true optimal solution). For First De Jong FDE arrives at the termination criteria on average 52% quicker, for the Rosenbrock function it is found to be 11% quicker, for the Step function 46% quicker for a focus of –1, while on average it is 11% quicker for the other focus targets. Only in the Rastrigin function is there no significant computation time improvements.

The results indicate that FDE performs better than classic DE in terms of convergence speed (a more optimal solution, more computationally quick), independently of the selected target initialization value for the First De Jong and Rastrigin functions and dependent on the selected target initialization value for the Modified Third De Jong function. This can especially be seen in the first 400–500 generations in Table 2, while later on the convergence starts to taper off. This is attributed mostly to the more focused initialization strategy of FDE. Furthermore, the results in Tables 3 and 4 show that the quality of optimal solution improves based on the proximity of the initial focus target value to the true solution. For example, the convergence speed incrementally improves for the First De Jong function (see Table 3) as the subjective focus value approaches the true optimal solution of zero. The magnitude of the optimal solution differences between the varying targets does not directly correlate with the magnitude of the target differences themselves.

In Table 5, the benchmark function optimization comparison is made between DE with drastically smaller bounds (DE-SB) and FDE with focusing target of 1. The values column in Table 5 lists the initial (f^i_{DE}, f^i_{FDE}) and final (f^f_{DE}, f^f_{FDE}) objective function values based on 100 trails through using the DE and FDE algorithm, respectively. Decreasing the initialization bounds in DE-SB and keeping FDE bounds wider shows that FDE performs similarly to or better than traditional strategies, without limiting the search space by implying more certainty than is available. Additionally, the outcomes shown in Table 5 indicate that the improved results using FDE over DE are not just attributed to the better initialization values (due to the more focused smaller initial parameter range) but are affected by the novel mutation strategy as well.

The Rosenbrock function in particular appears to perform worse using the FDE algorithm, then the traditional DE algorithm. It can be seen in Table 2 that the Rosenbrock function using the FDE algorithm appears to stall due to misconvergence while the traditional DE algorithm continues to converge towards the optimal solution. This may be a result of the particular control parameters selected (CR and F) not being adequate for the FDE algorithm when it comes to this particular function, or it could be that the algorithm itself does not cater as well as DE to such a function. In Table 5 at first glance it appears that the performance of the Rosenbrock function using the FDE algorithm is again worse, this may partially be attributed to the DE algorithm giving a significant initial advantage due to the smaller initialization bounds used (based on the initial objective values) as well, FDE may just not be suitable for this particular function to give best performance.

Therefore, with some functions FDE may not perform better than the original DE scheme. This is due to misconvergence or stalling of the algorithm based on the objective function itself and the control parameters selected. FDE shares this robustness problem with many other DE scheme variants. Therefore, care needs to be taken when selecting FDE alongside the control parameters for an objective function to ensure that it is the correct choice in achieving the best convergence efficiency. Currently, as with most other variants, validation of selection may only be confirmed through trial and error procedure. Future research may be directed into sensitivity analysis of FDE to a multitude of benchmark functions, with the purpose of determining the general set of best handled function

Table 2 | Performance comparison of FDE and DE algorithms at various focus targets

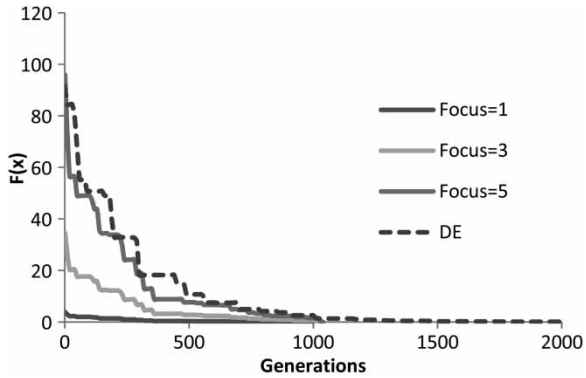
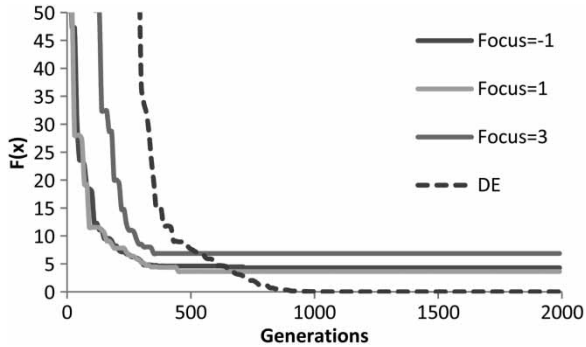
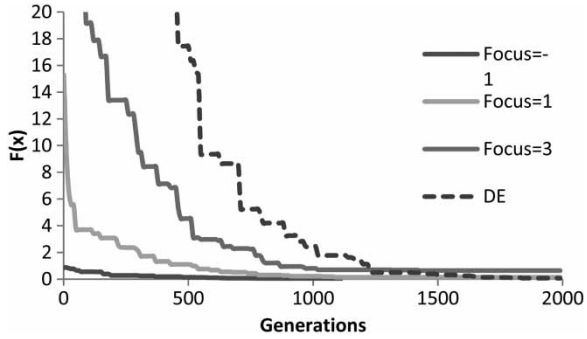
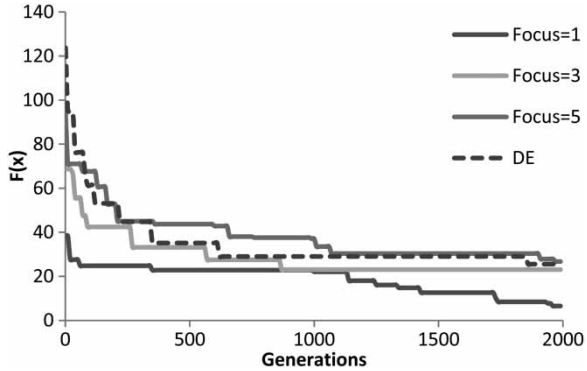
	Comparison of DE and FDE	
Functions	**Values**	**Curves of best solutions**
First De Jong (Sphere)	$f(X^*)$: $f(0) = 0$ $D = 20$ $f_{DE} = 5.68e{-2}$ $f^1_{FDE} = 0$ $f^3_{FDE} = 0$ $f^5_{FDE} = 0$	
Rosenbrock	$f(X^*)$: $f(1) = 0$ $D = 10$ $f_{DE} = 2.099e{-8}$ $f^{-1}_{FDE} = 4.311$ $f^1_{FDE} = 3.638$ $f^3_{FDE} = 6.86$	
Modified Third De Jong (Step)	$f(X^*)$: $f(-0.5) = 0$ $D = 20$ $f_{DE} = 6.141e{-2}$ $f^{-1}_{FDE} = 0$ $f^1_{FDE} = 1.445e{-1}$ $f^3_{FDE} = 6.406e{-1}$	
Rastrigin	$f(X^*)$: $f(0) = 0$ $D = 10$ $f_{DE} = 25.557$ $f^1_{FDE} = 6.543$ $f^3_{FDE} = 23.074$ $f^5_{FDE} = 26.745$	

Table 3 | Summary of 100 optimization trial results

Functions		f_{DE}	f^1_{FDE}	f^3_{FDE}	f^5_{FDE}
First De Jong (Sphere)	f_{mean}	0.063	0 [1,000.9]	0 [0,020.6]	0 [1,036.6]
	f_{sd}	0.015	0 [28.2]	0 [26.1]	0 [28.3]
Rastrigin	f_{mean}	22.785	13.777	21.4	21.761
	f_{sd}	4.799	4.873	5.407	5.258

Table 4 | Summary of 100 optimization trial results

Functions		f_{DE}	f^1_{FDE}	f^{-1}_{FDE}	f^3_{FDE}
Rosenbrock	f_{mean}	7.45e–09	3.839	5.03	6.86
	f_{sd}	4.57e–09	0.845	0.519	0.618
Modified Third De Jong (Step)	f_{mean}	5.92e–2	9.71e–2	0[1,117.8]	0.676
	f_{sd}	1.52e–2	3.52e–2	0[54.45]	0.121

Table 5 | Performance comparison between the original DE algorithm with smaller bounds and FDE with a focus equal to one

Functions		f^i_{DE}	f^i_{FDE}	f^f_{DE}	f^f_{FDE}
First De Jong (Sphere)	f_{mean}	3.36	3.24	2.22e–2	0
	f_{sd}	0.46	0.40	4.64e–4	0
Rosenbrock	f_{mean}	115.15	55.56	2.68e–10	3.84
	f_{sd}	25.05	14.65	1.87e–10	0.84
Modified Third De Jong (Step)	f_{mean}	4.83	13.56	3.17e–3	9.69e–2
	f_{sd}	0.68	1.20	7.55e–4	3.52e–2
Rastrigin	f_{mean}	45.65	45.28	16.49	13.86
	f_{sd}	6.76	7.17	5.11	4.82

types. However, the potential reduction in application capacity does not lessen the undeniable value of the FDE algorithm in being included in the optimization toolbox.

As concluded from the experimental results of the test functions, the addition of the decision maker's supplied inherent knowledge better guides the algorithm, resulting in a faster convergence towards an optimal solution when compared with traditional DE scheme. This is the main benefit of FDE. Alternatively, the decision maker can reduce the initialization bounds in the traditional algorithm to attempt to mimic the focusing achieved by FDE. This method incorrectly implies certainty that the solution is indeed within such bounds where the FDE strategy allows for the benefit of focusing on a certain region while still allowing for a wider search space to account for uncertainty.

In addition to the main benefits, the benchmark functions show that even when compared with decreased initial parameter bounds of DE, FDE may still outperform or perform comparatively to that of DE. While the benchmark function results show instances where the initial search bounds are equal, FDE appears to outperform DE regardless of the focusing target in the search space. In addition, focusing is important in the search for the optimal solution, but even in circumstances where the focusing target is highly inaccurate the algorithm still often performs better than DE.

Case study

The reservoir operation case study presented in this section demonstrates the practical application of the novel FDE algorithm for optimization in the field of water resource management. The main objective of this case study is to show optimization practitioners how to supply the required initialization inputs, particularly the focus targets based on all available information both certain and uncertain in nature, such as a hunch or feel of a reservoir operator based on past trends.

Study area background

This study is focused on the optimization of the operation of the Wildwood reservoir in the Upper Thames River basin. The basin is located in the Great Lakes Region, between Lake Erie and Lake Huron in Southwestern Ontario, Canada (see Figure 5). The watershed encompasses an area of 3,482 square kilometers, with a total population of 485,000 (UTRCA 1993). Most of the basin area is rural except for the larger urban centers of London, Stratford and Woodstock.

Figure 5 | Location of the Upper Thames basin.

Seasonal flooding has historically been a major hazard for the Upper Thames River basin. Typically, flooding occurs in early March during snowmelt and in the summer seasons as a result of extreme rainfall events (UTRCA 1993). In 1937, the city of London experienced a massive flooding event. As a result, this sparked the creation of the Upper Thames River Conservation Authority (UTRCA) in 1947. Since the creation of the UTRCA, three major water management reservoirs were created: Pittock, Wildwood and Fanshawe (see Figure 5).

Among the aforementioned reservoirs, Wildwood was the first major project commissioned by UTRCA in 1948 and finally constructed in 1965. The Wildwood reservoir is located on Trout Creek, upstream of the Town of St. Mary's. The reservoir is designed to control downstream flooding and to increase summer stream flows. The reservoirs also provide a range of recreational opportunities for thousands of people each year. The primary goals of the reservoir include flood control during the snowmelt period and summer storm season, low flow augmentation during the drier summer months from May to October and recreational uses during the summer season. Among these goals, the most important one is flood control. Floods in the basin result from a combination of snowmelt and intense precipitation during December to April. In addition to the primary goals of the reservoir, it is also used for recreational purposes, hydro power generation and by local fisheries.

Wildwood is operated by the Upper Thames Conservation Authority in a coordinated manner with reservoirs at Fanshawe (London) and at Pittock (Woodstock) (UTRCA 2012). This optimizes flood control and low flow augmentation efforts for the North Thames River in St. Mary's and for the Thames River watershed in general. Operating the reservoir involves control of one or more of the three outflow structures. The outflow components include: four large sluice gates, three small valves and concrete baffle walls. The sluice gates are used to provide coarse control of flows from the dam during peak runoff periods. This may include the spring runoff period (March–April) and during the fall and early winter when the soil may be frozen or saturated and thus susceptible to runoff. Otherwise, the valves provide fine control of outflow during the summer and periods of low flows. The valves are located in the core of the dam. As such, they allow for maintenance and discharge of cooler water from the bottom of the reservoir. Concrete baffle walls above the gates provide some automatic control during the early summer months when the reservoir level is at or close to its highest level. Water can spill over the walls when the reservoir rises following summer storms.

Problem definition

A release strategy for the optimal operation of the Wildwood reservoir is required for the year 2010. The year 2010 in this study represents the future so that the available historical 2010 inflow data can be used for problem formulation. The operation of the reservoir must be optimized in order to ensure that the reservoir meets the primary requirements of flood control and low flow augmentation. In addition to the primary goals, the reservoir must be operated keeping in mind constraints put forth by the fisheries industry and recreational reservoir use.

Mathematical formulation

Optimization can be defined as a process searching for an optimal solution that provides a maximum or minimum value of an objective function. Therefore, formulation of the objective function is the most important step in solving an optimization problem.

The objective function is formulated as shown in Equation (19), based on primary flood control operation goals and based on some additional constraint descriptions in Table 4.

$$\text{Minimize } f = \frac{\left(\sum_{t=1}^{3} S_t^{\min} + \sum_{t=5}^{12} S_t^{\min}\right)}{\sum_{t=4} S_t^{\max}} \tag{19}$$

The above is a minimization optimization objective concerning reservoir storage S_t and a $t = 12$ month time horizon. Where $t = 1$ corresponds to January and $t = 12$ to December. It can be seen that the objective function, though globally a minimization problem, has a dual objective for both minimization and maximization. The months requiring minimization of storage ($S^{\min}$) are for the purpose of flood control and furthermore preventing damage as a

result of flood inundation to upstream properties. The maximization of storage (S^{max}) is required by fisheries and hydro power, based on the description given in Table 6. This occurs for the month of April or $t = 4$.

In order to perform the optimization of the proposed objective function, additional equations are required to properly model the Wildwood reservoir system. These equations and their variables are simplifications of the complex real-world system and as such can only approximate the true behavior. The model is defined in the form of constraints of which the continuity constraint is the most important one in that it ensures that the reservoir system is balanced with inflow and release, properly accounting for changes in reservoir storage.

Continuity constraint:

$$S_{t-1} + i_t - R_t = S_t \; t = 1,\ 2,\ \ldots,\ 12 \tag{20}$$

where R_t is the release at the current time step, i_t is the inflow at the current time step, similarly S_t represents the storage at the current time step, while S_{t-1} is the storage in the previous time step. Therefore, in order to utilize the above equation for a 12-month time horizon, the initial reservoir storage S_0 must be given.

Table 6 | Constraints of the Wildwood reservoir (UTRCA 1993)

Categories	Constraint description
Physical constraints	Reservoir maximum capacity $18{,}470 \times 10^3$ m^3 and minimum capacity $2{,}430 \times 10^3$ m^3
Flood control	The release from reservoir should not exceed 10 m^3/s to avoid significant flooding. Release should be less than 3 m^3/s to avoid nuisance flooding at St Mary's golf course
Low flow augmentation	In the months of May to October the release from reservoir should target at least 1.13 m^3/s
Recreation	Wide fluctuations should be avoided, particularly in the summer time
Fisheries	Peak storage should be achieved by the first week of April and then subsequently reduced during spring. The reservoir storage level should remain stable at summer levels until late fall
Hydro power	Peak storage should be achieved by the first week of April

In addition to the continuity constraint, there are release and storage constraints that are governed by the physical capacities of the reservoir given in Table 6.

Subject to release constraints:

$$0 \le R_t \le R_{\max} \; t = 1,\ 2,\ 3,\ 4,\ 11,\ 12 \tag{21}$$

In addition to the reservoir physical release capacity constraints, there is a minimum release constraint for low flow augmentation in the summer months, as detailed in Table 4.

$$R_{\text{augmented}} \le R_t \le R_{\max} \; t = 5,\ 6,\ \ldots,\ 10 \tag{22}$$

where $R_{\max}$ is the maximum physical capacity for the outflow structure (sluice gates, etc.) and $R_{\text{augmented}}$ is the minimum target release for low flow augmentation.

Subject to storage constraints:

$$S_0 \le S_{12} \tag{23}$$

This storage constraint is to ensure that the released storage does not exceed the initially available one.

Storage capacity constraint:

$$S_{\min} \le S_t \le C \; t = 1,\ 2,\ \ldots,\ 12 \tag{24}$$

where $S_{\min}$ is the physical minimum capacity of the reservoir (for structural and mechanical integrity of the dam components) and C is the maximum physical capacity of the reservoir beyond which significant flooding will occur.

The final constraint that appropriately models the reservoir is intended to ensure that the fisheries industry has a stable reservoir level for fishing from late summer to late fall. In other words, the August storage levels (S_8) are maintained.

Fisheries stability constraint:

$$S_8 = S_t \; t = 9,\ 10,\ 11 \tag{25}$$

The FDE algorithm, as with the DE algorithm, cannot directly deal with equality constraints. The penalty function method is used to transform the constrained problem (Equations (20) and (25)) into an unconstrained one by penalizing infeasible solutions, in which a penalty term is

added to the objective function for any violation of the constraints (Chen &Gen 1997).

The additional penalties added to the objective function force the solution to fall into the feasible space after a few generations. This results from solutions that have the penalty added on to the objective failing in order to compete with solutions without penalty in the selection process of DE. It needs to be emphasized that infeasible solutions may not be rejected outright in each generation, as they may provide much more useful information about optimal solution than some feasible solutions. The major concern is how to determine the penalty term so as to strike a balance between keeping some infeasible solutions and rejecting others. An overly low penalty term constant may keep too many infeasible solutions, whereas a very high penalty constant may reject all the solutions preventing the optimization from convergence to an optimal solution.

The DE algorithm is modified to take account of constraint functions using the penalty function method. The fitness function modified for taking account of the penalty function may be expressed as follows (Chen & Gen 1997):

$$\text{eval}(x) = f(x) + p(x) \tag{26}$$

where x represents the parameter vector, $f(x)$ the objective function of the problem and $p(x)$ the penalty function. For an optimization problem, it is required that

$$\begin{aligned} &p(x) = 0 \text{ if } x \text{ is feasible} \\ &p(x) > 0 \text{ for minimization problems} \\ &p(x) < 0 \text{ for maximization problems} \end{aligned} \tag{27}$$

To demonstrate how the function in Equation (26) is formulated, the objective function Equation (19) is added to the penalty of the constraint violation of Equations (20) and (25):

$$\begin{aligned} \text{Minimize eval}(x) = &\frac{\left(\sum_{t=1}^{3} S_t^{\min} + \sum_{t=5}^{12} S_t^{\min}\right)}{\sum_{t=4} S_t^{\max}} \\ &+ p_1(|S_{t-1} + i_t - R_t - S_t|) + p_1(|S_8 - S_9|) + p_1(|S_8 - S_{10}|) \\ &+ \ p_1(|S_8 - S_{11}|) \ t = 1, \ 2, \ \ldots, \ 12 \end{aligned} \tag{28}$$

For the sake of convenience, just one penalty constant (p_1) was used for all the constraints, resulting in more relaxed but still adequate constraint representation.

Algorithm and optimization inputs

Having formulated the Wildwood reservoir optimization problem, the FDE algorithm inputs must be assigned. Given in Table 7 are the control parameter inputs for the FDE algorithm itself. These values were subjectively chosen using trial and error, as they produce best results for the problem formulation. In addition to FDE, the classical DE/rand/1/bin strategy is used with the same inputs for comparison.

There are 24 decision variables in the mathematical formulation, divided evenly between variables for release and storage. In order for the optimization algorithm to proceed, these decision variables/parameters must be initialized. In order for initialization to take place, the parameter range and target (focus) values must be established. This may be done by utilizing a decision maker's inherent knowledge to establish the parameter bounds. In this case the knowledge was extracted from historical data provided by UTRCA for the period 1985–2011. The parameter range, or the upper and lower bounds for each parameter, were determined by analyzing the monthly historical data and selecting the maximum and minimum values within the data set. Thus, the feasible range for release and storage is established without the need for subjective decision maker inputs. In practice, however, the process is not so easy for the selection of the target or focus for each parameter. The goal of our optimization problem is, in essence, to find future operation optimal release and subsequent storage strategies. To do this, we therefore must establish a subjective target for the release and storage that is believed to be an adequate representation of where the optimum would be. To establish such a target for each parameter, subjective (and likely vague) decision maker knowledge is required.

Table 7 | DE algorithm inputs

Number of generations	1,500
Number of individuals, NP	50
Mutation factor, F	0.8
Crossover factor, CR	0.9
Strategy	DE/rand/1 & FDE/rand/1

Typically, forecasting information from several sources is used to operate the Wildwood Dam. Computer models of floods, operating tables, weather data and water level information from above and below the dam enable staff to assess and respond to flood potential. In practice, combing these existing methods for operation could establish the subjective target values required for initialization of the optimization algorithm. In this case study we had available historical data of storage and release; based on these values we could choose an appropriate target. Conveniently, since we already had operational data for the year 2010, we could use these values as the basis for our targets.

Tables 8 and 9 show the storage and release initialization inputs including parameter range and target values for the year 2010. When using the classical DE algorithm, the same initialization parameter range was used as for FDE.

In addition to the initialization inputs given, feasible space constraints and inflow inputs were required. The release constraints given in Table 6 are converted to corresponding monthly equivalent values for convenience in Table 10. The monthly inflow data for the Wildwood reservoir were provided by UTRCA and are given in Table 11.

Storage constraints corresponding to physical reservoir capacity and minimum storage:

$$S_{\min} = 2,430 \times 10^3\ \mathrm{m}^3$$

$$C = 18,470 \times 10^3\ \mathrm{m}^3$$

Initial Storage (Storage amount in last month of previous year, 2009), $S_0 = 6,564 \times 10^3\ \mathrm{m}^3$.

Study results and discussions

The optimization results of combining the mathematical formulation with the algorithm and optimization inputs are presented in this section. Two optimization trials were performed; one using the classic DE strategy and the other using the novel FDE strategy. The parameter ranges for initialization were kept constant throughout all the trials.

Table 8 | Storage initialization inputs for the year 2010 [10^3 m^3]

Month	1	2	3	4	5	6	7	8	9	10	11	12
Lower bound	2,417	2,930	5,278	12,856	13,939	13,426	12,561	11,038	8,764	4,986	2,790	3,081
Upper bound	9,626	10,399	15,618	17,685	18,354	18,300	17,499	16,434	14,463	13,420	10,836	9,492
Target	6,908	6,610	10,090	15,359	17,516	17,860	17,194	15,523	11,660	8,449	4,222	4,039

Table 9 | Release initialization inputs for the year 2010 [10^3 m^3]

Month	1	2	3	4	5	6	7	8	9	10	11	12
Lower Bound	2,605	10,147	1,426	1,743	1,714	1,607	2,000	2,426	2,766	2,807	2,677	1,423
Upper bound	16,364	1,987	12,961	12,514	11,204	8,328	14,530	8,057	8,615	13,491	19,336	16,382
Target	4,345	3,463	2,387	1,763	1,966	4,596	3,180	4,719	5,786	8,511	6,153	4,940

Table 10 | Release constraints [10^3 m^3]

Month	1	2	3	4	5	6	7	8	9	10	11	12
Max	26,784	24,192	26,784	25,920	26,784	25,920	26,784	26,784	25,920	26,784	25,920	26,784
Min	0	0	0	0	3,027	2,929	3,027	3,027	2,929	3,027	0	0

Table 11 | Monthly inflows for the Wildwood reservoir [10^3 m^3]

Month	1	2	3	4	5	6	7	8	9	10	11	12
Inflow	4,187	2,656	9,419	4,901	3,350	4,433	2,421	1,413	2,617	4,819	4,341	4,859

Figure 6 shows the convergence speed of the objective function for each of the trials. As expected, FDE performed much better than traditional DE, especially in the early generations later on the convergence rate seems to taper and be surpassed by that of traditional DE.

The Wildwood optimal reservoir storage and release policy for the year 2010 is shown in Figures 7 and 8, respectively. The results follow the problem formulation closely. The storage for the month of April is indeed maximized, while the late summer to fall storage is indeed kept consistent. Similarly, the release policy meets the minimum release requirement for low flow augmentation. Thus, the optimization can be deemed satisfactory.

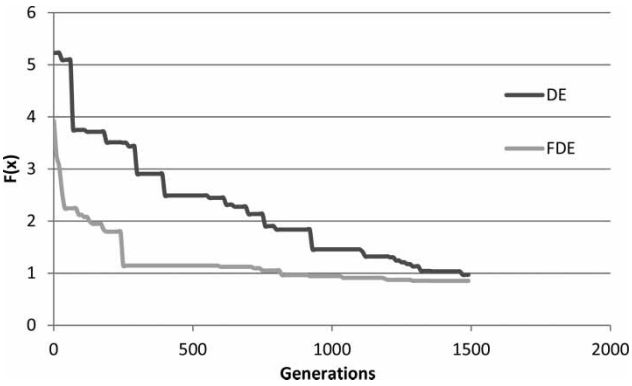


Figure 6 | Wildwood reservoir optimization progress.

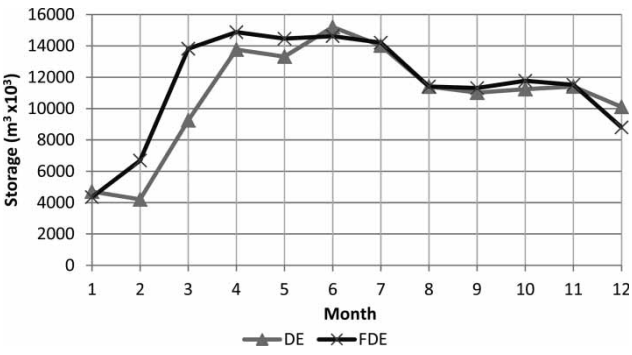


Figure 7 | Wildwood reservoir storage for a 12 month time horizon.

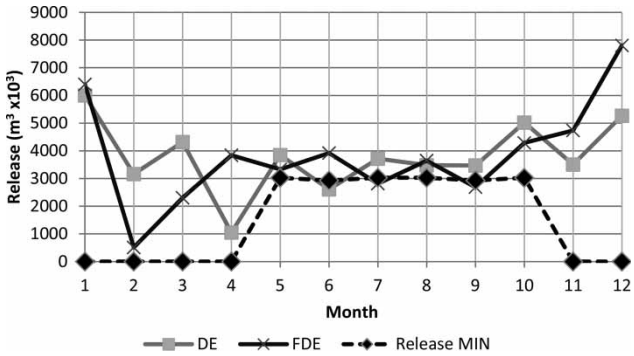


Figure 8 | Wildwood reservoir release for a 12 month time horizon.

CONCLUSION

This paper proposes a novel method, FDE algorithm, which utilizes fuzzy triangular membership functions for initialization combined with random alpha-cuts to create alpha-cut intervals to be perturbed through mutation by fuzzy interval arithmetic. This approach through the utilization of fuzzy theory concepts takes advantage of uncertain available knowledge. The FDE algorithm has flexibility in being used for a wide range of linear and non-linear optimization problems. The novel algorithm with fuzzy set theory elements allows for decision makers to give supplementary knowledge for defining a more focused search space.

Emphasis is placed on the fact that the FDE algorithm, just like all EA, does not guarantee an optimal solution to be found. Furthermore, misconvergence may result using FDE in certain instances as seen in the test function experimental results. Therefore FDE may not be better than DE in the absolute sense, but it does provide an alternative to be used where more knowledge is available to provide a more efficient faster convergence on a function case by case basis. In addition, using this approach gives more freedom in expressing available knowledge without incorrectly claiming full certainty or uncertainty because of the limitation of the algorithm itself.

The FDE algorithms value to the water resource management field was shown through the Wildwood reservoir case study, particularly demonstrating how initialization inputs may be established. The addition of subjective targets for initialization with FDE led to a focused search, ultimately resulting in FDE outperforming the traditional DE algorithm in the convergence towards the optimal solution.

REFERENCES

Arunachalam, V. 2008 Optimization Using Differential Evolution. Water Resources Research Report no. 060, Facility for Intelligent Decision Support, London, Canada, 42 pp.

Back, T. 1996 *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, New York.

Bojadziev, G. & Bojadziev, M. 1995 *Fuzzy Sets, Fuzzy Logic, Applications*. World Scientific, Singapore.

Chen, R. & Gen, M. 1997 *Genetic Algorithms and Engineering Design*. John Wiley, New York.

Fogel, D. B. 2006 *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3rd edn. John Wiley and Sons, New Jersey, 296.

Fogel, L. J., Owens, A. J. & Walsh, M. J. 1966 *Artificial Intelligence Through Simulated Evolution*. John Wiley, Chichester.

Holland, J. H. 1975 *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Ilonen, J., Kamarainen, J. K. & Lampinen, J. 2003 Differential evolution training algorithm for feed-forward neural networks. *Nat. Proc. Lett.* **17**, 95–105.

Joshi, R. & Sanderson, A. C. 1999 Minimal representation multi sensor fusion using differential evolution. *IEEE Trans. Syst. Man Cybernet. A.* **29**, 63–76.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. 1983 Optimization by simulated annealing. *Science* **220**, 671–680.

Kisi, O. 2004 Daily suspended sediment modeling using a fuzzy-differential evolution approach. *Hydrol. Sci. J.* **49**, 183–197.

Liu, J. & Lampinen, J. 2004 A fuzzy adaptive differential evolution algorithm. *Soft Comput. Fusion Found. Methodol.* **9**, 448–462.

Lockwood, C. & Moore, T. 1993 Harvesting scheduling with spatial constraints: a simulated annealing approach. *Can. J. Forest Res.* **23**, 468–478.

Molga, M. & Smutnicki, C. 2005 Test functions for optimization needs. Available from: www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf (last accessed: 10 October 2011).

Onwubolu, G. C. 2008 Design of hybrid differential evolution and group method of data handling networks for modeling and prediction. *Inf. Sci.* **178**, 3616–3634.

Pan, Q. K., Wang, L. & Qian, B. 2009 A novel differential evolution algorithm for bi-criteria no wait flow shop scheduling problems. *Comput. Op. Res.* **36**, 2498–2511.

Reddy, J. & Kumar, N. 2008 Evolving strategies for crop planning and operation of irrigation reservoir system using multi-objective differential evolution. *Irrig. Sci.* **26** (2), 177–190.

Regulwar, D., Choudhari, S. & Raj, P. 2010 Differential evolution algorithm with application to optimal operation of multipurpose reservoir. *J. Water Resour. Protect.* **2** (6), 560–568.

Rogalsky, T., Derksen, R. W. & Kocabiyik, S. 1999 Differential evolution in aerodynamic optimization. *Proceedings of 46th Annual Conference of Canadian Aeronautics and Space Institute*. Montreal, Quebec, pp. 29–36.

Ross, T. J. 2004 *Fuzzy Logic with Engineering Applications*. John Wiley and Sons, UK.

Simonovic, S. P. 2009 *Managing Water Resources: Methods and Tools for a Systems Approach*. UNESCO, Paris and Earthscan James & James, London.

Storn, R. 1996 On the usage of differential evolution for function optimization. Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS), Berkeley, pp. 519–523.

Storn, R. & Price, K. 1995 Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012. Berkeley, CA.

Storn, R. & Price, K. 1997 Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **11**, 341–359.

Storn, R., Price, K. & Lampinen, J. 2005 *Differential Evolution: A Practical Approach to Global Optimization*. Springer, New York.

UTRCA 1993 Low Flow Hydrology and Operations Optimization Study. Wildwood and Pittock Reservoirs. Final Report.

UTRCA 2012 'Wildwood Dam'. Retrieved January 22 2012 from www.thamesriver.on.ca/Water_Management/Wildwood_Dam.htm.

Zadeh, L. A. 1965 Fuzzy sets. *Inf. Control.* **8**, 338–358.

First received 11 July 2012; accepted in revised form 16 April 2013. Available online 16 May 2013