Chemometrics and Intelligent Laboratory Systems 199 (2020) 103955

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemometrics

Using Differential Evolution to design optimal experiments

Zack Stokes [a,*], Abhyuday Mandal [b], Weng Kee Wong [c]

[a] Department of Statistics, University of California, Los Angeles, Los Angeles, CA, 90095, USA
[b] Department of Statistics, University of Georgia, Athens, GA, 30602, USA
[c] Department of Biostatistics University of California, Los Angeles, Los Angeles, CA, 90095, USA

ARTICLE INFO

Keywords:
D-optimality
Evolutionary algorithms
Experimental design
Mixture experiments
Reaction rates

ABSTRACT

Differential Evolution (DE) has become one of the leading metaheuristics in the class of Evolutionary Algorithms, which consists of methods that operate off of survival-of-the-fittest principles. This general purpose optimization algorithm is viewed as an improvement over Genetic Algorithms, which are widely used to find solutions to chemometric problems. Using straightforward vector operations and random draws, DE can provide fast, efficient optimization of any real, vector-valued function. This article reviews the basic algorithm and a few of its modifications with various enhancements. We provide guidance for practitioners, discuss implementation issues and give illustrative applications of DE with the corresponding R codes to find different types of optimal designs for various statistical models in chemometrics that involve the Arrhenius equation, reaction rates, concentration measures and chemical mixtures.

1. Introduction

Nature-inspired, metaheuristic approaches to solve all kinds of optimization problems have skyrocketed in the last few decades, especially in engineering and computer science [1,2]. Some key reasons for their popularity are their speed, simplicity, flexibility, availability of computer codes and ease of implementation. Additional compelling reasons for their widespread use are: (a) they do not require any assumptions, so they can be applied to solve very different types of optimization problems, including those in which the objective function is non-differentiable, multi-modal, multi-objective or high dimensional, and (b) they tend to provide exact or approximate solutions of high quality to complicated optimization problems even though there is often no rigorous proof of convergence.

As the name suggests, nature-inspired metaheuristic algorithms are motivated by the behavior of animals or other natural processes. Some examples of popular nature-inspired metaheuristic algorithms are Genetic Algorithms [3], Cuckoo Search [4], Ant Colony Optimization [5], Grey Wolf Optimization [6], Jumping Frogs Optimization [7], Bat Algorithm [8], and Particle Swarm Optimization (PSO) [9], among many others. While all being in the same general family of evolutionary algorithms, each of these methods operates with a different number of tuning parameters and behavioral characteristics. Naturally, some tend to perform better than others in selected situations [1,2]. provide an overview of the development of nature-inspired metaheuristic algorithms and show systematically how they now dominate the optimization literature for solving real world problems.

In the field of metaheuristic algorithms, there are two primary classes of algorithms: swarm intelligence and genetic representations. The former includes methods in which a group of agents work together to explore the search space and share information as necessary. While many of the algorithms listed above fall into this category, we will instead focus on the latter in this work. Methods in the sub-class of genetic representations involve a set of agents, each made up of chromosomes and genes, in an evolutionary, survival-of-the-fittest battle to be part of the best generation. Fraser appeared to be the first to study candidate solutions in the framework of genetic representations [10]. From this framework, two main algorithms have been developed. Holland established the original Genetic Algorithm (GA) in 1975, and more recently as computational power has grown, Price and Storn created Differential Evolution (DE) in 1997 [11,12]. Since then, many extensions have been proposed to improve their performance in different situations [13–15].

The primary difference between these two algorithms is that GA is used to search exclusively over discrete spaces while DE can also be used over any continuous space. Current optimization literature suggests that in many situations, DE produces better, more stable solutions than GA [16,17]. DE employs only vector operations and random number generators, allowing it to compete more effectively with the speed of the

* Corresponding author.
E-mail address: zstokes@ucla.edu (Z. Stokes).

https://doi.org/10.1016/j.chemolab.2020.103955
Received 8 August 2019; Received in revised form 3 December 2019; Accepted 22 January 2020
Available online 28 January 2020

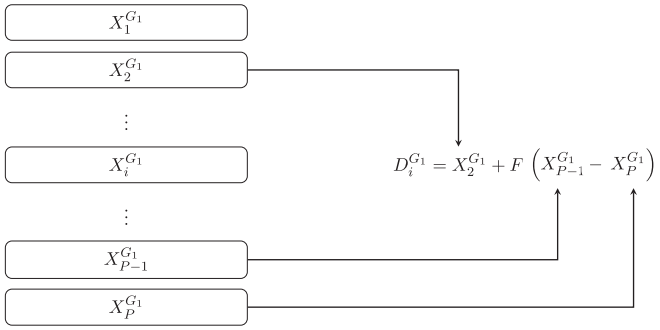

Fig. 1. An illustration of mutation for a single agent in the Differential Evolution algorithm where a donor vector $D_i^{G_1}$ is created by blending 3 randomly drawn agents. In this case $X_2^{G_1}$, $X_{P-1}^{G_1}$ and $X_P^{G_1}$ were chosen.

simpler, yet limited, GA. Additionally, DE has been shown to work well for complex problems, such as solving multiple-objective problems, finding clustering weights, and training neural networks [15]. However, despite these advantages, GA is still more well known and popular than DE in many disciplines. For example, in chemometrics, there are well over 100 recently published articles that use GA as of June 2019, but only a handful that apply DE, see Refs. [18–23]. These few articles do not provide details and implementation issues for DE or its many variants. Interestingly, DE is also rarely used for optimization in the statistical literature. To date, there are only a handful of papers in statistical journals, that use DE as a general tool for solving various types of optimization problems [24–26].

The aim of this paper is to describe DE in more detail and show the usefulness and versatility of DE as yet another powerful tool that requires few, if any, assumptions to solve complex optimization problems. In what follows, we review the fundamental concepts of DE and demonstrate how it solves optimization problems in chemometrics through the design of efficient chemical experiments. We do not compare the performance of DE with its competitors, such as PSO or Cuckoo search, since several papers, including [27], have reported that DE outperforms several other nature-inspired metaheuristics.

In the next section, we describe the key components of DE, the various steps of the algorithm and both the empirical and theoretically-motivated rules for tuning its parameters. Section 3 briefly reviews optimal design methodology and Section 4 demonstrates how DE can be applied to find optimal designs for various linear and non-linear models. Section 5 concludes with a discussion on recent enhancements of the basic DE. Some of these improvements are targeted for solving specific types of optimization problems and others are more on the algorithm itself, such as more effective ways for finding suitable tuning parameters. Codes for generating the optimal designs discussed in this paper can be found in the supplementary materials.

2. DE algorithm and developments

Differential Evolution is a general purpose evolutionary algorithm and was proposed by Rainer Storn and Kenneth Price in 1997 as a means of quickly optimizing functions that do not necessarily have nice properties, like differentiability or continuity, which are required for many standard optimization procedures [11,28]. The goal of DE is to find the optimal solution efficiently and to do so in an easy-to-implement way. Below we present the standard formulation of DE and then discuss additional considerations for choosing the key features that determine its computational cost and convergence.

2.1. Algorithm overview

Without loss of generality, suppose we want to minimize a real-valued function with V variables $h : \mathbb{R}^V \to \mathbb{R}$ by finding $x^* \in \mathbb{R}^V$ such that $h(x^*) \leq h(x)$ for all $x \in \mathbb{R}^V$. The search space of candidate solutions is defined by the limits of each of the V variables and constitute the landscape of the fitness function. These limits may be specified naturally by the application or selected by the experimenter.

Having defined the problem, the next phase is to initialize the DE algorithm. The first step is to choose the number of candidate solutions per generation, P, also known as the population size. Each solution is represented by a vector of length V, so each generation of candidate solutions has dimension $V \times P$. In addition to the population size, there are two parameters F and CR to be selected. We defer discussing the roles and bounds of these parameters but note that there are several approaches for choosing their values. The final step in the initialization process is to specify a stopping rule or condition. In our case we consider a stopping condition defined by a maximum number of generations G_{max}. Once these preliminary steps are complete, the five basic steps for the DE algorithm are as follows:

1 **Genetic Representation:** The initial population must be of size $P > 4$ to ensure that there is enough genetic diversity. Members in the population are candidate solutions of the optimization problem and are called agents. Each is represented by a vector of length V and labeled as $X_1^{G_1}, X_2^{G_1}, \ldots, X_P^{G_1}$. These comprise the first generation of solutions. The initial value for each entry in each agent is randomly chosen over the interval specified for the particular variable.

2 **Mutation:** Much like GA, a mutation process helps to expand the search space of DE. For each target vector $X_i^{G_1}$, this mutation produces a "donor" vector $D_i^{G_1}$ by adding the weighted difference of two agents to a third, all randomly chosen and distinct from the target. For this process the weighting factor F is chosen on the interval [0,2]. Thus, with this constant, P vectors are created according to $D_i^{G_1} = X_{r_0}^{G_1} + F(X_{r_1}^{G_1} - X_{r_2}^{G_1})$, where $r_0 \neq r_1 \neq r_2 \neq i$. $X_{r_0}^{G_1}$ is referred to as the base vector used for generating donor vector $D_i^{G_1}$. Depending on the bounds and choice of F this process may in some cases lead to values that fall outside of the acceptable region for each dimension. There are many strategies in the general optimization literature for dealing with this problem, but we do not explicitly discuss them here.(See Fig. 1)

3 **Crossover:** Crossover blends the current generation of agents with the population of donor vectors in order to form candidates for the next generation known as "trial" vectors. This process differs from the crossover mechanisms typically used in GA in that a decision is made for each element of the vector and not at a few defined points. In DE this technique requires the crossover constant CR, chosen from [0,1]. For each i from 1 to P, one of the V elements of $D_i^{G_1}$ is randomly selected to directly enter the trial vector $T_i^{G_1}$. In this way one variable is forced to change so that each $T_i^{G_1}$ will certainly differ from its original target vector. Next, with probability CR more elements are taken from $D_i^{G_1}$ and placed in the trial vector. Whichever variables do not take their value from the donor vector inherit their original value from $X_i^{G_1}$.(See Fig. 2) Assuming variable j is randomly chosen to take its value from the trial this process is driven by the following equation:

$$T_i^{G_1} = \begin{cases} D_i^{G_1} & \text{for } i = j, \\ D_i^{G_1} \quad \text{with probability} \quad CR, & \text{for } i \neq j, \\ X_i^{G_1} \quad \text{with probability} \quad 1 - CR, & \text{for } i \neq j. \end{cases} \tag{1}$$

4 **Selection:** Selection creates the next generation of agents by comparing each target vector to its respective trial vector. Whichever is measured to be the most fit using h becomes $X_i^{G_2}$. In the case of minimization this process is given by

$$X_i^{G_2} = \begin{cases} T_i^{G_1} & \text{if} \quad h\left(T_i^{G_1}\right) < h\left(X_i^{G_1}\right), \\ X_i^{G_1} & \text{otherwise.} \end{cases} \tag{2}$$

5. **Repeat:** Repeat steps 2 through 4 over many generations until G_{max} is reached or another specified stopping condition is satisfied.

These 5 steps are summarized in Algorithm 1.

Algorithm 1. Pseudocode for the standard DE algorithm.

Algorithm 1: Pseudocode for the standard DE algorithm.

```
initialization;
while iter≤ G_max do
    for (i = 0; i < P; i++) do
        r_0 = floor(rand(0,1)×P); while (r_0 == i);
        r_1 = floor(rand(0,1)×P); while (r_1 == r_0 or r_1 == i);
        r_2 = floor(rand(0,1)×P); while (r_2 == r_1 or r_2 == r_0 or r_2 == i);
        J = floor(V×rand(0,1))
        for (j = 0; j < V; j++) do
            if rand(0,1) ≤ CR or j == J then
                T_ij = X_r0j + F × (X_r1j − X_r2j);
            else
                T_ij = X_ij;
            end
        end
    end
    for (i = 0; i < P; i++) do
        ensure that each T_i is within the search region;
        if (h(T_i) ≤ h(X_i)) then
            X_i = T_i
        end
    end
    iter++;
end
```

2.2. Parameter tuning

The DE algorithm has only 3 tuning parameters, P, F, and CR, which is fewer than many other metaheuristic algorithms, but their values play a large role in the convergence of the algorithm. Improper selection of their values could cause the algorithm to become stuck in a local optimum if the agents do not efficiently explore the space. The process for tuning each parameter is in part driven by the application of interest and prior knowledge of the function landscape. However, it is also important to consider whether any modifications should be made to the standard algorithm and how these may affect the selection of parameter values. In this section we describe some of the adaptations at each step in basic DE and discuss general guidelines for tuning each parameter. For a full study of the individual parameters and choices for other adjustable features of DE, see Ref. [12]. There are also several variants of DE that change the underlying process and in some cases add additional parameters or change the usage of the existing ones. We review some of the most substantial of these variants and hybrid algorithms in Section 5.

In the initialization phase, an important choice is the number of agents P to be used per generation. The search space itself is usually determined by the application of interest, but the value of P must be carefully considered to enhance performance. In setting this parameter it is important to strike a balance between having enough points to explore the space and the computational time of the program. For example, in a space where $P = 6$ gives sufficient new information in each generation for the algorithm to converge in 100 generations, choosing $P = 12$ may not provide any more information per generation but will double the time complexity of each one. If the fitness function is computationally difficult this could result in a drastic slow-down of the method. In fact, in many instances in which the stopping criteria is a fixed number of generations G_{max} it is better to fix an appropriate number of allowable function evaluations and maintain that $P \times G_{max}$ does not exceed this number for all choices of P. This approach necessitates balancing between P and G_{max} to achieve the best result. Beyond this, a general rule of thumb is that P should be at least 10 times the number of function inputs V to ensure sufficient diversity [12].

The mutation factor F is arguably the most sensitive DE parameter. Its value will determine the trade-off between exploration and exploitation of the search space, meaning that a large mutation factor will move points quickly across the space towards the general direction of the global optimum, but it will be difficult to make the smaller, exact steps necessary to reach a precise value. There are many solutions available to this problem including altering the underlying mutation procedure to one that is better suited to the search space of interest, dithering (using a new F for each agent), jittering (using a new F for each variable), and implementing a self-adaptive DE variant that chooses a new mutation strategy and parameter value for each agent. Such methods could be adjusted to give an annealing effect to F that shrinks it as the agents get closer to the optimum so that they can make more precise steps. As far as general rules for selecting F [29], found a lower bound based on the values of P and CR such that choosing a value lower than that will cause the diversity of each generation to decrease as the total number of generations increases. On the other hand empirical evidence has indicated that values chosen between 0.6 and 0.9 will perform well for general optimization problems [12].

The crossover constant CR is a measure of the rate of mutation of the population. Even in cases where an appropriate F has been chosen, an inappropriate CR will prevent the agents from evolving at the right pace for escaping local optima. There are many procedures for altering the crossover strategy to better accommodate the search space and also self-adaptive procedures similar to those for the mutation factor such as the one presented in Ref. [30]. If the standard selection procedure described in Section 2.1 is used, then the value of CR is naturally bounded between 0 and 1. However, the original authors of DE discovered through empirical study that the appropriate range of values is actually limited to $CR \in \{[0, 0.3] \cup [0.8, 1]\}$. It was found that small values of CR lead to convergence in cases where the fitness function can be rewritten as the sum of single variable optimization problems (i.e. separable) and larger values are used if this is not possible [12].

In addition to the tuning parameters, the fitness function and the stopping condition determine the total amount of time the algorithm will run. The more complex the function to be optimized, in terms of magnitude, order, non-linearity, discontinuity, etc., the longer it will take to reach the optimum. Restricting the search region as much as possible can alleviate some of the costs of evaluating a complex fitness function many times. The stopping condition is user-specified and often based around a maximal number of iterations or function evaluations. Other popular options are computational time limits, convergence measures to test for small changes between generations, and stopping values if the algorithm is being used for benchmarking. The selection of a stopping condition is driven mainly by the application and the computational resources available.

3. An overview of optimal design methodology

Before showing how DE can be usefully applied to solve optimal design problems, we first briefly review background for constructing a model-based optimal design given a statistical model and an objective. Throughout we assume that there are resources to collect a set of N predetermined observations for the study and that the assumed statistical model takes the form

$$y_i = f(x_i, \beta) + \varepsilon_i, \quad i = 1, \ldots, N, \quad x \in \mathscr{X}, \tag{3}$$

where y_i is the response at the vector of explanatory variables x_i. The errors ε_i are identically and independently distributed with zero mean and some constant variance. $f(x_i, \beta)$ is a known continuous function assumed to capture the relationship between the input x and output y through the vector of unknown model parameters β. The region available for experimentation, known as the design space, is $\mathscr{X}$ and it contains all possible values of the vectors $x_i, i = 1, \ldots, N$. We require the design space to be a compact set so that the optimum exists. If the model $f(x, \beta)$ can be written as $f(x)^{\mathrm{T}}\beta$, we refer to it as linear. Otherwise we refer to it as non-linear.

Given the study objective and the fixed total number of observations N available for the study, our goal is to optimally select a set of so-called "support points" x_i's, $i = 1, 2, \ldots, k$, from $\mathscr{X}$ to observe responses. If the model of interests has v experimental factors then each x_i can be represented as a vector with components $x_{i1}, \ldots, x_{iv}$. If each x_i is replicated n_i times, we have an exact design, subject to $\sum_{i=1}^{k} n_i = N$. It follows that the total number of variables to optimize over can increase rapidly as the number of support points k increases. For example, if we wish to estimate all parameters in a linear model with $v = 5$ and all main effects and two-factor interactions, then there are 16 parameters to estimate and k must be at least 16. This means that our optimization problem consists of a total of 95 variables, $16 \times 5 = 80$ factor settings plus $16 - 1 = 15$ replicate counts. In practice, optimal designs for a high-dimensional model with several factors are rarely minimally supported (i.e. contain one support point per parameter), so the number of variables to optimize can be even larger.

The optimality criteria for measuring design fitness are generally very complicated mathematical formulations and very few model-criterion pairings result in simple, closed-form solutions to this challenging optimization problem. The analytic descriptions that have been found are primarily for simple models, such as low-order, linear polynomials or nonlinear models with one or two predictors. Consequently, determining optimal exact designs is difficult as there is no general theory for finding them or confirming the optimality of candidate designs. For this reason we refer to the exact designs found by DE as optimal exact designs, but note that their optimality cannot be confirmed and instead rely on comparisons to designs implemented in practice.

An alternative approach is to instead work with approximate designs. These designs are defined by the proportion of runs $p_i = n_i / N$ at each support point, with $\sum_{i=1}^{k} p_i = 1$. When the criterion is a convex function of the design, there are numerous important advantages of working with approximate designs versus exact designs. They include (i) a unified framework for studying and finding optimal approximate designs, (ii) theory to confirm the optimality of a design, and (iii) an assessment of proximity of a design to the optimum without knowing the latter. For background in approximate designs, see design monographs, such as [31, 32], among several others.

In practice, an optimal design is implemented by rounding each Np_i to the nearest integer and making sure the resulting replicate counts sum to N. This approximate design formulation was first presented by Kiefer and is now the standard approach to designing a study for nonlinear models and special cases of linear models [31]. For this reason our primary goal in each example is to find optimal approximate designs, but in many cases we also present the results of searching for optimal exact designs. We denote a k-point approximate design with weight p_i at $x_i, i = 1, \ldots, k$ by ψ. In what follows, we drop the "approximate" qualifier and assume that all designs are approximate unless otherwise specified.

Let $\nabla f(x, \beta)$ be the partial derivative of the mean function with respect to the model parameters β and define the normalized information matrix for the design ψ by

$$M(\psi, \beta) = \sum_{i=1}^{k} p_i (\nabla f(x_i, \beta))(\nabla f(x_i, \beta))^{\mathrm{T}}. \tag{4}$$

The covariance matrix of the maximum likelihood estimates for the parameters β is inversely proportional to $M(\psi, \beta)$, so making the information matrix large in some sense is desirable. One popular choice of criterion to meet this goal is to maximize the determinant of M. When errors are independent and normally distributed, it can be shown that such a design minimizes the volume of the confidence ellipsoid for β and thus provides the most precise estimates. This is commonly called D-optimality (with D standing for determinant) and the fitness function is defined as

$$h(\psi, \beta) = -\log|M(\psi, \beta)|. \tag{5}$$

The logarithmic function in this criterion serves a critical purpose; for fixed β, it can be shown that Equation (5) is a convex function of ψ in the space of all designs defined over $\mathscr{X}$. Accordingly, a design that minimizes it over all designs on $\mathscr{X}$ is labeled D-optimal. For linear models, the information matrix does not depend on the parameters β, so any convex function of it can be minimized directly by choices of k, $x_1, \ldots, x_k$ and $p_1, \ldots, p_k$. In the case of a non-linear model, the matrix $M(\psi, \beta)$ depends directly on β, which it is our goal to estimate. The simplest way to overcome this circular issue is to assume a set of initial estimates or nominal values β^0 for the parameters. These can come from expert knowledge, related experiments or a pilot study. With this approximation, the criterion only depends on ψ and can be optimized directly in the same manner as the linear case. We refer to such designs for non-linear models as locally D-optimal since they depend on the choice of nominal values [33].

While there are many designs for other functions of the information matrix, D-optimal designs are by far the most widely used in practice. Another design criterion for estimating parameters is to minimize the sum of the variances of the estimated parameters. This is tantamount to finding a design that minimizes the trace of $M^{-1}(\psi, \beta)$. Such a criterion can also shown to be convex and the design that minimizes it over all designs on $\mathscr{X}$ is known as the A-optimal design.

Sometimes a researcher may be interested in estimating a function of the model parameters. In this case, a different type of criterion is required. If $c(\beta)$ is the user-selected function of interest, β^0 is the nominal value for β and $\nabla c(\beta)$ is the derivative of $c(\beta)$ with respect to β, we minimize $(\nabla c(\beta^0))^{\mathrm{T}} M^{-1}(\psi, \beta^0)(\nabla c(\beta^0))$ over all designs on $\mathscr{X}$. This is minimizing the asymptotic variance of the estimated function of interest. We call such designs c-optimal. Applications of c-optimal designs are abundant. For example, suppose we want to find the dose level of a drug that produces an 80% response rate, or we want to estimate the lethal dose that results in a 5% death rate. In both of these cases, the quantities of interest can be expressed as a nonlinear function of the model parameters β and c-optimal designs can be used to properly estimate them.

For each of these criteria, we compare designs by measuring the relative magnitude of their fitness function values. For instance, if we want to compare two designs, ψ_1 and ψ_2 and the criterion is A-optimality, we use the ratio of their criterion values. Specifically, the A-efficiency of ψ_1 relative to ψ_2 is

$$\frac{tr(M^{-1}(\psi_2, \beta))}{tr(M^{-1}(\psi_1, \beta))}. \tag{6}$$

If the ratio is less than 1, ψ_1 is less A-efficient than ψ_2, and vice versa. It can be shown that if the above ratio is 0.5, then the design ψ_1 requires twice as many observations to perform as well as ψ_2. If ψ_2 is known to be A-optimal, then we refer to this ratio as the A-efficiency of ψ_1. Similar formulae hold for other design criteria, except for D-optimality, in which the ratio must be raised to the power $1/q$, where q is the length of the parameter vector β. This transformation allows us to maintain the interpretation that design ψ_1 with efficiency x requires $1/x$ times as many runs to achieve the same performance as ψ_2.

Finding optimal designs, regardless of the criterion, is always a challenging problem, both mathematically and computationally. In the case of approximate designs with a convex optimality criterion, there is a powerful and simple tool known as the general equivalence theorem for

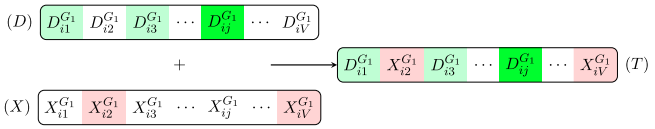

Fig. 2. An illustration of the crossover procedure for a single agent in the Differential Evolution algorithm. Here we create trial vector $T_i^{G_1}$ by combining $X_i^{G_1}$ and $D_i^{G_1}$. The light green elements of $T_i^{G_1}$ come from the donor vector $D_i^{G_1}$ and the light red elements come from the target vector $X_i^{G_1}$. The jth element of $T_i^{G_1}$ is marked in dark green as it comes from $D_i^{G_1}$ with probability one.

checking the optimality of a design over all designs defined on $\mathscr{X}$. This result is based on directional derivative considerations of the convex design criterion and is widely discussed in the primary optimal design literature: see Ref. [34] and design monographs [32,35].

Each convex optimality criterion has a unique equivalence theorem. For example, when we have a nonlinear model and the nominal value for the vector of model parameters is β^0, then for the three criteria discussed above, a design ψ^* is locally.

- D-optimal if and only if it satisfies

$$\left(\nabla f(x,\beta^0)\right)^{\mathrm{T}} M^{-1}(\psi^*,\beta^0)\left(\nabla f(x,\beta^0)\right) - q \le 0 \quad \text{for all} \quad x \in \mathscr{X}, \tag{7}$$

- A-optimal if and only if it satisfies

$$\left(\nabla f(x,\beta^0)\right)^{\mathrm{T}} M^{-2}(\psi^*,\beta^0)\left(\nabla f(x,\beta^0)\right) - tr\left(M^{-1}(\psi^*,\beta^0)\right) \le 0, \quad \text{for all} \quad x \in \mathscr{X}. \tag{8}$$

- c-optimal if and only if it satisfies

$$\left(\left(\nabla f(x,\beta^0)\right)^{\mathrm{T}} M^{-1}(\psi^*,\beta^0)\nabla c(\beta^0)\right)^2 - \left(\nabla c(\beta^0)\right)^{\mathrm{T}} M^{-1}(\psi^*,\beta^0)\left(\nabla c(\beta^0)\right) \le 0,$$

$$\text{for all} \quad x \in \mathscr{X}. \tag{9}$$

Moreover, each of the above inequalities becomes an equality at the support points of the optimal design ψ^*. The functions on the left hand side of the inequalities are sometimes called sensitivity functions. In practice, achieving exact equality at every support point is unlikely, but up to some small threshold (e.g. 10^{-6} for our purposes) optimality is still guaranteed.

The equivalence theorems have several important consequences. If $\mathscr{X}$ is one or two-dimensional, the optimality of a design can easily be confirmed by plotting the sensitivity function across the design space and ascertain if the conditions of the equivalence theorem are met. If $\mathscr{X}$ is 3-dimensional or higher, the features of the multivariate sensitivity plot are harder to appreciate visually. Equivalence theorems can also be directly used in construction algorithms for generating optimal designs and checking whether a design is close to the optimum. If the design is not optimal, its proximity to the optimum can be measured by an efficiency lower bound. In practice a design with a high efficiency lower bound may be adequate. In the following section we implement DE to quickly locate various types of optimal designs for chemical experiments.

4. DE applications to optimal design

This section provides illustrative applications of DE to find optimal designs for a variety of chemical experiments. We start by finding designs for simple models and incrementally augment the problem difficulty. For each application, we describe the problem setup and how the features of DE can be adapted to fit the situation. We then show that DE can

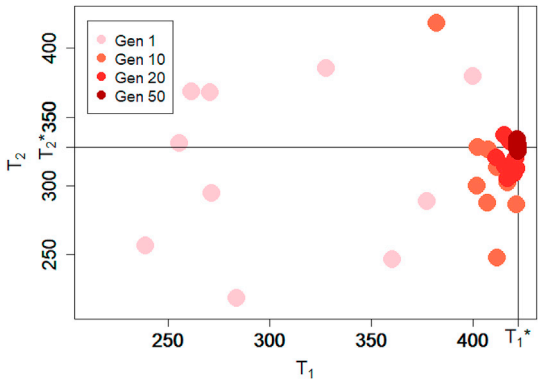

Fig. 3. DE convergence to the locally D-optimal design for the Arrhenius Equation.

effectively find optimal designs that coincide with published results or outperform them. The utility of DE becomes clearer when it is laborious to calculate the optimal design from theory and DE finds it almost instantly with the proper setup.

DE algorithms, like other notable nature-inspired metaheuristic algorithms, are widely and freely available in different formats. We choose to use the R codes available from the package DEoptim in R Version 3.4.3 [36]. All computations were carried out using a 2016 Lenovo P50 Thinkpad 2.9 GHz Intel Core i7 with 16 GB RAM on 64 bit Windows 10. In what is to follow, we first use DE to search for a minimally supported optimal design. This is a common starting strategy because the optimization problem is simpler with fewer variables to optimize and the search is confined to all designs on $\mathscr{X}$ with q points. However, the resulting minimally supported optimal design may not be optimal among the class of all designs on $\mathscr{X}$. An equivalence theorem is needed to confirm its optimality among all designs on $\mathscr{X}$. Later, we discuss what happens if DE is initialized with more points than the optimal design requires and how DE may be altered to solve more difficult design problems.

4.1. Estimating parameters in the Arrhenius Equation

We begin by finding locally D-optimal designs for estimating the two parameters in the Arrhenius equation commonly used in chemical experiments [37]. This model describes the relationship between the mean temperature and reaction rate of a process. Its basic form is given by

$$Er = A\exp(-B/T), \quad \beta = (A,B)^{\mathrm{T}}. \tag{10}$$

Here Er denotes the expectation of the random variable r, denoting the reaction time. It is an exponential function of the parameter B, the activation temperature, and the parameter A is a multiplicative constant called the Arrhenius frequency parameter. The design variable is the temperature T and good choices for its settings produce efficient estimates for A and B. Many experiments have been performed to determine the parameter values for different chemical reactions to better understand their dependence on temperature [38–40].

Our specific application concerns the reaction given by $NO + O_3 \rightarrow NO_2 + O_2$, which captures the loss process for ozone in the troposphere and stratosphere [41]. In order to study the temperature dependence of this reaction we seek an experimental design that is supported at two temperature points. Since the equation is non-linear with respect to B any optimal design will depend on a set of initial parameter values β^0. For this example we choose $\beta^0 = (3.0 \times 10^{-12}, 1500)^{\mathrm{T}}$ according to preliminary results from NASA's Jet Propulsion Lab [42]. The acceptable range of temperatures, the design space $\mathscr{X}$, is fixed at $[212, 422]$, so that we may compare the results of DE with the designs found in Ref. [43]. From this we calculate the vector ∇Er as the first-order partial derivatives of the model with respect to A and B, which is given by

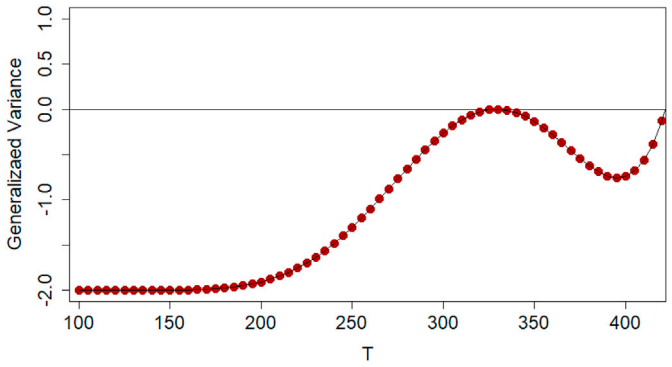

Fig. 4. Sensitivity function for local D-optimality of the DE-generated design for the Arrhenius Equation.

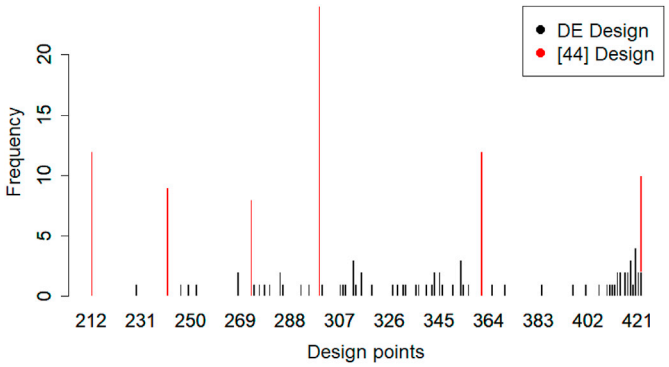

Fig. 5. Design points for 75-point exact designs for the Arrhenius Equation.

$$\nabla Er = \left(\exp(-B/T), \frac{A\exp(-B/T)}{T}\right)^{\mathrm{T}}. \tag{11}$$

Evaluating at the nominal values for the parameters β^0, we calculate the information matrix in Equation (4) and plug it into our fitness function. Assuming the design of interest ψ has two points T_1 and T_2 with weights p_1 and p_2, then the D-optimality criterion is

$$h(\psi, \beta^0) = -\log\left|\sum_{i=1}^{2} p_i(\nabla Er_{\beta^0}(T_i))(\nabla Er_{\beta^0}(T_i))^{\mathrm{T}}\right|. \tag{12}$$

We implement DE to find the locally D-optimal design. In addition to the two temperatures we also need to search for the proportion of observations to take at one of the temperature levels (since the other is then determined by the unity constraint). The number of variables to optimize is therefore $V = 3$, so this is a relatively small optimization problem. Accordingly, we set a population size of $P = 10$ agents and limit our DE search for the optimum to $G_{max} = 50$ generations, resulting in a total of 500 evaluations of the fitness function. Each candidate design is represented by a vector with the two design points first followed by the weight of the first point. We choose the standard values $F = 0.8$ and $CR = 0.9$ from Section 2 for the parameters since the optimization of h is a non-separable problem .

The algorithm took only 0.1 s to produce a design equally supported (0.5 wt at each point) at the temperature values 329.3 and 422.0. Fig. 3 shows the evolution of the ten candidate solutions through the fifty generations across the two temperature dimensions. Since the algorithm is able to find the correct weight in the first few generations this variable is omitted from the plot. Candidates from earlier generations are given a lighter color to represent that they are far from optimal while those from generation 50 are dark, indicating their optimality. We observe that the algorithm locates an optimal support point at the boundary of the design space after just twenty generations and the remaining generations are used to finely tune the second point.

To confirm that the design found by DE is locally D-optimal, Fig. 4 shows the sensitivity function from (7) for the generated design. We observe that over the design space, the function is bounded above by 0, with equality at the two temperature settings. By the equivalence theorem, this confirms the local D-optimality of the DE-generated design when the vector of nominal values is $\beta^0 = (3.0 \times 10^{-12}, 1500)^{\mathrm{T}}$.

While the approximate design we found is locally D-optimal, conducting an experiment in the laboratory would require an exact design. One such experiment that studied the reaction we are considering was conducted in Ref. [44]. Following their setup, we use DE to search for a 75-run D-optimal exact design. Increasing the population size to $P = 50$ and the number of iterations to $G_{max} = 200$ while leaving the other parameters and nominal values the same, we were able to find a 75-run exact design with a lower criterion value (85.77 versus 86.50) than the implemented design. This result further demonstrates DE's ability to locate practical designs for non-linear models with ease. A visual representation of the design found by DE is given in Fig. 5.

Having shown that DE can find exact and locally optimal designs for this simple model, we now apply it to an extended version known as the Modified Arrhenius equation [45]. This modification is used when temperature settings are allowed to vary over a wide range. The model is given by

$$Er_{mod} = \frac{A'}{T^m}\exp(-B/T), \tag{13}$$

where A' is now a positive parameter independent of the temperature setting, m is a new parameter that describes how the exponential scales with temperature and Er_{mod} is the mean reaction time under this modified model. Following [43] we fix $m = 5$ and set the nominal parameters to $\beta^0 = (A' = 1, B = 1500)^{\mathrm{T}}$. This setting for B was retained from the standard model and we set $A' = 1$ for simplicity because the determinant of the information matrix does not depend on its value.

To find a locally D-optimal design for this 2-parameter model (since m is fixed) using DE, we first compute the gradient of the mean function

$$\nabla Er_{mod} = \left(T^{-m}\exp(-B/T), -A'T^{-(m+1)}\exp(-B/T)\right)^{\mathrm{T}}. \tag{14}$$

We then repeat as before and use a 2-point design to initiate the DE algorithm to search for a locally D-optimal design for estimating A' and B using the criterion in Equation (5) with the above gradient. For this more challenging problem, we increase G_{max} to 60 and leave the remaining DE parameters as they were. We keep the design space as $[212, 422]$. After reaching the maximum number of generations in 0.1 s, DE found a design equally supported at 392.72 and 212.60. This design has 99.9% D-efficiency relative to the D-optimal design supported equally at 390.5 and 209.5 on an unbounded design space. See Ref. [42] for details.

The above examples show that DE can solve simple optimization problems very fast with little or no tuning of the parameters required and minimal computational expense. The next few applications are more complicated to optimize; they either have additional constraints on the optimization problems or more variables to optimize. We show that DE can find the optimum with similar ease even if we initialize the algorithm with candidate designs that have more than the required number of support points.

4.2. *Estimating the effect of reaction order and decay rate on concentration*

For a more complicated example, consider a modification of the model in Ref. [46] for studying the influence of reaction order and decay rate on the concentration of a chemical at a given time. The original model is given by

$$Ec = (1 - (1 - \lambda)\theta T)^{\frac{1}{1-\lambda}}, \tag{15}$$

where c is the concentration at time T, λ is the reaction order and θ is the

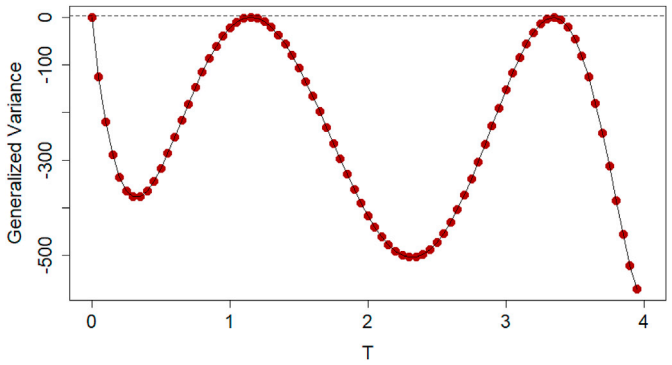

Fig. 6. Sensitivity function for local A-optimality of the DE-generated design for the modified Atkinson model.

decay rate. In some circumstances, it is appropriate to include a third parameter ν to make the model more flexible. The modified version of Equation (15) now takes the form

$$Ec^* = \frac{(1-(1-\lambda)\theta T)^{\frac{1}{1-\lambda}}}{1+\exp(\nu\lambda)}, \quad \beta = (\lambda, \theta, \nu)^{\mathrm{T}}. \tag{16}$$

For this model we seek a locally A-optimal design with 3 support points. Taking inspiration from Ref. [46] we select the nominal parameters $\beta^0 = (0.5, 0.5, 0.1)^{\mathrm{T}}$. Following a similar procedure as in the previous example, we begin by calculating the first order partial derivatives of the mean function to obtain the information matrix $M(\psi, \beta^0)$, where ψ is the design and β^0 is the nominal value for the vector of model parameters. We recall that the A-optimality criterion is given by

$$h(\psi, \beta^0) = tr\left(M^{-1}(\psi, \beta^0)\right). \tag{17}$$

We seek a minimally supported design with 3 points over the design space $\mathscr{X} = [0, 4]^3$. Since there is only a single explanatory variable, the number of dimensions to search over is $3 \times 2 - 1 = 5$. We set $P = 50, F = .8, CR = .9$ and $G_{max} = 100$. Running the algorithm with these settings did not allow for enough exploration of the space, so G_{max} was increased from 100 in increments of 50 until 300. This led to a design equally supported at 0.000, 1.151, and 3.343.

Generating this design took only 1.5 s of CPU time and required minimal tuning of the parameters. Fig. 6 shows that the sensitivity function of the DE-generated design satisfies the general equivalence theorem (8), confirming the local A-optimality of the design on $\mathscr{X}$ with the nominal values β^0. For this example we also used DE to search for A-optimal exact designs for this model, but in all cases we considered ($N = 3, 10, 40, 100$) the design found by DE is actually the rounded approximate design, so the details have been omitted.

Can DE find an optimal design if we instead start the search using candidate designs with more points than the (unknown) number of support points of the optimal design? To investigate this issue, we repeat the above analysis, but instead of optimizing over 5 variables with 3 support points and 2 wt, we supply designs with 6 support points and 5 wt, making for a total of 11 variables. DE was again able to converge to the equally-supported A-optimal design by clustering the excessive points. Due to the increased dimension of the search space, achieving this result required 300 generations with all other parameters maintaining their values from the 5-variable situation.

Fig. 7 shows the best design from every 100 generations. The *x*-axis gives the values of the 6 support points while the *y*-axis shows the corresponding weights. As before, support points from later generations are given a darker color to indicate their convergence to the optimal design. When the best design from a given generation has several points that are indistinguishable their weights are summed and a single point is shown. This plot demonstrates the rapid pace at which DE begins to cluster the additional points and move towards the optimal weights. Table 1 gives

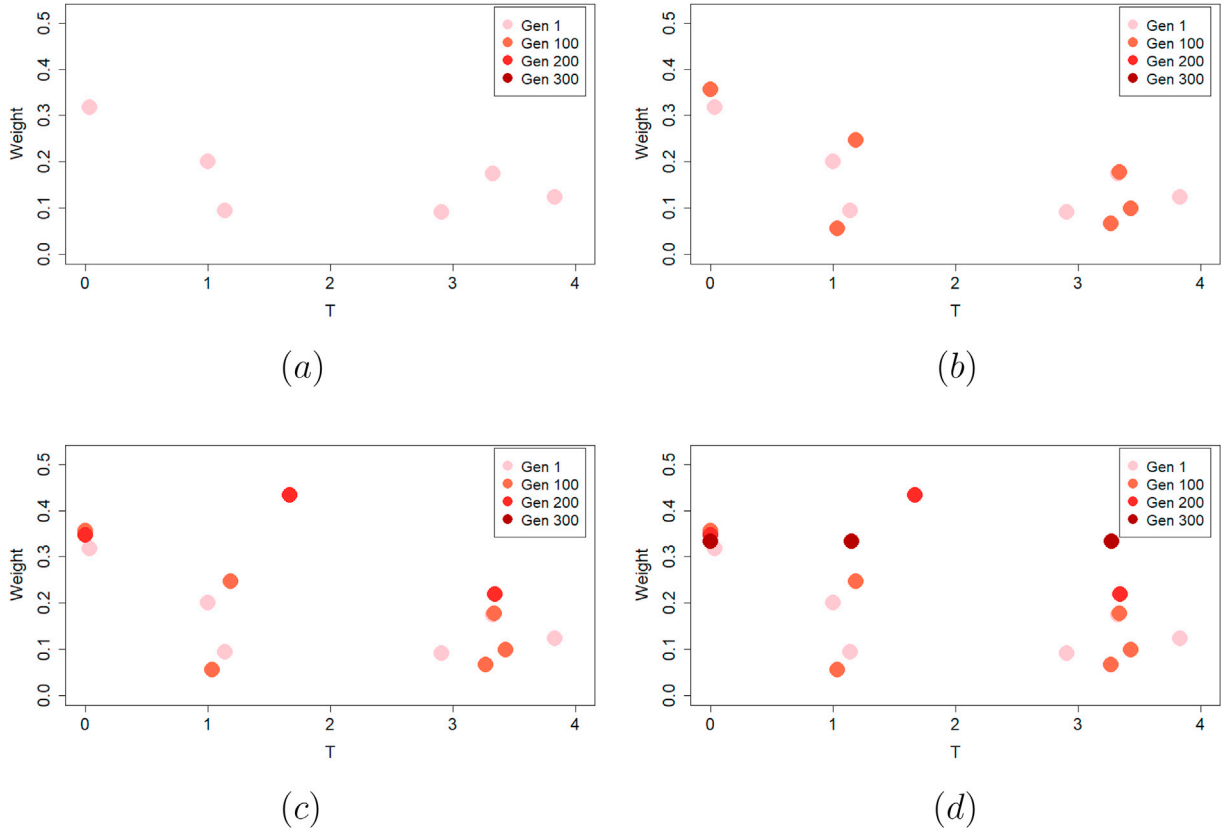

Fig. 7. Convergence of DE for the Atkinson example initialized with 6 support points. Each panel (a)-(d) displays the best design found after every 100 generations.

Table 1
The best design found by DE for Atkinson's model after every 100 generations.

Gen	$T_1(p_1)$	$T_2(p_2)$	$T_3(p_3)$	$T_4(p_4)$	$T_5(p_5)$	$T_6(p_6)$
1	0.038	1.004	1.143	2.910	3.329	3.832
	(0.318)	(0.201)	(0.094)	(0.091)	(0.174)	(0.123)
100	0.000	1.036	1.187	3.267	3.435	3.339
	(0.357)	(0.055)	(0.246)	(0.066)	(0.099)	(0.177)
200	0.000	1.628	1.632	3.338	3.339	3.345
	(0.347)	(0.153)	(0.281)	(0.048)	(0.004)	(0.167)
300	0.000	1.151	1.151	3.344	3.344	3.344
	(0.333)	(0.268)	(0.062)	(0.046)	(0.219)	(0.068)
Final	0.000	1.151 (0.333)		3.344 (0.334)		
	(0.333)					

the raw values from each design shown. We observe that earlier generations have six unique support points; however, by generation 200 the points form three clusters and the focus shifts to finding the appropriate weights. By generation 300 the collective weight at the three points yields the equally-supported design we report. From this example, we observe that even when the user starts with candidate designs with more points than needed, DE is still able to find the optimal design.

4.3. Estimating the effect of microemulsion mixtures on drug solubility

In our third application, we apply DE to find an optimal design for a mixture experiment with several factors. In mixture experiments, each run is a mixture of the same ingredients, but the relative proportion of each ingredient in each run may be different, and thus affect the mean measured response. These problems arise frequently in life and physical sciences. For example [47], studied the effect of microemulsion formulation composition on the solubility and dissolution of drugs.

There are several possible models for a mixture experiment with a fixed number of ingredients. If there are d ingredients, the most frequently used linear model is the Scheffé polynomial [48] given by

$$Ey = \sum_{i=1}^{d} \beta_i x_i + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \beta_{ij} x_i x_j. \tag{18}$$

In the above equation, there are $q = d + \binom{d}{2}$ parameters $\beta_1, \ldots, \beta_d$, $\beta_{12}, \ldots, \beta_{(d-1)d}$ and d inputs $x_1, \ldots, x_d$. This model is similar to a standard linear model with d factors and all two-factor interactions, except that the design space is a $d-1$ simplex, i.e. each $x_i \geq 0$ subject to $x_1 + x_2 + \ldots + x_d = 1$.

To find D-optimal designs for this problem over the $(d-1)$-dimensional simplex using DE, we proceed as before by first computing the information matrix. In this experiment, there are $d = 3$ ingredients: oil, water, and a surfactant to study the drug dissolution properties y of microemulsion formulations. The model used by Ref. [47] to study this relationship is given by

$$Ey = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 + \beta_{123} x_1 x_2 x_3, \tag{19}$$

which is equivalent to Equation (18) with the addition of a three-way interaction of all ingredients. In Ref. [47] the authors fit this model with the 13-points exact design in Table 3(b). We apply DE to find an approximate design of comparable size with a better ability to estimate the model parameters. Since the three ingredient proportions must sum to 1 in each run, we only need to optimize over two experimental factors. A search among all 13-point designs shows the total number of variables that DE has to optimize is 38. We initially set $P = 175$ and ran the algorithm for $G_{max} = 2000$ generations. Clearly this setting does not follow the rule of thumb $P \geq 10V$ in Section 2. However, we find that the marginal improvement to the final design does not warrant using additional resources to produce 380 agents. Instead it is more efficient to use only half as many agents and run the algorithm for twice as many generations. We set $F = 0.8, CR = 0.7$ and slowly increases the value of CR to 0.9 through multiple trials as it improves performance. We observe that the basic form of DE is not able to quickly locate an optimal design for this problem due to the large amount of time required per generation and this limits the number of generations possible. To overcome this issue we implemented a parallel version of DE [49] that spreads the calculations required in each generation over many nodes. DE naturally lends itself to parallel computing since many of its operations can be performed in isolation and then reassembled. Further details of this approach and other DE variants are discussed in Section 5.

Table 2
13-point approximate design for the emulsion mixture experiment. (a) is the 13-point design found by DE and (b) is a 7-point D-optimal design that results from merging the rows of (a).

(a)

Run	x_1	x_2	x_3	p_i
1	1	0	0	0.143
2	0	1	0	0.142
3	0	0	1	0.143
4	0	0.5	0.5	0.083
5	0	0.5	0.5	0.059
6	0.5	0	0.5	0.077
7	0.5	0	0.5	0.065
8	0.5	0.5	0	0.143
9	0.34	0.33	0.33	0.035
10	0.33	0.34	0.33	0.012
11	0.33	0.33	0.34	0.037
12	0.34	0.33	0.33	0.058
13	0.33	0.34	0.33	0.002

(b)

x_1	x_2	x_3	$\sum p_i$
1	0	0	$\frac{1}{7}$
0	1	0	$\frac{1}{7}$
0	0	1	$\frac{1}{7}$
0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{7}$
$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{7}$
$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{7}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$

Using this variant of DE we were able to locate the D-optimal design given in Table 2(a). It took 24 s of CPU time across 8 nodes to converge to a 13-point design with many duplicated points. Further investigation reveals that the design is equally weighted at 7 points and its D-optimality can be confirmed visually using the equivalence contour plot shown in Fig. 8, which shows that the contour values at each of the 7 points from our design is approximately 0. In this plot x_1 and x_2 are given on the x and y axes respectively. With these two values determined, x_3 can be inferred. The contour value is derived from the sensitivity function from Equation (7). The key implication is that the 13-point design used by the authors is not optimal for the proposed model and a direct calculation shows that the relative D-efficiency of their design to the one found with DE is only 79%. Through this comparison we observe the benefits of DE: DE is fast and finds the optimal design effortlessly and accurately.

We note that this comparison between the design in Ref. [47] and the DE generated design is not exactly fair. While both designs have 13 points, our design has only 7 unique supports and is an approximate design, as defined in Section 3. To implement this design in practice we need to multiply each weight by the desired run size (in this case 13). Since the weights are not regular fractions of 13 we will need to do some rounding in order to achieve a useable design. Through this process we lose some efficiency. In order to mitigate this loss we could instead find an 13-point D-optimal exact design directly, but this is known to be a very

Table 3
13-point exact designs for the emulsion mixture experiment.(*a*) is the 13-point design found by DE and (*b*) is the design implemented in Ref. [47].

(a)		
x_1.	x_2	x_3
0	0	1
0.01	0.01	0.98
0	0.50	0.50
0.01	0.49	0.50
0	1	0
0.01	0.97	0.02
0.32	0.33	0.35
0.34	0.34	0.32
0.49	0	0.51
0.51	0	0.49
0.49	0.51	0
0.53	0.47	0
1	0	0
(b)		
x_1	x_2	x_3
0.65	0.18	0.17
0.65	0.22	0.13
0.65	0.25	0.10
0.69	0.16	0.15
0.69	0.19	0.12
0.72	0.12	0.17
0.73	0.13	0.14
0.75	0.15	0.10
0.76	0.09	0.15
0.78	0.05	0.17
0.79	0.09	0.12
0.82	0.05	0.13
0.85	0.05	0.10

difficult problem to solve analytically for general run size. However, DE is able to overcome this challenge. To do this we search over the 26-dimensional space for a good exact design. Setting $P = 150$, $F = 0.8$ and $CR = 0.9$ DE required 15 s across 8 nodes and $G_{max} = 2000$ generations to find the design presented in Table 3(a). We observe that the design points all come from the optimal approximate design, with some points replicated once. This design has 99% efficiency relative to the optimal one found earlier, again showing a clear advantage over the design chosen by the researchers. Thus, DE is capable of producing both exact and approximate designs for problems with a large number of optimization parameters.

4.4. Estimating the effect of substrate concentration and inhibition amount on reaction velocity

Until now we have used DE to find optimal approximate and exact designs for linear and non-linear models of various complexities. However, we have so far only considered non-linear models with a single input variable and with the prior knowledge of a nominal set of values. It is useful to consider a final example in which neither of these situations apply. For this example we will search for Bayesian D-optimal designs for the 4-parameter mixed inhibition model described in Ref. [50,51]. This model considers the relationship between substrate concentration s, inhibition amount i, and reaction velocity v and is given by

$$Ev = \frac{V_{max} \cdot s}{k_m\left(1 + \frac{i}{k_{ic}}\right) + s\left(1 + \frac{i}{k_{iu}}\right)}. \tag{20}$$

In this model the vector of parameters is $\beta = (V_{max}, k_m, k_{ic}, k_{iu})^{\mathrm{T}}$, representing the velocity at the maximum concentration, the Michaelis-Menten constant, and two dissociation constants respectively. For this model we still assume that we have some prior knowledge of the true value of these parameters, but unlike the previous examples it takes the form of a multivariate distribution. The goal is to use DE to find a Bayesian D-optimal design that minimizes the D-optimality criterion in Equation (5) averaged over the prior distribution. Formally, we seek a design that minimizes

$$h(\psi, \beta) = -\log\left(\int_{R^4} |M(\psi, \beta)|\pi(\beta)d\beta\right), \tag{21}$$

where $\pi(\beta)$ is the multivariate prior we assume for β.

We consider several choices for $\pi(\beta)$ including both discrete and continuous distributions with and without correlation. Following [50,51] we assume that the mean of each continuous prior is given by $\beta^{\mu} = (7.298, 4.386, 2.582, 5.0)^{\mathrm{T}}$. Furthermore, we consider two design spaces: $9 \le s \le 30, 0 \le i \le 60$ and $0 \le s \le 30, 18 \le i \le 60$. The set of plausible parameter values is given by $k_m \in [4,5], k_{ic} \in [2,3]$, and $k_{iu} \in [4,5]$. To approximate the integral in Equation (21) we average over a systematic sample from the prior distribution. Taking inspiration from Ref. [52], the sample is derived from a series of Halton draws. This reduces computational time and provides a reasonable estimate of the integral. To further speed up the algorithm we use the same parallel version of DE that was implemented in the previous example.

We consider uniform priors, both continuous and discrete, and two multivariate normal distributions, one with independence and the other with weak to moderate correlation. For each prior we search for a minimally supported design by initializing DE with a population size of $P = 120$ for $G_{max} = 250$ generations. We begin with $CR = 0.1$ and $F = 0.8$ and increase CR in steps of size 0.1 between iterations if the algorithm fails to converge. For each continuous prior we use 125 Halton draws to evaluate the integral [52].

Table 4 shows the DE designs under each combination of prior and design space. In this table Σ_1 and Σ_2 are variance-covariance matrices with diagonal equal to $(0.50, 0.11, 0.11, 0.20)$, and an average correlation of 0 and 0.46 respectively. Specifically, Σ_1 and Σ_2 are given by

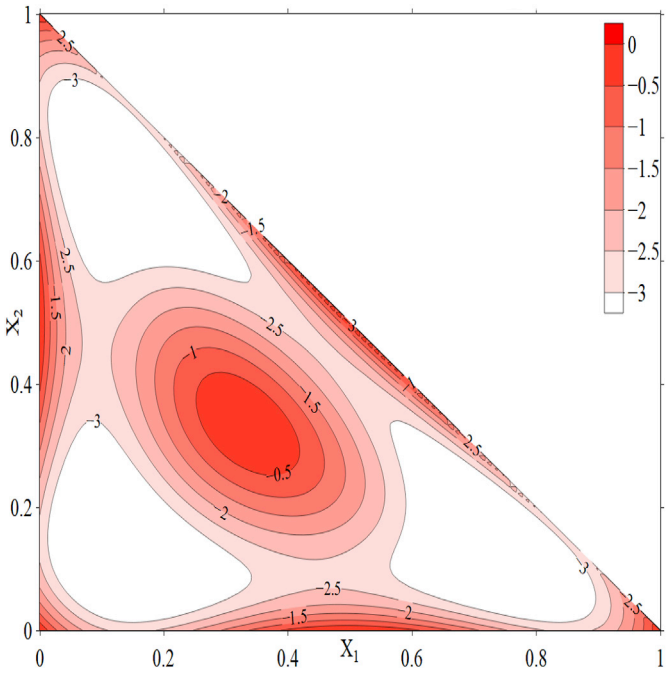

Fig. 8. Sensitivity contours of the DE-generated design for the emulsion mixture experiment. The function is maximized at the 7 points included in the design, confirming its D-optimality.

Table 4
Bayesian D-optimal designs for the enzyme kinetic model under four prior distributions for β and two design spaces.

$s \in [9,30], i \in [0,60]$												
Run	MVNorm (β^{μ}, Σ_1)			MVNorm (β^{μ}, Σ_2)			U $(\beta^{\mu}-1, \beta^{\mu}+1)$			U_d		
	s	i	p	s	i	p	s	i	p	s	i	p
1	30	4.07	0.25	30	4.07	0.25	30	4.32	0.25	30	4.35	0.25
2	9	3.57	0.25	9	3.61	0.25	9	3.82	0.25	9	3.81	0.25
3	30	0	0.25	30	0	0.25	30	0	0.25	30	0	0.25
4	9	0	0.25	9	0	0.25	9	0	0.25	9	0	0.25
$s \in [0,30], i \in [18,60]$												
Run	MVNorm (β^{μ}, Σ_1)			MVNorm (β^{μ}, Σ_2)			U $(\beta^{\mu}-1, \beta^{\mu}+1)$			U_d		
	s	i	p	s	i	p	s	i	p	s	i	p
1	29.81	18.03	0.25	29.95	18.11	0.25	29.92	18	0.25	29.87	18.03	0.25
2	4.28	18.06	0.25	4.64	18.03	0.25	4.75	18.14	0.25	4.59	18.05	0.25
3	29.96	41.44	0.25	29.71	39.11	0.25	29.59	40.16	0.25	29.77	42.58	0.25
4	4.77	39.56	0.25	4.78	39.17	0.25	5.25	41.05	0.25	5.06	40.47	0.25

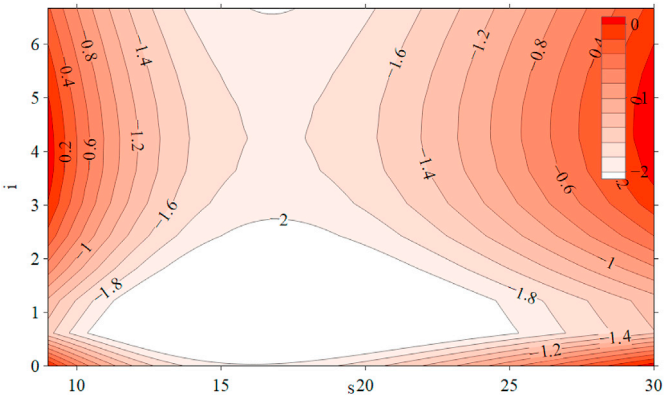

Fig. 9. Sensitivity contour of the design found by DE under the independent multivariate normal prior and design space $s \in [9,30], i \in [0,60]$.

$$\Sigma_1 = \begin{bmatrix} 0.500 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.110 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.110 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.200 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 0.500 & 0.052 & 0.148 & 0.010 \\ 0.052 & 0.110 & 0.092 & 0.098 \\ 0.148 & 0.092 & 0.110 & 0.052 \\ 0.010 & 0.098 & 0.052 & 0.200 \end{bmatrix}$$

U_d is a discrete uniform distribution of $4^4 = 256$ grid points over the parameter space. For the design space that requires a larger concentration $s \geq 9$ we see that each design places points at extreme values of s, two where there is no inhibition and two where there is slight inhibition. Under the second design space that requires more inhibition $i \geq 18$, we observe that the optimal design points lie away from the boundaries of the space, but are still at the extreme ends of the concentration range but more towards the lower bound of the inhibition range. Using the Bayesian extension of the general equivalence theorem we can visually verify that the designs found by DE are indeed optimal under each particular prior and design space. Fig. 9 displays one such plot of the sensitivity function of the DE-generated design.

The above approximate designs are implementable if the sample size is a multiple of 4. Researchers may be interested in a design of a specific size that does not meet this requirement. For this reason it is also useful to showcase DE's ability to find optimal exact designs in a situation where prior information about the parameters is limited. To do this we consider the 21-point design implemented in Ref. [51]. We repeat the same search as before, but this time over a 42-dimensional space. Due to the increased size of the problem we initialize DE with a population size of $p = 210$ for $G_{max} = 1250$. In the same manner as the approximate case we begin with

Table 5
Relative D-efficiency of design from Ref. [51] to DE 21-point exact designs for the enzyme kinetic model under four prior distributions for β and two design spaces.

	$s \in [9,30], i \in [0,60]$	$s \in [0,30], i \in [18,60]$
MVNorm (β^{μ}, Σ_1)	79.0%	78.9%
MVNorm (β^{μ}, Σ_2)	79.0%	79.0%
U $(\beta^{\mu}-1, \beta^{\mu}+1)$	79.0%	78.9%
U_d	78.9%	79.1%

CR = 0.1 and F = 0.8 and increase CR in steps of size 0.1 as necessary.

Table 5 reports that the exact designs found by DE have much higher D-optimality than the design used in practice. This indicates that even when a large number of design points is desired DE is still capable of finding an efficient design. This observation holds regardless of the prior distribution we assume for the parameter values.

5. Popular DE variants and hybrid algorithms

In the two previous example we employed a parallel version of DE to arrive at designs that outperforms the ones available in the published literature. There are many such advancements to the basic DE algorithm. The most popular variations on the basic DE algorithm fall into two classes: adaptive-tuning and multiple-objective. For many optimization problems solved with metaheuristic algorithms the selection of tuning parameters is a bottleneck. DE attempts to solve this issue by having only a few such parameters, yet sometimes many trials are required to choose their correct values. Many of the enhancements for DE use a self-adaptive parameter search that studies the history of past agents to appropriately tune parameters for future generations. Some popular members of this class include JADE, SADE, and SHADE [53–55], which have collectively been cited hundreds of times. The other major class of DE variants concerns optimization problems with multiple objectives. There are many novel algorithms based on such multi-objective approaches to DE (for example, see Ref. [56–58]) and the leading methods involve minimizing the Pareto frontier or utilizing Lagrange multipliers.

There are also many new algorithms developed by hybridizing DE with other metaheuristic algorithms. Akin to crafting ensemble statistical models, these hybrid algorithms allow for a more thorough search of the space and the relative weight given to the approaches of each method when crafting the algorithm determines its efficacy. Some examples of hybrid DE algorithms involve blending it with Simulated Annealing, PSO, and k-means clustering [59–61]. These methods have shown great success at tackling optimization problems that the individual algorithms struggled to solve.

This overview covers some DE variants but there are many other modifications of the basic DE algorithm that have been proposed since its initial conception. Some of these methods are motivated by the

application at hand while others are motivated by further heuristic considerations aimed at improving the performance of DE in various ways. For a most recent review of the latest innovations of DE see [62].

6. Conclusion

In this article we reviewed the basic formulation of Differential Evolution and apply it to find various types of optimal designs in the chemical sciences. The algorithm is metaheuristic and is based upon the ideas of evolutionary biology. Through the genetic representation of candidate solutions to a real-valued fitness function, DE uses mutation and crossover of the genes of each agent to produce sequential improvements to the fit of subsequent generations. We discussed the foundational ideas of the algorithm and the specific methods and ideas behind tuning each parameter.

We implemented DE codes and demonstrates DE's ability to solve design problems for statistical models with different number of parameters. Both approximate and exact optimal designs can be found by DE in a few seconds to estimate some or all parameters in linear and non-linear models. DE or its variants can also be effective in solving high dimensional and other complex design problems. For high dimensional optimization problems it would be appropriate to apply different nature-inspired metaheuristic algorithms and observe whether they produce the same or approximately the same solution. If they do, we are much more assured that the solution is likely to be correct. If needed, readers can hybridize two or three such algorithms to take advantage of the positive qualities of each to enhance the hybridized algorithm performance. We close by reminding the reader that DE is a general purpose optimization tool and can solve both design and non-design optimization problems. We hope that this introduction to DE stimulates further chemometric research in this direction.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Wong was partially supported by a grant from the National Institute of General Medical Sciences of the National Institutes of Health under Award Number R01GM107639. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.chemolab.2020.103955.

References

[1] J.M. Whitacre, Recent trends indicate rapid growth of nature-inspired optimization in academia and industry, Computing 93 (2011a) 121–133.
[2] J.M. Whitacre, Survival of the flexible: explaining the recent dominance of nature-inspired optimization within a rapidly evolving world, Computing 93 (2011b) 135–146.
[3] J.H. Holland, Adaptation in Natural and Artificial Systems, second ed., MIT Press, 1975.
[4] X.S. Yang, S. Deb, Cuckoo Search via Levy Flights. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), IEEE Publications, 2009, pp. 210–214.
[5] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a Colony of cooperating agents, IEEE Trans. Syst., Man, Cybern. Part B 26 (1996) 29–41.
[6] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf optimizer, Adv. Eng. Software 69 (2014) 46–61.
[7] F.J.M. García, J.A. Moreno-Pérez, Jumping Frogs Optimization:a New Swarm Method for Discrete Optimization. Technical Report DEIOC 3/2008, Univ. of La Laguna, Tenerife, Spain, 2008. Dept. of Statistics, O.R. and Computing.
[8] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, 2010, pp. 65–74.
[9] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, Wiley, 2005.
[10] A.S. Fraser, Simulation of genetic systems by automatic digital computers. I. Introduction, Aust. J. Biol. Sci. 10 (1957) 484–491.
[11] R. Storn, K. Price, Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[12] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, 2005.
[13] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1998.
[14] V. Feoktistov, Differential Evolution: in Search of Solutions, Springer, 2006.
[15] V.P. Plagianakos, D.K. Tasoulis, M.N. Vrahatis, A review of major application areas of differential evolution, in: Advances in Differential Evolution, Springer, 2008, pp. 197–238.
[16] T. Tusar, B. Filipic, Differential evolution versus genetic algorithms in multi-objective optimization. Evolutionary multi-criterion optimization, in: 4th International Conference, Matshushima, Japan, 2007, pp. 257–271. March 2007.
[17] A.D. Lilla, M.A. Khan, P. Barendse, Comparison of differential evolution and genetic algorithm in the design of permanent magnet generators, in: Proc. IEEE International Conf. Industr. Tech., Feb. 2013, 2013, pp. 266–271.
[18] S. Tayyebi, S. Soltanali, A new approach of GA-based type reduction of interval type-2 fuzzy model for nonlinear MIMO system: application in methane oxidation process, Chemometr. Intell. Lab. Syst. 167 (2017) 152–160.
[19] Q. Yang, M. Wang, H. Xiao, L. Yang, B. Zhu, T. Zhang, X. Zeng, Feature selection using a combination of genetic algorithm and selection frequency curve analysis, Chemometr. Intell. Lab. Syst. 148 (2015) 106–114.
[20] A.G. Mercader, P.R. Duchowicz, Enhanced replacement method integration with genetic algorithms populations in QSAR and QSPR theories, Chemometr. Intell. Lab. Syst. 149 (2015) 117–122.
[21] K. Yu, X. Wang, Z. Wang, Self-adaptive multi-objective teaching-learning-based optimization and its application in ethylene cracking furnace operation optimization, Chemometr. Intell. Lab. Syst. 146 (2015) 198–210.
[22] S. Salcedo-Sanz, J.A. Portilla-Figueras, E.G. Ortiz-Garcia, A.M. Perez-Bellido, R. Garcia-Herrera, J.I. Elorrieta, Spatial regression analysis of NO_x and O_3 concentrations in Madrid urban area using Radial Basis Function networks, Chemometr. Intell. Lab. Syst. 99 (2009) 79–90.
[23] O. Deeb, E.F.F. da Cunha, R.A. Cormanich, T.C. Ramalho, M.P. Freitas, Computer-assisted assessment of potentially useful non-peptide HIV-1 protease inhibitors, Chemometr. Intell. Lab. Syst. 116 (2012) 123–127.
[24] P. Cizek, Robust and efficient adaptive estimation of binary-choice regression models, J. Am. Stat. Assoc. 103 (2008) 687–696.
[25] H. Miao, H. Wu, H. Xue, Generalized ordinary differential equation models, J. Am. Stat. Assoc. 109 (2014) 1672–1682.
[26] S. Favaro, L.F. James, A note on nonparametric inference for species variety within Gibbs-type priors, Electron. J. Stat. 9 (2015) 2884–2902.
[27] M.N. Wahab, S. Nefti-Meziani, A. Atyabi, A comprehensive review of swarm optimization algorithms, PloS One 10 (5) (2015).
[28] R. Storn, On the usage of differential evolution for function optimization, NAFIPS 1996 Biennial Conf. North Am. Fuzzy Inf. Proc. Soc. 5 (1996) 519–523.
[29] D. Zaharie, Parameter adaptation in differential evolution by controlling the population diversity, in: Proceedings of the International Workshop on Symbolic and Numeric Algorithms for Scientific Computing, 2002, pp. 385–397.
[30] Q. Fan, Y. Zhang, Self-adaptive differential evolution algorithm with crossover strategies adaptation and its application in parameter estimation, Chemometr. Intell. Lab. Syst. 151 (2016) 164–171.
[31] J.C. Kiefer, Optimum experimental designs, J. Roy. Stat. Soc. B 21 (1959) 272–319.
[32] S.D. Silvey, Optimal Design, Chapman and Hall, 1980.
[33] H. Chernoff, Locally optimal designs for estimating parameters, Ann. Math. Stat. 24 (1953) 586–602.
[34] J.C. Kiefer, Optimum designs in regression problems, II, Ann. Math. Stat. 32 (1961) 298–325.
[35] V.V. Fedorov, Theory of Optimal Designs, Academic Press, 1972.
[36] K.M. Mullen, D. Ardia, D. Gil, D. Windover, J. Cline, DEoptim: an R package for global optimization by differential evolution, J. Stat. Software 40 (2011) 1–26.
[37] S.A. Arrhenius, Über die Dissociationswärme und den Einflu der Temperatur auf den Dissociationsgrad der Elektrolyte, Z. Phys. Chem. 4 (1889) 96–116.
[38] J. Stráská, J. Stráský, M. Jancek, Activation energy for grain growth of the isochronally annealed ultrafine grained magnesium alloy after hot extrusion and equal-channel angular pressing (EX-ECAP), Proc. Int. Symp. Phys. Mater. 128 (2015) 578–581.
[39] Z. Qin, S.K. Balasubramanian, W.F. Wolkers, J.A. Pearce, J.C. Bischof, Correlated parameter fit of Arrhenius model for thermal denaturation of proteins and cells, Ann. Biomed. Eng. 42 (2014) 2392–2404.
[40] I. Marsi, L. Seres, Determination of the Arrhenius parameters of the decomposition of azoisopropane: investigation of possible systematic errors via computer simulation, Chemometr. Intell. Lab. Syst. 50 (2000) 53–61.
[41] H.H. Lippmann, B. Jesser, U. Schurath, The rate constant of $NO+O_3 \rightarrow NO_2+O_2$ in the temperature range of 283-443K, Int. J. Chem. Kinet. 7 (1980) 547–554.
[42] Jet Propulsion Laboratory, Chemical Kinetics and Photochemical Data for Use in Atmospheric Studies, California Institute of Technology, NASA, California, February 2003. Evaluation Number 14, Pasadena.
[43] L.J. Rodríguez-Aragóna, J. López-Fidalgo, Optimal designs for the Arrhenius equation, Chemometr. Intell. Lab. Syst. 77 (2005) 131–138.
[44] G.W. Ray, R.T. Watson, Kinetics of the reaction $NO+O_3 \rightarrow NO_2+O_2$ from 212 to 422 K, J. Phys. Chem. 85 (1981) 1673–1676.

[45] J.M. Rodrígue -Díaz, M.T. Santos-Martín, Study of the best designs for modifications of the Arrhenius equation, Chemometr. Intell. Lab. Syst. 95 (2008) 199–208.
[46] A.C. Atkinson, B. Bogacka, Compound D- and D_S-optimum designs for determining the order of a chemical reaction, Technometrics 39 (1997) 347–356.
[47] S. Furlanetto, M. Cirri, G. Piepel, N. Mennini, P. Mura, Mixture experiment methods in the development and optimization of microemulsion formulations, J. Pharmaceut. Biomed. Anal. 55 (2011) 610–617.
[48] H. Scheffé, Experiments with mixtures, J. Roy. Stat. Soc. B 20 (1958) 344–360.
[49] D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Parallel differential evolution, in: Proceedings of the 2004 Congress on Evolutionary Computation, 2004, pp. 2023–2029.
[50] P.-Y. Chen, R.-B. Chen, H.-C. Tung, W.K. Wong, Standardized maximum D-optimal designs for enzyme kinetic inhibition models, Chemometr. Intell. Lab. Syst. 169 (2017) 79–86.
[51] B. Bogacka, M. Patan, P.J. Johnson, K. Youdim, A.C. Atkinson, Optimum design of experiments for enzyme inhibition kinetic models, J. Biopharm. Stat. 21 (3) (2011) 555–572.
[52] A. Ruseckaite, P. Goos, D. Fok, Bayesian D-optimal choice designs for mixtures, Appl. Stat. 66 (2017) 363–386.
[53] J. Zhang, A.C. Sanderson, JADE: self-adaptive differential evolution with fast and reliable convergence performance, in: 2007 IEEE Congress on Evolutionary Computation, Singapore, vol. 2007, 2007, pp. 2251–2258.
[54] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: 2005 IEEE Congress on Evolutionary Computation 2, Edinburgh, Scotland, 2005, pp. 1785–1791.
[55] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 71–78.
[56] T. Robic, B. Filipic, DEMO: differential evolution for multiobjective optimization, in: International Conference on Evolutionary Multi-Criterion Optimization EMO 2005: Evolutionary Multi-Criterion Optimization, 2005, pp. 520–533.
[57] W. Long, X. Liang, Y. Huang, Y. Chen, A hybrid differential evolution augmented Lagrangian method for constrained numerical and engineering optimization, Comput. Aided Des. 45 (2013) 1562–1574.
[58] I. Saha, U. Maullik, M. Lukasik, D. Plewczynski, Multiobjective differential evolution: a comparative study on benchmark problems, Man-Mach. Interact. 3 (2014) 529–536.
[59] B. Chen, W. Zeng, Y. Lin, W. Zhong, An enhanced differential evolution based algorithm with simulated annealing for solving multiobjective optimization problems, J. Appl. Math. 2014 (2014).
[60] X. Yu, J. Cao, H. Shan, L. Zhu, J. Guo, An adaptive hybrid algorithm based on Particle swarm optimization and differential evolution for global optimization, Sci. World J. 2014 (2014).
[61] O. Sahin, B. Akay, Comparisons of metaheuristic algorithms and fitness functions on software test data generation, Appl. Soft Comput. 49 (2016) 1202–1214.
[62] Weinan Xu, W.K. Wong, K.C. Tan, J.X. Xu, Finding high-dimensional D-optimal designs for logistic models via differential evolution, IEEE Access 7 (2019) 7133–7146.