Fuzzy Self-Tuning Differential Evolution for Optimal Product Line Design

Tsafarakis, Stelios; Zervoudakis, Konstantinos; Andronikidis, Andreas; Altsitsiadis, Efthymios

Document Version
Accepted author manuscript

Published in:
European Journal of Operational Research

DOI:
10.1016/j.ejor.2020.05.018

Publication date:
2020

Fuzzy Self-Tuning Differential Evolution for Optimal Product Line Design

ABSTRACT

Designing a successful product line is a critical decision for a firm to stay competitive. By offering a line of products, the manufacturer can maximize profits or market share through satisfying more consumers than a single product would. The optimal Product Line Design (PLD) problem is classified as NP-hard. This paper proposes a Fuzzy Self-Tuning Differential Evolution (FSTDE) for PLD, which exploits Fuzzy Logic to automatically calculate the parameters independently for each solution during the optimization, thus resulting to a settings-free version of DE. The proposed method is compared to the most successful mutation strategies of the algorithm as well as previous approaches to the PLD problem, like Genetic Algorithm and Simulated Annealing, using both actual and artificial data of consumer preferences. The comparison results demonstrate that FSTDE is an attractive alternative approach to the PLD problem.

Keywords: OR in Marketing, Product line design, Differential Evolution, Self-tuning, Fuzzy logic

1 Introduction

Introduction of new products or redesigning existing ones, is one of the key decision areas that a product development team has to deal with, in order to maintain the sustainability and profitability of a firm. However, such processes can be uncertain and expensive (Kotler & Armstrong, 2012). As a result, the potential success of a new product needs to be estimated, before its production process begins. Such estimations are achieved by addressing the optimal product design problem, which is usually formulated in the context of Conjoint Analysis and has been studied by researchers for more than 40 years. A product line is a set of several related products with similar structures but different configurations or product attributes. The purpose of the product line development is to satisfy a variety of consumers. Its objective usually is the maximization of the total firm's profit or products' market share. Kohli and Krishnamurti (1989) proved that the product line design problem

(PLD) belongs to the class of NP-hard combinatorial optimization problems, thus no algorithm can verify in tractable time that it identifies the global optimum of the problem (Papadimitriou & Steiglitz, 1982). Hence, different optimization mechanisms have been proposed in an attempt to provide (near) optimal solutions to the PLD problem (for a review see Tsafarakis & Matsatsinis, 2010).

In this paper, we propose a Fuzzy Self-Tuning Differential Evolution (FSTDE) as an alternative approach to the optimal PLD problem. Because this is the first reported application of the Differential Evolution (DE) optimization method (Storn & Price, 1997) to the problem, we first explore the best values for the algorithm's tuning parameters through statistical analysis. Although DE is one of the most powerful stochastic optimization algorithms, its performance strongly depends on the proper settings of its parameters, hence we subsequently develop the Fuzzy Self-Tuning Differential Evolution (FSTDE) algorithm, which calculates the settings of parameters independently for each solution during the optimization process, using an automatic FL-based methodology. FSTDE as well as the most commonly used mutation strategies of DE are applied to the PLD problem and their performance is compared to those of previous approaches, like Genetic Algorithm (GA) and Simulated Annealing (SA), using both actual and artificial consumer-related data preferences using the datasets from Belloni et al. (2008).

The rest of the paper is organized into 6 sections as follows: Section 2 provides a brief description of the optimal PLD problem, while in Section 3 the algorithmic structure of the original DE is described. In Section 4, we describe the problem formulation, we explore how sensitive is original DE to the problem depending on its initial parameters and the FSTDE is presented. In Section 5 we evaluate the effectiveness of the FSTDE and other DE approaches through a comparison of their performance with that of GA and SA. Finally, Section 6 provides an overview of the main conclusions of the study and future research areas are suggested.

2 The optimal product line design problem

The optimal (single) product design problem was originally introduced by Zufryden (1977), and a few years later, the optimal PLD problem was addressed by Green and Krieger (1985). In PLD, firms aim at introducing a number of products, while optimizing

a specific objective, such as profit or market share. Each product is usually represented as a set of attributes (characteristics), each one taking specific levels. A camera for example, consists of the attributes *picture resolution*, *video resolution*, and *water resistance*, which take the levels 10 MP, 14 MP or 20 MP, HD, Full HD or 4K Ultra HD, Yes or No, respectively. As a result, a consumer may select a product according to its attributes that satisfy his/her needs, hence firms need to be aware of consumer preferences. Conjoint analysis (Luce & Tukey, 1964), is a widely known method for customer preference measurement, which generates the perceived utility value of each consumer for each level of a product's attributes. These values, the so called "parthworths", are used along with choice models to find a product line that optimizes a specific objective.

Since the PLD problem is NP-hard, different optimization approaches have been proposed to provide near optimal solutions in tractable time, the most important being Dynamic Programming (Kohli & Sukumar, 1990), Beam Search (Nair, Thakur, & Wen, 1995), Lagrangian Relaxation with Branch and Bound (Belloni et al., 2008; Camm, Cochran, Curry, & Kannan, 2006) and Simulated Annealing (Belloni et al., 2008; Tsafarakis, 2016). Nature-inspired and evolutionary-based optimization algorithms have also been introduced to the problem. For example, Alexouda and Paparrizos (2001) dealt with the PLD problem by implementing a GA, which was initialized in two different ways. In the first way the GA was initialized using a random population, while in the second way, the solution of the Beam Search method was included in the initial population of the GA. They tested their approaches in varying degrees of problem sizes using artificial data and the results supported the substantially better performance of their first approach. It was the first attempt to show that GAs have significant potential to solve PLD problems, followed by others that confirmed the finding (Balakrishnan, Gupta, & Jacob, 2004; Belloni et al., 2008; Steiner & Hruschka, 2003).

Ant Colony Optimization (ACO) is another evolutionary method which has been also applied to the PLD problem by Albritton and McMullen (2007). When compared to complete enumeration of all possible solutions, ACO was found to generate near-optimal results for this problem. Finally, Particle Swarm Optimization (PSO) was also introduced to the particular problem by Tsafarakis et al., (2011). The authors proposed a population-based algorithm and employed a Monte Carlo simulation to compare its performance to

that of GAs. The results revealed that PSO constitutes an attractive alternative because its performance is comparable to that of GAs concerning the best solution found, while it outperforms GAs regarding the diversity of the final set of provided solutions. Saridakis et al. (2015) verified the performance of PSO on designing optimal car lines.

The present study extends previous research in three important ways. First, we apply the original DE algorithm for the first time in the PLD literature and the broader area of marketing research. Second, a novel Fuzzy Self-Tuning Differential Evolution (FSTDE) algorithm is introduced, which calculates the settings of parameters independently for each solution during iterations, using an automatic FL-based methodology. Third, from the foregoing analysis, it is verified that except for continuous optimization problems, DE variants like the FSTDE perform with great success on combinatorial problems as well.

3 Differential Evolution

The original DE algorithm was introduced by Storn and Price (1997) as a new heuristic approach for optimizing continuous space functions. It belongs to the class of Evolutionary Algorithms (EAs) like GAs. Evolution is the process of improving the survival capabilities through mechanisms such as natural selection, survival of the fittest, reproduction, mutation, competition and symbiosis (Engelbrecht, 2007). Even though DE is a continuous parameter optimizer, numerous attempts to modify and use DE for optimizing binary and combinatorial problems are reported in the literature. Baioletti, Milani, and Santucci, (2018) for example, introduced a novel differential evolution algorithm for learning the structure of a Bayesian Network, while Ali, Essam, and Kasmarik (2018) introduced a novel differential evolution algorithm, which incorporates several effective components to increase search effectiveness by providing a good balance between exploration and exploitation processes, when performing on 0–1 Knapsack Problems. Santucci, Baioletti, Di Bari, and Milani (2019) introduced a Memetic Algebraic Differential Evolution algorithm for the Binary search space, while Ali et al. (2019) proposed a novel mapping method for DE which maps continuous variables to discrete ones and directs them towards optimality.

In DE, like most EAs, a population of individuals is generated (most of the times randomly to achieve a high diversity), where each individual represents a potential solution

to the problem, with an objective function evaluating the solution's performance. During the iterative procedure, a set of operators known as mutation and crossover is applied.

DE has been extensively applied in optimization problems, because as a stochastic direct search method, it handles complex objective functions and it is easy to program, since it requires very few control parameters (population size, scale factor, and crossover probability). According to Lampinen and Storn (2004) it displays good convergence capabilities and has a high probability of finding optimal solutions.

3.1 Mutation

During each iteration of DE, a mutant vector v_i^t is created. The five most commonly used mutation strategies are listed below (Das et al., 2016):

$$\text{DE/rand/1: } v_i^t = x_{R_{1i}}^t + F\left(x_{R_{2i}}^t - x_{R_{3i}}^t\right) \tag{1}$$

$$\text{DE/best/1: } v_i^t = x_{best}^t + F\left(x_{R_{1i}}^t - x_{R_{2i}}^t\right) \tag{2}$$

$$\text{DE/current-to-best/1: } v_i^t = x_i^t + F(x_{best}^t - x_i^t) + F\left(x_{R_{1i}}^t - x_{R_{2i}}^t\right) \tag{3}$$

$$\text{DE/best/2: } v_i^t = x_{best}^t + F\left(x_{R_{1i}}^t - x_{R_{2i}}^t\right) + F\left(x_{R_{3i}}^t - x_{R_{4i}}^t\right) \tag{4}$$

$$\text{DE/rand/2: } v_i^t = x_{R_{1i}}^t + F\left(x_{R_{2i}}^t - x_{R_{3i}}^t\right) + F\left(x_{R_{4i}}^t - x_{R_{5i}}^t\right) \tag{5}$$

The vectors x_{R_1} to x_{R_5} are randomly chosen solution vectors from the current population, which differ from the current solution vector x_i and are randomly generated for each solution vector. F is a positive mutation control parameter for scaling the difference vectors and x_{best} is the best solution vector found so far.

Strategies using the best solution found so far as DE/best/1, DE/best/2 and "DE/current-to-best/1," usually display fast convergence speed. As a result, while they perform well when addressing unimodal objectives, they are more likely to get stuck at a local optima points when performing on multimodal objectives. On the contrary, the DE/rand/1 strategy usually exhibits slow convergence speed and better exploration capability. Consequently, it may perform better when performing on multimodal problems compared to the strategies relying on the best solution found so far. The two-difference-vectors-based DE/best/2,

DE/rand/2 and DE/current-to-best/1 strategies may result in better perturbation than one-difference-vector-based strategies (Qin, Huang, & Suganthan, 2009).

One of the many attempts that have been made to improve DE's performance is the *dither* method, in which F varies during iterations. According to Price et al. (2005), in this method F is randomized according as:

$$F = F_{low} + rand * \left(F_{high} - F_{low}\right) \tag{6}$$

where F_{low} and F_{high} are the highest and lowest values of F respectively, and $rand$ is a uniform random number in the range of [0, 1]. The F can be either randomized in each generation of the algorithm or for each target vector of the population in a specific generation.

3.2 Crossover

Through crossover, the donor vector mixes its components with the target vector x_i^t to form the offspring vector $u_i^t = \left(u_{i,1}^t, u_{i,2}^t, \dots, u_{i,d}^t\right)$ according to a predefined probability. According to Das et al. (2016), binomial crossover can be described as:

$$u_{i,j}^t = \begin{cases} v_{i,j}^t, & if\ j = k\ or\ rand_{i,j} \le Cr \\ x_{i,j}^t, & otherwise \end{cases} \tag{7}$$

where k is randomly chosen in $\{1,2, \dots, d\}$, $rand_{i,j}$ is a random number in the range [0,1] and $j = k$ guarantees that at least one component from v_i^t is chosen by u_i^t, to ensure that the new solution does not duplicate the original one. Finally, Cr is the predefined crossover probability.

3.3 Selection operation

Selection determines whether the target (parent) or the trial (offspring) vector survives to the next generation. The selection operation for maximization problems is described as:

$$x_i^{t+1} = \begin{cases} v_i^t, & f(v_i^t) \ge f(x_i^t) \\ x_i^t, & otherwise \end{cases} \tag{8}$$

where $f(v_i^t)$ and $f(x_i^t)$ are the function values of v_i^t and x_i^t, respectively.

3.4 Working steps of DE

According to Zhou et al. (2016) the working steps of DE are the follows:

Step 1: Initialization. Initialization of parameters and solution vectors. Evaluation of each solution vector.

Step 2: Mutation. Mutant vectors are generated using one of the mutation operations (1-5).

Step 3: Crossover. Candidate vectors are generated using the crossover operation (7).

Step 4: Selection for next iteration. Determine vectors for the next iteration's population using formula (8).

Step 5: If stopping criteria are met, return the individual with the best fitness found. If not, continue to the next iteration (Step 2).

4 Applying FSTDE and other DE variants to the PLD problem

This section describes the formulations of the PLD problem on which FSTDE and other DE variants are evaluated. All the algorithms have been programmed with the use of the MATLAB platform. The simulations have been carried out on an i5 3.3 GHz desktop computer, with 8 GB of RAM and a 64-bit operating system. Furthermore, the Pearson's correlation coefficient (r) is used to investigate the bivariate correlation level between the parameters and the average values obtained from the results of DE algorithm after 10 runs.

4.1 Problem formulation

Following Belloni et al. (2008), the first problem concerns an actual PLD problem faced by Timbuk2, a manufacturer of school bags. A conjoint study that focused on price and nine binary product features was conducted by a group of academic researchers in cooperation with the company (additional details can be found in Toubia et al., 2003).

After the market survey, the preferences of 324 consumers on the ten characteristics of each bag are known. The first feature is the price that can take seven different levels ($70, $75, $80, $85, $90, $95 and $100). The remaining nine features are yes / no values (exists / does not exist). In addition, the company provides the cost of each feature.

According to Belloni et al. (2008), the number of possible products that can be designed by combining these features is 3,584, of which 4.9×10^{15} different product lines consisting of five products can be created. The profit for each of these 3,584 products is equal to its

sale price, minus the cost of its features and a fixed production cost of $35. Table 1 presents the marginal cost of each feature, as well as its average part-worth among the 324 consumers.

Table 1: Incremental marginal cost and average part-worth of each feature

Feature	Average part-worth	Incremental marginal cost ($)
$5 Price increase	-7.6	-5.00
Large size	17.9	3.50
Red color (not black)	-36.0	0.00
School logo	9.0	2.00
Handle	37.7	3.50
Gadget holder	5.2	3.00
Cell phone holder	5.5	3.00
Mesh pocket	9.7	2.00
Velcro flap	18.2	3.50
Reinforcing boot	24.4	4.50

Using the first choice rule (maximum utility) the preferred product among the firm's five bags and three competitive ones is identified for each consumer.

By summing up the earnings from the sales of each bag, we calculate the earnings of the product line, which constitutes the objective function of the problem. Through running a Lagrangian relaxation with branch and bound for one week computational time, Belloni et al. (2008) found the global optimum solution, which is a product line of five bags with $12226 predicted earnings.

4.2 Application of the original DE to the Product Line Design Problem

To design a product line of five bags that maximizes the predicted earnings, we apply the original DE algorithm to the data set obtained by Belloni et al. (2008). One of the most critical issues when developing a DE algorithm is the solution representation. DE operates in continuous spaces, while our application consists of a discrete domain of solutions, since all attributes take a limited number of discrete levels. Hence, a rule must be employed that coverts the real vector produced by the algorithm to the discrete domain required by the problem. There are two ways of presenting a solution, a binary representation and an integer one. In the binary representation, each vector's element represents the level of an attribute. The element that corresponds to the selected level for each attribute takes a value

of 1, while the rest of the levels take the value zero. In the integer representation, each element corresponds to an attribute and its value represents the attribute's level. However, even though integer representation requires smaller vector lengths and less computational time, binary representation have shown better performance in similar problems (Tsafarakis et al., 2011).

Using the binary representation scheme, the length of each vector (total number of attribute levels) is 25. Furthermore, a rule which ensures that within each attribute only a single element takes the value of 1 must be applied. For instance, as shown in Table 2, a large red bag, priced at $85, with handle and Cell phone holder, would be represented as [0001000 01 01 10 01 10 01 10 10 10]:

Table 2: Representation of a random bag configuration

Price ($)							Large size		Red color		School logo		Handle		Gadget holder		Cell phone holder		Mesh pocket		Velcro flap		Reinforcing boot	
70	75	80	85	90	95	100	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
0	0	0	1	0	0	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0

Since DE operates in continuous spaces, a representation with four-decimal points of a potential vector for a single product could be: y = [0.1269 0.5468 0.9571 -0.5007 0.8491 0.3922 0.2769 -0.3419 0.3958 0.6463 0.6550 0.9831 0.5059 0.7241 0.8142 0.2510 0.5852 0.7537 1.3449 0.4693 0.3112 0.6540 0.2289 0.9961 0.0046]. To convert a continuous vector to the required binary representation, the modified Smallest Position Value (SPV) rule was used, which has shown the best performance in the optimal product line design problem, even though it is the most time consuming (Tsafarakis et al., 2011). According to the SPV rule, the element with the smallest value on each attribute takes a value of 1, while the rest take a value of 0. The SPV rule performs great and it is widely used because it allows an algorithm to search in the continuous space, without setting any boundaries, or using any rounding off procedures (e.g. dropping the sign and fractional part of a real number, or truncate it to the nearest integer), which may lead to suboptimal solutions. SPV has already been successfully applied to DE in Tasgetiren et al. (2009). By applying the rule in the case of vector y we have: y = [0.1269 0.5468 0.9571 **-0.5007** 0.8491 0.3922 0.2769 | **-0.3419** 0.3958 | **0.6463** 0.6550 | 0.9831 **0.5059** | **0.7241** 0.8142 | **0.2510** 0.5852 |

0.7537 1.3449 | 0.4693 **0.3112** | 0.6540 **0.2289** | 0.9961 **0.0046**], which is converted to y = [0001000 10 10 01 10 10 10 01 01 01].

4.3 Adjustment of Differential Evolution parameters

To adjust the DE parameters, we tested different values for population size, F and Cr, using DE/rand/1, which is the most commonly used strategy (Zhou et al., 2016). Initially, we tested the performance of the algorithm by changing the population size from 30 to 100 with a step size of 10. Each population size was tested for 10 runs. The algorithm runs until 100,000 function evaluations are reached. The results are presented in Figure 1.

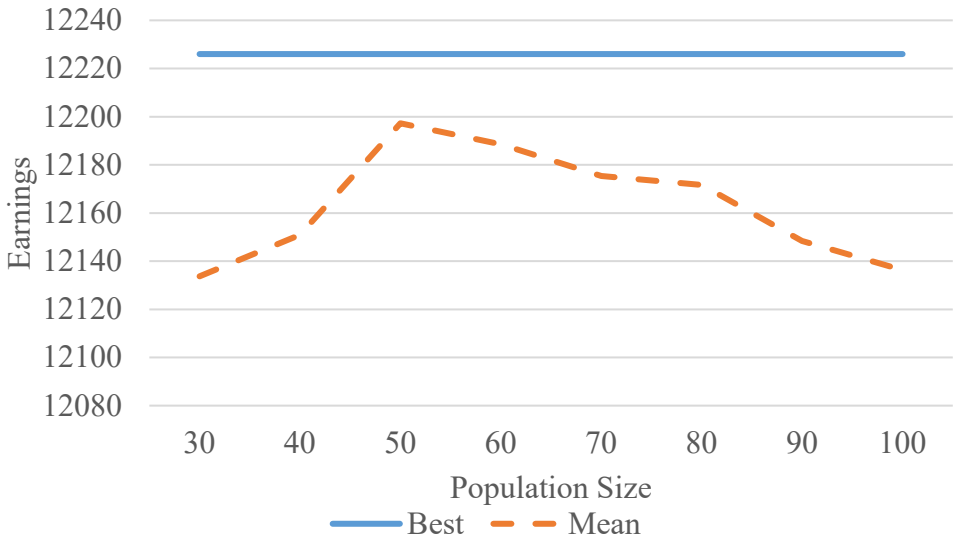

Figure 1. Results of DE with different values of population size

Regarding the bivariate correlation level between the population size values and the mean of the DE's results after 10 runs, no correlation was observed ($r = -.10, p = .81$). As a result, there is no relation between the values of population size and DE's results, hence we decided to use a population size of 50, because it was found that the algorithm needed less function evaluations (about 60,000 in total) to converge than the rest sizes.

Moreover, we tested the performance of DE by changing the mutation control parameter F from 0.1 to 0.9 with a step size of 0.1. Each value was tested for 10 runs. The results are demonstrated in Figure 2.

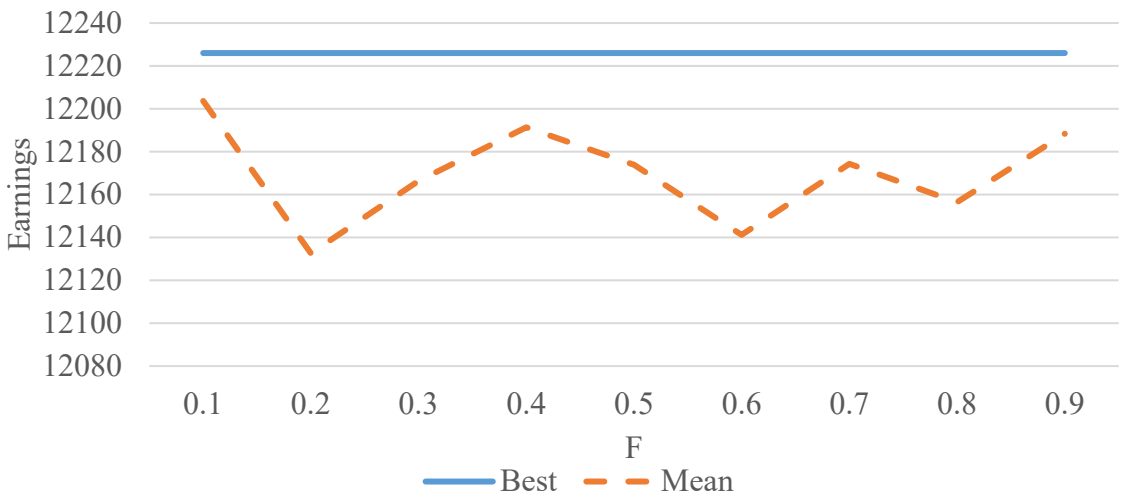

Figure 2. Results of DE with different values of F

Regarding the bivariate correlation level between the F values and the mean of the results given by DE after 10 runs, no correlation was observed ($r = -.01, p = .98$). As a result, there is no relation between the values of F and DE's results. For that reason, we decided to use the *dither* version (6), with an F_{low} of 0.1 and an F_{high} of 0.9.

Finally, we tested the performance of DE by changing the probability Cr from 0.01 to 0.99 with a step size of 0.01. Each value was tested for 10 runs. The results are demonstrated in Figure 3.

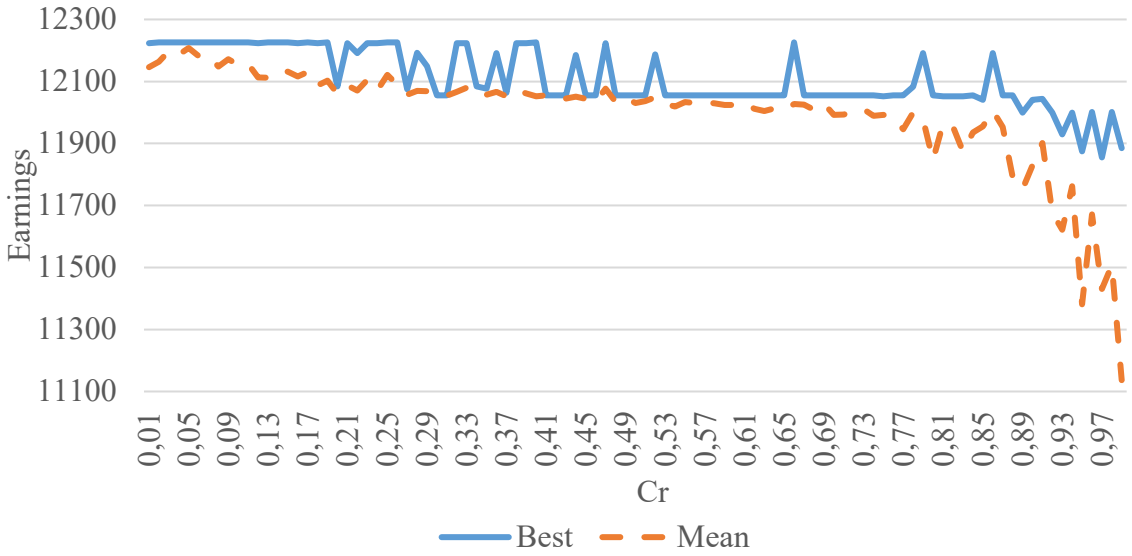

Figure 3. Results of DE with different values of Cr

Regarding the bivariate correlation level between the Cr values and the mean of the results given by DE after 10 runs, negative strong correlation was observed ($r = -.76$), which means that DE performs better with small values of Cr. Consequently, we chose the $Cr = 0.05$, since it achieved the highest performance of both best and mean values.

4.4 Fuzzy Self-Tuning DE

Even though in Subsection 4.3 we explored the best values for the algorithm's parameters, the performance of the original DE is not only highly dependent on its parameter settings, but also on the dataset. As a result, DE may not be as effective using the same parameters when performing on different datasets. To overcome this, an automatic parameter tuning method can be used, such as the self-adaptive DE (Al-Anzi & Allahverdi, 2007; Fan, Yan, & Zhang, 2018; Salman, Engelbrecht, & Omran, 2007; Zhao et al., 2016). In this subsection the fully-automated version of DE, called Fuzzy Self-Tuning DE (FSTDE) is described, where the values of DE settings are dynamically controlled by means of FL. According to literature, FL is already applied to various metaheuristics that suffer from the parameters tuning, which is considerably dependent on the problem (Noorbin & Alfi, 2018; Olivas, Valdez, Castillo, & Melin, 2018).

To determine the population size, FSTDE exploits the heuristic $N = \left[10 + 2\sqrt{number\ of\ bags \cdot \text{total number of attribute levels}}\right]$ which sets the value of population size according to the number of dimensions of the search space, as suggested by Nobile et al. (2018).

To dynamically determine the values of DE's parameters in an automatic way, independently for each solution, we make use of a Fuzzy Rule-Based System (FRBS) consisting of nine fuzzy rules, reported in Table 3. These rules are based on the distance of each solution from the best solution found so far, and a function measuring the fitness improvement of each solution with respect to the previous iteration.

Since we dynamically determine the values of DE's parameters, we decided to use a combination of DE/rand/1 and DE/current-to-best/1 mutation strategies defined as:

$$\text{FSTDE: } v_i^t = x_{R_{1i}}^t + F_1\left(x_{R_{2i}}^t - x_{R_{3i}}^t\right) + F_2\left(x_{best}^t - x_{R_{4i}}^t\right) \tag{9}$$

The purpose of this combination is to take advantage of the better exploration capability of DE/rand/1, along with the fast convergence speed of DE/current-to-best/1, as mentioned in Subsection 3.1, depending on the distance of each solution from the best solution found so far, and the solution's improvement with respect to the previous iteration.

As Nobile et al. (2018) use in their research, the distance between two solutions is calculated as:

$$\delta\left(\mathbf{x}_i^t, \mathbf{x}_j^t\right) = \left\|\mathbf{x}_i^t - \mathbf{x}_j^t\right\| = \sqrt{\sum_{k=1}^{M}\left(x_{i,k}^t - x_{j,k}^t\right)^2} \tag{10}$$

where $x_{i,k}^t$ and $x_{j,k}^t$ denote the k-th component of the position vectors $\mathbf{x}_i^t$ and $\mathbf{x}_j^t$, respectively.

The normalized fitness incremental factor $\varphi: \mathbb{R}^M \times \mathbb{R}^M \to [-1,1]$, considers the positions of solution i at the current and previous iterations:

$$\varphi(\mathbf{x}_i^t, \mathbf{x}_i^{t-1}) = -\frac{\delta(\mathbf{x}_i^t, \mathbf{x}_i^{t-1})}{\delta_{max}} \cdot \frac{\min\{f(\mathbf{x}_i^t), f_\Delta\} - \min\{f(\mathbf{x}_i^{t-1}), f_\Delta\}}{|f_\Delta|} \tag{11}$$

where δ_{max} is the length of the diagonal of the hyper-rectangle corresponding to the search space, while f_Δ represents the worst fitness value found so far (for more details see: Nobile et al., 2018). The main purpose of δ is to characterize the proximity to the global best, while the purpose of φ is to characterize the improvement of a solution with respect to the fitness value it assumed in the previous iteration. During each iteration, each solution compute independently their own values for δ and φ, which are used to calculate the output variables according to the rules reported in Table 3.

According to Nobile et al. (2018), the variable of δ corresponds to the interval $[0, \delta_{max}]$. The term set of this variable is composed by three linguistic values, Same, Near and Far. The membership function of Same is: a) 1, if $0 \leq \delta < \delta_1$, b) $\frac{(\delta_2 - \delta)}{(\delta_2 - \delta_1)}$, if $\delta_1 \leq \delta < \delta_2$, and c) 0, if $\delta_2 \leq \delta < \delta_{max}$. The membership function of Near is: a) 0, if $0 \leq \delta < \delta_1$, b) $\frac{(\delta - \delta_1)}{(\delta_2 - \delta_1)}$, if $\delta_1 \leq \delta < \delta_2$, c) $\frac{(\delta_3 - \delta)}{(\delta_3 - \delta_2)}$, if $\delta_2 \leq \delta < \delta_3$, and d) 0, if $\delta_3 \leq \delta < \delta_{max}$. The membership function of Far is: a) 0, if $0 \leq \delta < \delta_2$, b) $\frac{(\delta - \delta_2)}{(\delta_3 - \delta_2)}$, if $\delta_2 \leq \delta < \delta_3$, and c) 1, if $\delta_3 \leq \delta < \delta_{max}$. The values of δ_1, δ_2 and δ_3 are set according to the size of the search space as: $\delta_1 = 0.2 \cdot \delta_{max}$, $\delta_2 = 0.4 \cdot \delta_{max}$ and $\delta_3 = 0.6 \cdot \delta_{max}$. As Nobile et al. (2018) mention in their

research, the general-purpose multipliers are created to avoid any overfitting to the benchmark function and implement a general and fuzzy concept of distance from the best solution found so far.

The variable of φ corresponds to the interval $[-1,1]$. The term set of this variable is composed by three linguistic values, Better, Same and Worse. The membership function of Better is: a) 1, if $\varphi = -1$, b) $-\varphi$, if $-1 < \varphi < 0$, and c) 0, if $0 \leq \varphi \leq 1$. The membership function of Same is: $1 - |\varphi|$, and the membership function of Worse is: a) 0, if $-1 \leq \varphi < 0$, b) φ, if $0 \leq \varphi < 1$, and c) 1, if $\varphi = 1$. However, since DE accepts a new solution only if it improves the previous one, the Worse scenario is not technically applied in this research.

Given a set of a specific number of R rules having the same output variable in their consequent (e.g., rules 1, 2, 3 in Table 3 for $F_{low,1}$ and $F_{high,1}$), the final numerical value of this output variable is calculated using the Sugeno method (Sugeno, 1985) as:

$$output = \frac{\sum_{r=1}^{R} \rho_r z_r}{\sum_{r=1}^{R} \rho_r} \tag{12}$$

where ρ_r denotes the membership degree of the input variable of the r-th rule, and z_r represents the output crisp value for the r-th rule, as given in Table 4.

Table 3: Fuzzy rules used by FSTDE

Rule no.	Rule definition
1	if (δ is *Far*) then ($F_{low,1}$ and $F_{high,1}$ are *Low)*
2	if (φ is *Same* or δ is *Same* or δ is *Near*) then ($F_{low,1}$ and $F_{high,1}$ are *Medium)*
3	if (φ is *Better*) then ($F_{low,1}$ and $F_{high,1}$ are *High)*
4	if (φ is *Better* or δ is *Near*) then ($F_{low,2}$ and $F_{high,2}$ are *Low)*
5	if (φ is *Same* or δ is *Same*) then ($F_{low,2}$ and $F_{high,2}$ are *Medium)*
6	if (δ is *Far*) then ($F_{low,2}$ and $F_{high,2}$ are *High)*
7	if (φ is *Same* or φ is *Better*) then (Cr is *Low)*
8	if (δ is *Same* or δ is *Near*) then (Cr is *Medium)*
9	if (δ is *Far*) then (Cr is *High)*

Table 4: Output variables and their defuzzification

Output Variable	Low	Medium	High
F_{low}	0.1	0.4	0.7
F_{high}	0.4	0.7	0.9
Cr	0.01	0.1	0.5

Rules 1, 2 and 3 control the strength of the difference of the two random vectors. If the solution is far from the best solution found so far (i.e. δ is *Far*), $F_{low,1}$ and $F_{high,1}$ are set to *Low,* because it is reasonable for it to "ignore" the information shared by two random solutions. On the contrary, when a better solution is found, it should rather "follow the advice" of the two random solutions to better explore the space nearby. An intermediate value is assigned to $F_{low,1}$ and $F_{high,1}$ if no relevant changes occur either in the fitness value or in the distance from the best solution (i.e., φ and δ are *Same* or δ is *Near*).

Rules 4, 5 and 6 control the strength of the attraction of a solution towards the best solution found so far. If either a better solution is found (i.e., φ is *Better*), or the best solution is being approached (i.e., δ is Near), then it is reasonable for it to "ignore" the information shared by the best solution found so far. Under these conditions, $F_{low,2}$ and $F_{high,2}$ are set to Low. On the contrary, when a better solution is not found or the solution is not close to the best solution found so far, it should rather "follow the advice" of the best solution found so far. Hence, we set $F_{low,2}$ and $F_{high,2}$ are set to High values. An intermediate value is assigned to $F_{low,2}$ and $F_{high,2}$ if no relevant changes occur either in the fitness value or in the distance from the best solution (i.e., φ and δ are Same).

Finally, rules 7, 8 and 9 control the Cr value. If a solution is far from the best solution found so far, then Cr is set to High in order to accept more values from the mutant vector. On the contrary, if the solution has any improvement or is the same, then Cr is set to Low. An intermediate value is assigned to Cr when the solution is near to the best solution found so far, or δ is *Same* in order help the algorithm escape local optima.

4.4.1 Determination of parameters by FSTDE

To demonstrate what happens when the proposed FSTDE determines the parameter values for the problem discussed in Subsection 4.3, we run the proposed FSTDE while tracking the calculated parameter settings independently for each solution during the optimization process. As a result, a set of values is created for each parameter (except the parameter of population size which was only calculated once in the beginning).

First, population size was set to 32, according to the heuristic described in Subsection 4.4. $F_{low,1}$ had an average value of 0.395, a standard deviation of 0.026 and a median value of 0.400 in the range of 0.100 to 0.443, while $F_{high,1}$ had an average value of 0.695, a standard deviation of 0.025 and a median value of 0.700 in the range of 0.550 to 0.729. Moreover, $F_{low,2}$ had an average value of 0.405, a standard deviation of 0.026 and a median value of 0.400 in the range of 0.100 to 0.550, while $F_{high,2}$ had an average value of 0.703, a standard deviation of 0.017 and a median value of 0.700 in the range 0.657 to 0.800. Finally, Cr had an average value of 0.064, a standard deviation of 0.044 and a median value of 0.055 in the range of 0.012 to 0.497.

As it was noticed, even though the value of population size differed to the value obtained in Subsection 4.3, the median value of Cr was really close to the value obtained in Subsection 4.3. Moreover, the FRBS tuning method managed to find more accurate values that were not checked during the tuning process in Subsection 4.3 (e.g. $Cr = 0.055$). As a result, the algorithm can successfully calculate its parameters according to the needs of each candidate solution.

To investigate any possible performance improvement or reduction using the FRBS tuning method, the proposed FSTDE was compared to all the mutation strategies mentioned in Subsection 3.1 that use the parameter settings mentioned in Subsection 4.3. The comparison results are demonstrated in Section 5.

5 Comparison

In this section, we compare the performance of FSTDE to that of the other DE approaches, as well as to that of GA and SA, which where the best performing methods in the study of Belloni et al. (2008). Particularly, FSTDE and all the mutation strategies mentioned in

Subsection 3.1 and 4.4, will be compared to the performance of GA and SA as they were configured by Belloni et al. (2008). At first, we use the real conjoint data set. Then, we test whether the model we have developed is affected by errors in consumer preference measurements, and whether the model can be generalized by changing the size of the problem. In our simulations, each algorithm runs until 70,000 function evaluations are reached, performing 50 replications for each case. Moreover, the Mann-Whitney test is used to compare differences between two optimization methods. Statistical analysis was performed using SPSS Version 21 statistic software package.

5.1 Real conjoint data set

Table 5 shows the results of the comparison of DE's (including FSTDE) performance using different mutation strategies, with that of GA and SA on the real conjoint data, while Figure 4 shows the convergence characteristic curves of them. Each DE strategy runs for about 20 seconds on average, while GA and SA need about 2 and 3.5 seconds on average, respectively. That verifies the findings of Tsafarakis et al. (2011) that the SPV mapping requires more computational time.

Table 5: Comparison of methods performing on the real conjoint data set

Statistics	FSTDE	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	DE/current-to-best/1	SA	GA
Best	**12226**	**12226**	**12226**	**12226**	**12226**	**12226**	11781.5	**12226**
Worst	**12055.5**	**12055.5**	12040.5	12032.5	12052.5	11961.5	10867	11895.5
Mean	**12202.1**	12201.38	12166.47	12148.13	12179.95	12099.57	11140.82	12083.62
Median	**12223.5**	**12223.5**	12208.75	12175	12218.5	12084	11140	12055.5
Sd.	**3.99E+01**	4.75e+01	6.84e+01	6.98e+01	6.39e+01	8.40e+01	1.91e+02	1.12e+02

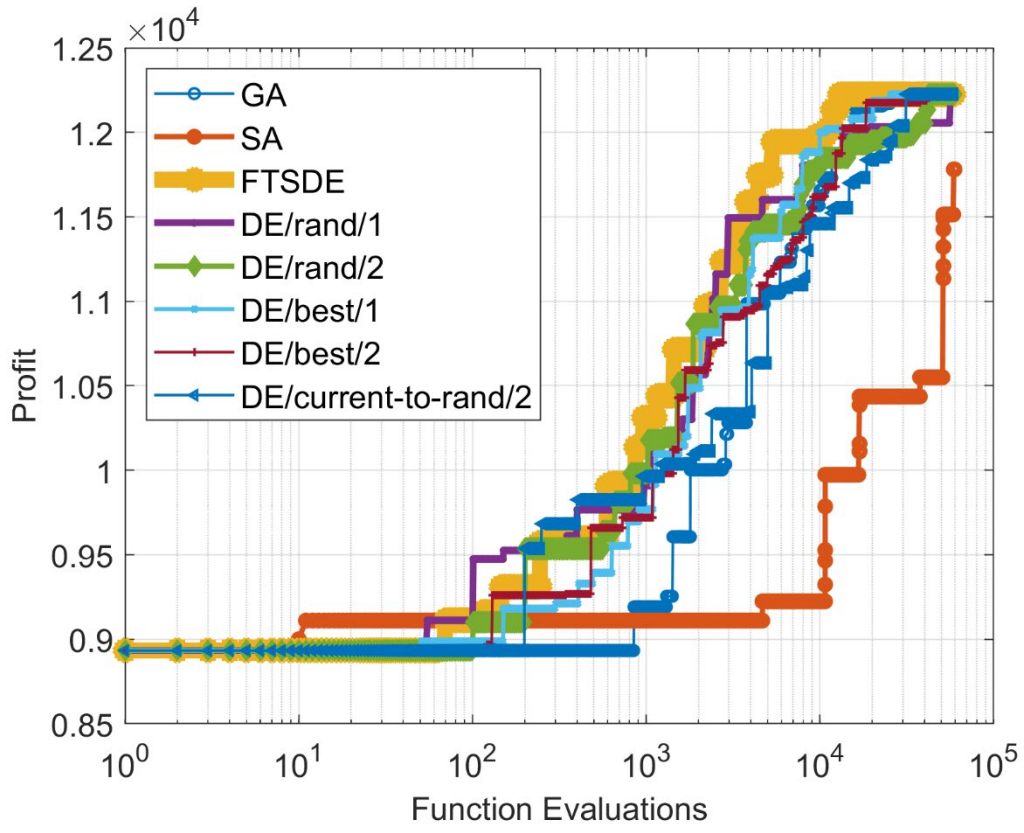

Figure 4. Convergence characteristic curves of the algorithms under comparison

The results obtained above indicate that superiority of all the DE mutation strategies over SA when performing on the real conjoint dataset. They also indicate that all the DE mutation strategies are really competitive to GA. According to Table 5, FSTDE is the most successful mutation strategy of DE.

Objective function values of FSTDE (*Mdn* = 12223.5) were higher than those of SA (*Mdn* = 11140). A Mann-Whitney test indicated this difference was statistically significant, $U(N_{FSTDE} = 50, N_{SA} = 50) = 0$, $z = -8.69$, $p < .001$. Moreover, objective function values of FSTDE (*Mdn* = 12223.5) were higher than those of GA (*Mdn* = 12055.5). A Mann-Whitney test indicated this difference was statistically significant, $U(N_{FSTDE} = 50, N_{GA} = 50) = 636$, $z = -4.29$, $p < .001$.

Moreover, Figure 4 demonstrates that FSTDE converges faster than all the other algorithms.

5.2 Robustness testing

The accuracy of the results does not only depend on the efficiency and effectiveness of the optimization method used to solve the problem, but also on other external factors. According to Belloni et al. (2008), one of the most important factors is the accuracy in the estimation of consumer preferences, that is the partworths of each feature, as estimated by conjoint analysis. To test the robustness of the methods in the presence of a measurement error during the conjoint analysis process, we repeated our analysis after perturbing the original partworth estimates, just like Belloni et al. (2008) did. These perturbations were accomplished by adding a (simulated) error to the partworths:

$$u'_{i,j} = u_{i,j} + \varepsilon_{i,j} \tag{13}$$

where $u_{i,j}$ is the original partworth for respondent i on product feature j, $\varepsilon_{i,j}$ is a zero-mean, independent normal error term which works differently across customers or attribute levels, and $u'_{i,j}$ is the perturbed part-worth.

We run each algorithm 50 times, obtaining 50 sets of perturbed partworths for each respondent. The perturbation terms are treated as measurement error. Under this interpretation, the original partworths that we analyzed in Table 1, is the only set of "true" partworths. However, we assume that the researcher can only observe the partworths that are subject to measurement error. The results are reported in Table 6.

Table 6: Comparison of methods performing under Measurement Error

Statistics	FSTDE	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	DE/current-to-best/1	SA	GA
Best	**12189**	**12189**	12073	12058	12077	12074	11311.5	12045
Worst	**11651.5**	11209.5	10920.5	10819.5	11185.5	11185.5	9813	11105
Mean	**12094.22**	11737.43	11662.33	11680.32	11734.49	11681.62	10711.38	11630.34
Median	**12187.5**	11754.25	11735.5	11722.25	11743	11673.75	10690.25	11676.5
Sd.	1.84e+02	**1.75e+02**	2.53e+02	2.49e+02	1.86e+02	2.11e+02	3.12e+02	2.17e+02
CP* (%)	**38**	37.6	36.8	36.4	37.6	36.8	23.2	30.8
CF* (%)	**86.31**	85.04	84.44	84.68	84.96	84.84	75.8	83.28

* CP and CF correspond to the common products and common features within the optimal line respectively.

From the results obtained from Table 6, we note that all the different mutation strategies of DE (especially the FSTDE strategy) are least affected by the measurement error on consumer preferences. On the contrary, SA is the most affected method.

Objective function values of FSTDE (*Mdn* = 12187.5) were higher than those of SA (*Mdn* = 10690.25). A Mann-Whitney test indicated this difference was statistically significant, $U(N_{FSTDE} = 50, N_{SA} = 50) = 0$, $z = -8.64$, $p < .001$. Moreover, objective function values of FSTDE (*Mdn* = 12187.5) were higher than those of GA (*Mdn* = 11676.5). A Mann-Whitney test indicated this difference was statistically significant, $U(N_{FSTDE} = 50, N_{GA} = 50) = 171$, $z = -7.65$, $p < .001$.

5.3 Results using simulated data

In this subsection, we also compare the performance of FSTDE and other DE variants to that of GA and SA, using simulated data. In each simulated problem, we design a product line consisting of 3 or 4 products, each composed of 3, 5, or 7 attributes that can take on 2, 3, 5, or 8 different levels. We assume that there are either 50 or 100 customers for each case. We select 12 different problem sizes as presented in Table 7 and 10 different replications for each size are generated, resulting in a total of 120 simulated data sets.

Table 7: Different problem sizes

Customers	Attributes	Levels	Products
50	3	5	4
100	3	5	4
50	5	3	4
100	5	3	4
50	7	2	4
100	7	2	4
50	3	8	3
100	3	8	3
50	5	5	3
100	5	5	3
50	7	3	3
100	7	3	3

The summary results are shown in Table 8, where each algorithm's average performance is shown as a percentage of the best solution found among all tested algorithms.

Table 8: Comparison of methods performing on simulated data

Statistics	FSTDE	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	DE/current-to-best/1	SA	GA
Average performance	99.85	99.68	**99.87**	97.17	97.04	97.28	90.90	99.09

From the comparison results, we notice that the DE/rand implementations exhibit the best performance.

To further test FSTDE and other DE variants in larger problem sizes, we increase specific dimensions of the above sets, as shown in Table 9.

Table 9. Problem with increased dimension sizes.

Customers	Attributes	Levels	Products
50	3	5	5
100	3	5	5
50	8	3	4
100	8	3	4
50	9	2	4
100	9	2	4
50	3	9	3
100	3	9	3
50	5	5	6
100	5	5	6
50	9	3	3
100	9	3	3

The comparison results of the methods are presented in Table 10.

Table 10: Comparison of methods performing on the problems presented in Table 9

Statistics	FSTDE	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	DE/current-to-best/1	SA	GA
Average performance	**99.31**	99.26	98.92	97.97	98.05	98.06	86.57	97.87

The results above indicate that all DE variants (especially the FSTDE strategy) are superior to both GA and SA as the problem sizes increased.

6 Conclusions

In this paper, a Fuzzy Self-Tuning Differential Evolution (FSTDE) along with various different mutation strategies of DE are applied on the PLD problem. The main idea of FSTDE is to overcome specific difficulties of DE when performing on different datasets using the same parameter settings. Because this is the first reported application of the DE algorithm to the problem, we explored the best values for the algorithm's tuning parameters through statistical analysis. Particularly, we showed that the values of mutation control parameter and population size parameter does not affect the results significantly, while DE works better with small values of the crossover probability. DE appeared to perform great with a population size of about 2 times the number of design variables, while GA requires a population size of 10 times the number of design variables when applied to the PLD problem.

The comparison results reveal that most of the times DE variants have superior performance to both SA and GA. Particularly, FSTDE appears to have a higher performance when performing on PLD, exhibiting a higher probability of finding a global optimum in less function evaluations. The results were verified using statistical analyses. FSTDE was found to be the most successful mutation strategy among all, when performing on PLD, followed by DE/rand/1 which verifies the statement of Qin et al. (2009) that the DE/rand/1 strategy is usually distinguished by better exploration capability. The convergence behavior of all DE's mutation strategies is also exceptional, since most of the times they reach the best overall solution in the early iterations. FSTDE was found to have high convergence capabilities compared to the rest, because of its ability to "follow the advice" of the best solution found so far when a better solution is not found, or the solution is not close to the best solution found so far. Moreover, while increasing the dimension sizes of the problem we noticed that DE's performance is superior over that of SA and GA in terms of accuracy and efficiency. Overall, DE constitutes a robust algorithm that has the potential to be generalized in large scale real world PLD problems of well-known firms, and therefore it is an attractive alternative solution process to the PLD problem.

An interesting area for future research is solving the PLD problem optimizing more than one objective at the same time. This requires the application of multiobjective optimization algorithms, instead of the single-objective algorithms that have been employed so far to

the problem. A multiobjective approach, along with a Pareto optimal analysis, will enable the firm to optimize two or more objectives (e.g., profit maximization and market share maximization) at the same time, while setting a number of constraints (e.g., cost of production not more than a specific value).

Acknowledgments

We wish to thank Prof. John Hauser, Olivier Toubia, and Matthew Selove for the provision of the datasets. We are also grateful for the support of the editor, Prof. José Fernando Oliveira, and the many comments of the anonymous reviewers, who helped much in improving this article.

References

Al-Anzi F.S, & Allahverdi, A., (2007). A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. *European Journal of Operational Research*, *182*(1), 80–94.

Albritton, M. D., & McMullen, P. R. (2007). Optimal product design using a colony of virtual ants. *European Journal of Operational Research*, *176*(1), 498–520. https://doi.org/10.1016/J.EJOR.2005.06.042

Alexouda, G., & Paparrizos, K. (2001). A genetic algorithm approach to the product line design problem using the seller's return criterion: An extensive comparative computational study. *European Journal of Operational Research*, *134*(1), 165–178. https://doi.org/10.1016/S0377-2217(00)00246-0

Ali, I. M., Essam, D., & Kasmarik, K. (2018). An Efficient Differential Evolution Algorithm for Solving 0-1 Knapsack Problems. *2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings*. https://doi.org/10.1109/CEC.2018.8477916

Ali, I. M., Essam, D., & Kasmarik, K. (2019). A novel differential evolution mapping technique for generic combinatorial optimization problems. *Applied Soft Computing Journal*, *80*, 297–309. https://doi.org/10.1016/j.asoc.2019.04.017

Baioletti, M., Milani, A., & Santucci, V. (2018). Learning bayesian networks with algebraic differential evolution. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11102 LNCS*, 436–448. https://doi.org/10.1007/978-3-319-99259-4_35

Balakrishnan, P. V., Gupta, R., & Jacob, V. S. (2004). Development of Hybrid Genetic Algorithms for Product Line Designs. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *34*(1), 468−483. https://doi.org/10.1109/TSMCB.2003.817051

Belloni, A., Freund, R., Selove, M., & Simester, D. I. (2008). Optimizing Product Line Designs: Efficient Methods and Comparisons. *Management Science*, *54*(9), 1544−1552. https://doi.org/10.1287/mnsc.1080.0864

Camm, J. D., Cochran, J. J., Curry, D. J., & Kannan, S. (2006). Conjoint Optimization: An Exact Branch-and-Bound Algorithm for the Share-of-Choice Problem. *Management Science*, *52*(3), 435−447. https://doi.org/10.1287/mnsc.1050.0461

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computation*, *27*, 1–30. https://doi.org/10.1016/J.SWEVO.2016.01.004

Engelbrecht, A. P. (2007). *Computational Intelligence: An Introduction*. Wiley.

Fan, Q., Yan, X., & Zhang, Y. (2018). Auto-selection mechanism of differential evolution algorithm variants and its application. *European Journal of Operational Research*, *270(2)*, 636−653. https://doi.org/10.1016/j.ejor.2017.10.013

Green, P. E., & Krieger, A. M. (1985). Models and Heuristics for Product Line Selection. *Marketing Science*, *4*(1), 1–19. https://doi.org/10.1287/mksc.4.1.1

Kohli, R., & Krishnamurti, R. (1989). Optimal product design using conjoint analysis: Computational complexity and algorithms. *European Journal of Operational Research*, *40*, 186−195. https://doi.org/10.1016/0377-2217(89)90329-9

Kohli, R., & Sukumar, R. (1990). Heuristics for Product-Line Design Using Conjoint Analysis. *Management Science*, *36*(12), 1464−1478. https://doi.org/10.2307/2661545

Kotler, P., & Armstrong, G. (2012). *Principles of Marketing* (12th ed.). New Jersey: Pearson/Prentice Hall.

Lampinen, J., & Storn, R. (2004). Differential Evolution. In G. C. Onwubolu & B. V. Babu (Eds.), *New Optimization Techniques in Engineering* (pp. 123–166). https://doi.org/10.1007/978-3-540-39930-8_6

Luce, R. D., & Tukey, J. W. (1964). Simultaneous conjoint measurement: A new type of fundamental measurement. *Journal of Mathematical Psychology*, *1*(1), 1–27. https://doi.org/10.1016/0022-2496(64)90015-X

Nair, S. K., Thakur, L. S., & Wen, K.-W. (1995). Near Optimal Solutions for Product Line Design and Selection: Beam Search Heuristics. *Management Science*, *41*(5), 767–785. https://doi.org/10.1287/mnsc.41.5.767

Nobile, M. S., Cazzaniga, P., Besozzi, D., Colombo, R., Mauri, G., & Pasi, G. (2018). Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization. *Swarm and Evolutionary Computation*, *39*, 70–85. https://doi.org/10.1016/j.swevo.2017.09.001

Noorbin, S. F. H., & Alfi, A. (2018). Adaptive parameter control of search group algorithm using fuzzy logic applied to networked control systems. *Soft Computing*, *22*(23), 7939–7960. https://doi.org/10.1007/s00500-017-2742-0

Olivas, F., Valdez, F., Castillo, O., & Melin, P. (2018). *Dynamic Parameter Adaptation for Meta-Heuristic Optimization Algorithms Through Type-2 Fuzzy Logic*. https://doi.org/10.1007/978-3-319-70851-5

Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorial optimization : algorithms and complexity*. Upper Saddle River, NJ: Prentice Hall.

Price, K. V., Storn, R., & Lampinen, J. A. (2005). *Differential evolution : a practical approach to global optimization*. Retrieved from https://dl.acm.org/citation.cfm?id=1121631

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, *13*(2), 398–417. https://doi.org/10.1109/TEVC.2008.927706

Santucci, V., Baioletti, M., Di Bari, G., & Milani, A. (2019). A binary algebraic differential evolution for the multidimensional two-way number partitioning problem. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial*

Intelligence and Lecture Notes in Bioinformatics), *11452 LNCS*, 17–32. https://doi.org/10.1007/978-3-030-16711-0_2

Salman, A., Engelbrecht, A.P., Omran M.G.H. (2007). Empirical analysis of self-adaptive differential evolution. *European Journal of Operational Research*, *40*, 186−195. https://doi.org/10.1016/j.ejor.2006.10.020

Saridakis, C., Tsafarakis, S., Delias, P., Baltas, G., & Matsatsinis, N. (2015). Optimizing differentiation and commonality levels among models in car line-ups: An empirical application of a nature-inspired heuristic mechanism. *Expert Systems with Applications*, *42*(5), 2323–2335. https://doi.org/10.1016/j.eswa.2014.11.008

Steiner, W., & Hruschka, H. (2003). Genetic Algorithms for Product Design: How Well do They Really Work? *International Journal of Market Research*, *45*(2), 1–13. https://doi.org/10.1177/147078530304500202

Storn, R., & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, (11), 341–359. https://doi.org/10.1023/A:1008202821328

Sugeno, M. (1985). *Industrial applications of fuzzy control*. North-Holland.

Tasgetiren, M. F., Chen, A., Gencyilmaz, G., & Gattoufi, S. (2009). Smallest Position Value Approach. In *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization* (pp. 121–138). https://doi.org/10.1007/978-3-540-92151-6_5

Toubia, O., Simester, D. I., Hauser, J. R., & Dahan, E. (2003). Fast Polyhedral Adaptive Conjoint Estimation. *Marketing Science*, *22*(3), 273–303. https://doi.org/10.1287/mksc.22.3.273.17743

Tsafarakis, S. (2016). Redesigning product lines in a period of economic crisis: a hybrid simulated annealing algorithm with crossover. *Annals of Operations Research*, *247*(2): 617–633. https://doi.org/10.1007/s10479-015-2032-0

Tsafarakis, S., Marinakis, Y., & Matsatsinis, N. (2011). Particle swarm optimization for optimal product line design. *International Journal of Research in Marketing*, *28*(1), 13–22. https://doi.org/10.1016/j.ijresmar.2010.05.002

Tsafarakis, S., & Matsatsinis, N. (2010). *Designing Optimal Products: Algorithms and Systems*. https://doi.org/10.1007/978-3-642-15606-9_19

Zhao, Z., Yang, J., Hu, Z., & Che, H. (2016). A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric Latin hypercube design for unconstrained optimization problems. *European Journal of Operational Research*, *250(1)*, 30−45.

Zhou, Y., Yi, W., Gao, L., & Li, X. (2016). Analysis of mutation vectors selection mechanism in differential evolution. *Applied Intelligence*, *44*(4), 904–912. https://doi.org/10.1007/s10489-015-0738-y

Zufryden, F. (1977). A conjoint measurement-based approach for optimal new product design and market segmentation. In A. D. Shocker (Ed.), *Analytical approaches to product and marketing planning*. Cambridge, MA: Marketing Science Institute.