Pakistan Journal of Statistics and Operation Research

Comparison of Metaheuristic Algorithms for Maximum Likelihood Estimation of the Transmuted Weibull Distribution with Applications

Shuaib Mursal Ibrahim[1*], Aydın Karakoca[2]

* Corresponding Author

1. Department of Statistics, Faculty of Economics, SIMAD University, Mogadishu, Somalia, shuaib.mursal@simad.edu.so
2. Department of Statistics, Faculty of Science, Necmettin Erbakan University, Konya, Türkiye, akarakoca@erbakan.edu.tr

Abstract

The Weibull distribution, widely utilized due to its flexibility, often requires generalization to improve its fit to real-world data. The Transmuted Weibull Distribution offers enhanced flexibility by incorporating a transmutation parameter. Metaheuristic algorithms have emerged as robust tools for parameter estimation, particularly for probability distributions with complex likelihood functions. This study compares the performance of four metaheuristic algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and Artificial Bee Colony (ABC) against the traditional Newton-Raphson (NR) algorithm for estimating parameters of the Transmuted Weibull Distribution (TWD). Extensive Monte Carlo simulations evaluated the algorithms' efficiencies using metrics like log-likelihood values, bias, mean squared error (MSE), and deficiency. Additionally, the methods are applied to real-world datasets to compare their practical utility. Both simulation and real data application results revealed that metaheuristic algorithms outperformed traditional Newton-Raphson (NR) optimization.

Key Words: Transmuted Weibull Distribution, Maximum Likelihood Estimation, Metaheuristic Algorithms, Nonlinear Optimization, Monte Carlo Simulation

1. Introduction

Probability distributions are fundamental in modeling phenomena across biology, engineering, economics, and environmental sciences. For instance, Weibull distribution is one of the widely used probability distribution in these fields. Proposed by Swedish physicist Weibull, it has attracted the attention of many researchers due its flexibility and wide applications to many fields. However, the standard Weibull distribution may not always provide an optimal fit to real-world data. Many generalizations and modified extensions of the Weibull distribution have been suggested in order to become more flexible and capable of modeling real world data. For example, Mudholkar et al. (1996) introduced a generalization of the Weibull distribution, Pal et al. (2006) proposed exponentiated Weibull distribution, Lai et al. (2003) presented modified Weibull distribution, and Lee et al. (2007) proposed Beta-Weibull distribution.

Aryal and Tsokos (2011) proposed the Transmuted Weibull Distribution (TWD), a generalization of the Weibull distribution based on the quadratic rank transmutation map (Shaw & Buckley, 2009). In their study, the authors added a transmutation parameter to two-parameter Weibull distribution. TWD is more flexible than its parent and other extensions of the Weibull distribution. For instance, Aryal et al. (2011) demonstrated the utility of TWD in real applications and compared it to the Weibull distribution and exponentiated Weibull distribution. Pobočíková et al.

(2018) applied TWD to lifetime data of engineering and biological sciences and compared TWD with two-parameter and three-parameter Weibull distributions.

Among the many parameter estimation methods in statistics, Maximum Likelihood Estimation (MLE) is widely favored due to its desirable mathematical properties and flexibility in modeling diverse probability distributions. MLE maximizes the likelihood function with respect to the unknown parameters. However, the solutions of the likelihood function of TWD cannot be obtained as an explicit function of the sample data as the likelihood is nonlinear function. Therefore, iterative algorithms such as Newton-Raphson, or iteratively reweighted least squares are needed to maximize the likelihood function. For example, Newton-Raphson (NR) algorithm is a traditional numerical algorithm used to solve the system of equations generated by partial derivatives of the likelihood function to find the estimated values of the parameter of interest for statistical distributions. However, its major drawback is that it uses a gradient-based search algorithm to find the best parameter values based on the inverse of the hessian matrix, making it only applicable to functions that can be differentiated at least twice. It is also important to note that traditional numerical algorithms are sensitive to the initial parameter values. Choosing the wrong starting point carries the risk of getting stuck in local optimum instead of finding the global optimum solution (Abbasi et al., 2006).

On the other hand, metaheuristic algorithms are inspired by natural phenomena such as the behavior of bee colonies or differential evolution. These algorithms work iteratively to find better solutions to the problem at hand. Metaheuristic algorithms are known for their ability to escape local optima and achieve global convergence, making them well-suited for complex and non-linear optimization problems such as parameter estimation for the TWD. This feature makes metaheuristic algorithms a useful tool for parameter estimation in TWD as an alternative to traditional numerical algorithms.

In this study genetic algorithm (GA), particle swarm optimization (PSO), differential evolution (DE) and artificial bee colony (ABC) metaheuristic algorithms will be used to maximize the likelihood function of TWD. Many studies have used metaheuristic algorithms to estimate some of the well-known probability distributions. For example, Kasap and Faouri (2024) used five metaheuristic algorithms to estimate the parameters of exponentially modified logistic distribution. Faouri and Kasap (2023) estimated the parameter of Nakagami distribution using PSO. Yalçınkaya et al. (2018) used GA to obtain the maximum likelihood estimates for the parameters of skew normal distribution. They compared the performance of GA with other traditional search techniques such as NR, Nelder Mead, and iteratively reweighted least squares Algorithm. They found that GA estimates were the most efficient compared to the traditional search techniques. Karakoca and Pekgör (2019) used a GA-based approach to estimate the maximum likelihood estimates of the parameters of progressively type-2 censored samples from the Weibull distribution. Based on extensive simulations, GA-based estimates outperformed the NR based estimates. Liu et al. (2011) highlighted GA's computational advantages in modeling wind speed distributions, while Wang et al. (2018) showed PSO and DE's strong performance alongside Grey Wolf Optimizer (GWO) in wind energy applications. Reddy and Singh (2014) used GA and PSO for drought risk modeling, while Wadi and Elmasry (2021) applied GA to fit Weibull, Poisson, and Lognormal distributions for wind power studies. Borges and Campos (2023) validated DE's utility in interval-censored data modeling. Jiang et al. (2015) assessed the wind speed frequency distribution using three probability distributions. Their study compared MLE, Method of Moments and three metaheuristic algorithms namely PSO, DE and Ant Colony Optimization (ACO). They found that parameter estimates based on the metaheuristic algorithms were more accurate than those of traditional numerical algorithms. Kılıç (2022) applied both ABC and GA to estimate parameters of the Weibull distribution, demonstrating their efficiency. Ravindra et al. (2012) modeled wind speed using a two-parameter Weibull distribution and showed that ABC provided better parameter estimates than the iterative MLE method. Xu et al. (2017) addressed the challenges of estimating q-Weibull distribution parameters, utilizing ABC and an adaptive hybrid ABC algorithm to handle the intricate nonlinear equations involved.

The objective of this study is to obtain the maximum likelihood estimates of the parameters of TWD by using four metaheuristic algorithms. We then compare the efficiencies of these estimators with corresponding estimates based on NR algorithm by conducting extensive Monte Carlo simulations. To the best of our knowledge, this is the first study to obtain the maximum likelihood estimates for the parameters of the TWD using metaheuristic algorithms.

The rest of the study is organized as follows: In Section 2, the TWD and its basic properties are presented. In section 3, the four metaheuristic algorithms and NR algorithm are introduced. In Section 4, the efficiencies of the parameter estimators are compared via a comprehensive Monte Carlo simulation. In section 5, two applications of real datasets are analyzed. In the last section, the study concludes.

2. Transmuted Weibull Distribution

Aryal and Tsokos (2011) introduced the transmuted Weibull distribution by generalizing the two-parameter Weibull distribution using the quadratic rank transmutation map (QRTM). The probability density function (pdf) and cumulative distribution function (cdf) of the two-parameter Weibull distribution are given by Equations (1) and (2) respectively.

$$f_1(x) = \frac{\eta}{\sigma}\left(\frac{x}{\sigma}\right)^{\eta-1} \exp\left(-\left(\frac{x}{\sigma}\right)^{\eta}\right), \qquad x > 0 \tag{1}$$

$$F_1(x) = 1 - \exp\left(-\left(\frac{x}{\sigma}\right)^{\eta}\right) \tag{2}$$

Using Equations (3) and (4), Aryal and Tsokos (2011) derived the cdf and pdf for the transmuted Weibull distribution, which are presented in Equations (5) and (6), respectively. In these equations, F_1 and f_1 denote the cdf and pdf of the parent distribution respectively, while F_2 and f_2 represent the cdf and pdf of the transmuted distribution respectively. The parameter λ in Equations (3), (4), (5) and (6), referred to as the transmutation parameter and is constrained to the interval $[-1,1]$.

$$F_2(x) = (1+\lambda)F_1(x) - \lambda F_1(x)^2 \tag{3}$$

$$f_2(x) = f_1(x)[1 + \lambda - 2\lambda F_1(x)] \tag{4}$$

$$F_2(x) = \left[1 - \exp\left(-\left(\frac{x}{\sigma}\right)\right)^{\eta}\right]\left[1 + \lambda \exp\left(-\left(\frac{x}{\sigma}\right)^{\eta}\right)\right] \tag{5}$$

$$f_2(x) = \frac{\eta}{\sigma}\left(\frac{x}{\sigma}\right)^{\eta-1} \exp\left(-\left(\frac{x}{\sigma}\right)^{\eta}\right)\left[1 - \lambda + 2\lambda \exp(-\left(\frac{x}{\sigma}\right)^{\eta}\right] \tag{6}$$

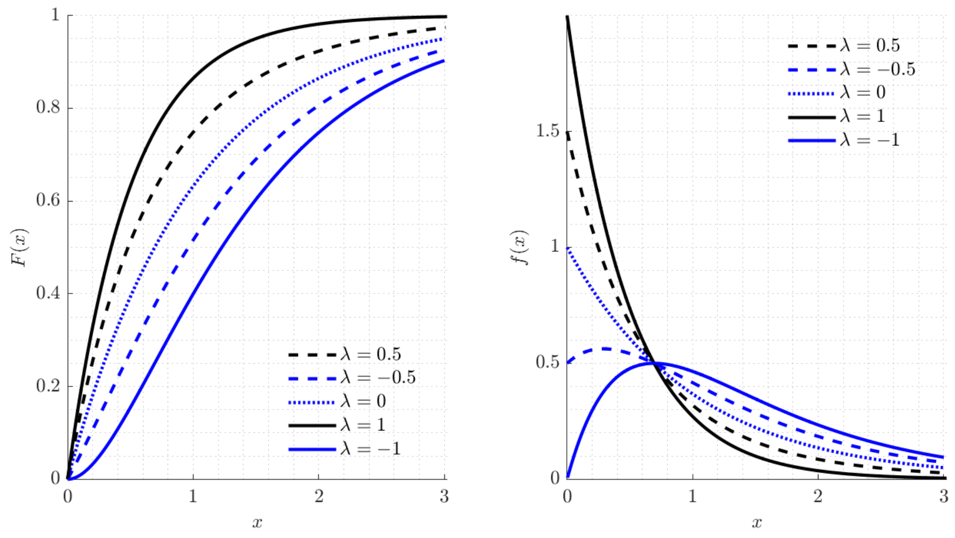

Figure 1. The left panel shows the cdf $F(x)$ and the right panel shows the pdf $f(x)$ of the TWD for different values of the transmutation parameter λ.

Aryal et al. noted that for $\eta = 1$, the distribution becomes a transmuted exponential distribution, for $\lambda = 0$, it reduces to the Weibull distribution, and when $\eta = \lambda = 1$ then resulting distribution is an exponential distribution with parameter $\frac{\sigma}{2}$. Possible shapes of the pdf and cdf of a transmuted Weibull distribution are visualized in Figure 1 for chosen values of λ, with η and σ held constant at 1.

The moments of the transmuted Weibull distribution are given by equation (7).

$$E(X^k) = \sigma^k \Gamma\left(1 + \frac{k}{\eta}\right)\left(1 - \lambda + \lambda 2^{\frac{k}{\eta}}\right) \tag{7}$$

The mean and variance can be obtained by using Equation (7) and is given in Equations (8) and (9) respectively.

$$E(X) = \sigma \Gamma\left(1 + \frac{1}{\eta}\right)\left(1 - \lambda + \lambda 2^{\frac{1}{\eta}}\right) \tag{8}$$

$$Var(X) = \sigma^2 \left\{\Gamma\left(1 + \frac{2}{\eta}\right)\left(1 - \lambda + \lambda 2^{\frac{2}{\eta}}\right) - \Gamma\left(1 + \frac{1}{\eta}\right)\left(1 - \lambda + \lambda 2^{\frac{1}{\eta}}\right)^2\right\} \tag{9}$$

3. Maximum Likelihood Estimation

The MLE method is used to estimate the parameters of the transmuted Weibull distribution. MLE maximizes the likelihood function by finding the optimal values of the unknown parameters of the Transmuted Weibull distribution. For computational simplicity, the log-likelihood is used. The log-likelihood of TWD is presented in Equation (10):

$$\begin{aligned} l &= n \ln \frac{\eta}{\sigma} - \sum_{i=1}^{n} \left(\frac{x_i}{\sigma}\right)^{\eta} + \sum_{i=1}^{n} \left(\frac{x_i}{\sigma}\right)^{\eta-1} + \sum_{i=1}^{n} \left[1 - \lambda + 2\lambda \exp\left(-\frac{x_i}{\sigma}\right)^{\eta}\right] \\ &= n \ln \eta - n\eta \ln \sigma + (\eta - 1) \sum_{i=1}^{n} \ln(x_i) - \sum_{i=n}^{n} \left(\frac{x_i}{\sigma}\right)^{\eta} + \sum_{i=1}^{n} \left[1 - \lambda + 2\lambda \exp\left(-\frac{x_i}{\sigma}\right)^{\eta}\right] \end{aligned} \tag{10}$$

The MLE for the unknown λ, η and σ is obtained by maximizing the log-likelihood function. If the partial derivatives of the log-likelihood are taken with respect to η, σ and λ, the resulting normal equations are provided in Equations (11), (12), and (13), respectively.

$$\frac{\partial l}{\partial \eta} = \frac{n}{\eta} + \sum_{i=1}^{n} \left[1 - \left(\frac{x_i}{\sigma}\right)^{\eta}\right] \ln\left(\frac{x_i}{\sigma}\right) - 2\lambda \sum_{i=1}^{n} \frac{\ln\left(\frac{x_i}{\sigma}\right)\left(\frac{x_i}{\sigma}\right)^{\eta} \exp\left(-\left(\frac{x_i}{\sigma}\right)^{\eta}\right)}{\left[1-\lambda+2\lambda \exp\left(-\left(\frac{x_i}{\sigma}\right)^{\eta}\right)\right]} = 0 \tag{11}$$

$$\frac{\partial l}{\partial \sigma} = -\frac{\eta}{\sigma} \sum_{i=1}^{n} \left[1 - \left(\frac{x_i}{\sigma}\right)^{\eta}\right] + \frac{2\lambda\eta}{\sigma} \sum_{i=1}^{n} \frac{\left(\frac{x_i}{\sigma}\right)^{\eta} \exp\left(-\left(\frac{x_i}{\sigma}\right)^{\eta}\right)}{\left[1-\lambda+2\lambda \exp\left(-\left(\frac{x_i}{\sigma}\right)^{\eta}\right)\right]} = 0 \tag{12}$$

$$\frac{\partial l}{\partial \lambda} = \sum_{i=1}^{n} \frac{2\exp(-\left(\frac{x_i}{\sigma}\right)^{\eta} - 1}{\left[1-\lambda+2\lambda \exp\left(-\left(\frac{x_i}{\sigma}\right)^{\eta}\right)\right]} = 0 \tag{13}$$

Since these equations are nonlinear, they can be solved iteratively. In this study, GA, PSO, DE, ABC and NR algorithms will be used to solve these equations.

3.1 Genetic Algorithm

GA proposed by Holland (1975), is a type of evolutionary algorithm that mimics the process of natural selection to find optimal or near-optimal solutions to complex problems. The first step is to decide how to represent potential solutions to the problem which is known as the chromosome representation. Chromosomes can be represented in many different ways, such as binary strings, real-valued vectors, or even graphs. Once a chromosome representation has been chosen, the next step is to initialize the population which involves creating a set of random chromosomes, each of which represents a potential solution to the problem. The objective function measures how well a candidate solution performs with respect to the problem, assigning a fitness value to each chromosome. In the Genetic Algorithm (GA), this fitness evaluation guides the search for the optimal solution. GA uses three genetic operators to evolve the population: selection, crossover, and mutation. Selection chooses chromosomes from the population to be parents of the next generation. There are different selection methods such as Random selection, Tournament selection and Roulette wheel selection. Chromosomes with higher fitness are more likely to be selected. Crossover combines two parent chromosomes to create two offspring chromosomes. This allows the GA to explore new regions of the search space. Mutation randomly changes one or more genes in a chromosome. This helps to prevent the GA from getting stuck in local optima.

3.2 Particle Swarm Optimization

PSO, proposed by Kennedy and Eberhart (1995), is a population-based metaheuristic algorithm which is inspired by the social behavior of the flocking of birds or schooling of fish in search of food. PSO, like other evolutionary algorithms such as GAs, aims to optimize a problem iteratively. However, unlike GA, PSO does not use genetic operators such as crossover and mutation. Instead, potential solutions, called particles, fly through the problem search space to find the optimal solution. Each particle in the swarm has a position and knows its personal best position and the fitness value associated with it. It also knows the global best position within the swarm, along with the current global fitness value.

Pseudo Code for GA Algorithm

Inputs: Objective function, lower bound (lb), upper bound (ub), Population number (nPop), number of offsprings (nC), number generations (Iteration), crossover probability (pC), Mutation rate (rM), mutation step size (sM).

```
1. Initialize a random population (nPop)
2. Evaluate the objective function
3. Memorize the best solution
4. Enter the evolution loop
   for it = 1: Iteration
       Perform random selection
       for k = 1: nC/2
           Select Parent 1
           Select Parent 2
           Perform Uniform Crossover
       end
       Perform mutation
       for i = 1: nC
           Perform mutation of the ith offspring
           Bound the offsprings
       end
       for i=1: nC
           Evaluate the new population
       end
       Merge and sort the populations
       Select the best nPop based on the objective function
       Update the best solution
   End
```

Each particle in the swarm is represented by a n-dimensional vector. Every particle in the swarm also has a velocity with the same dimension as its position. The velocity enables the particles to move toward the global best. The movement of particles is determined by the information exchange among the swarm particles, since each particle knows its personal best. Thus, their movement is based on communication and learning until they reach the global optimum of the problem. The velocity of each particle i in the swarm, at time (iteration) t, is updated according to Equation (14)

$$v_i(t+1) = wv_i(t) + c_1r_1(p_i(t) - x_i(t)) + c_2r_2[g(t) - x_i(t)] \tag{14}$$

where w is the inertia weight, used to balance global exploration and local exploitation. c_1 and c_2 are acceleration coefficients for the cognitive and social components, respectively. r_1 and r_2 are uniformly distributed n −dimensional random vectors between [0, 1]. $p_i(t)$ is the personal best position of particle i at time t. $g(t)$ is the current global best position of the swarm. The position of each particle at time t is updated according to Equation (15).

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{15}$$

The initial vectors of x_0 and v_0 can be generated using a uniform distribution. Likewise, the personal best position should be initialized by the current position of particle i that is $p_0 = x_0$.

Pseudo Code for PSO Algorithm

```
Inputs: Objective function, lower bound (lb), upper bound (ub), swarm size (nPop), inertia weight (w), cognitive
coefficient (c1), social coefficient (c2), number of cycles (T).
    1. Initialize a random population
    2. Evaluate the objective function
    3. Memorize the personal best (obj_pbest) and global best (obj_gbest) solutions
    4. Enter the cycle loop
       for it = 1: T
       for i = 1: nPop
               Update velocity
               Update position
               Bound the position
               Calculate the objective function
               Perform greedy selection
               if objective(i) > obj_pbest(i)
                       Update Personal best
                       if obj_pbest(i) > obj_gbest
                               Update Global best
                       end
               end
       end
       end
```

3.3 Differential Evolution

DE algorithm, proposed by Storn and Price (1997), is a population-based metaheuristic algorithm inspired by Darwin's theory of evolution. DE operates in generations, similar to biological evolutions. Each generation consists of a population of individuals represented by vectors where each vector can be thought as chromosome. Each individual in the population is a potential solution to the problem at hand. DE has three operators and applies these three operators to find the optimal solution given the parameter space. These operators are mutation, crossover, and selection.

The first operator of DE to be performed after the initial population is generated is mutation. In DE there are three different vectors namely Target (parent) vector, Donor (mutant) vector, and Trail (offspring) vector. Mutation is applied to the Target vector by selecting three distinct and randomly chosen individuals from the population. These three individuals should be different from the current individual. After every individual has undergone a mutation, a donor vector is created. Mutation is performed using Equation (16)

$$V = X_{r_1} + F\left(X_{r_2} - X_{r_3}\right) \tag{16}$$

$r1, r2,$ and $r3$ are indices of the randomly selected individuals (chromosomes) where $r1, r2,$ and $r3 \in (1, 2, 3, \dots, NP)$ and NP stands for number of populations.

The second operator of DE is recombination (crossover) operator. To increase the diversity of the population, a crossover is performed by exchanging the donor vector and the target vector. The crossover operator used in DE algorithm can be either exponential or uniform crossover. In uniform crossover, a crossover probability is predetermined first. A random number between 0 and 1 is generated for each decision variable in this case for each gene. Likewise, a random number between 1 and the length of the decision variables is generated. The crossover probability is compared to the generated random number. If the random number is less than or equal to the crossover probability or the generated number is equal to the current gene or decision variable, then this decision variable or gene comes from the donor vector, otherwise, it comes from the target vector. The mathematical expression for the recombination is given by Equation (17).

$$u_j = \begin{cases} v_j & if\ r \le P_c \quad or \quad j = j_r \\ x_j & if\ r > P_c \quad and \quad j \ne j_r \end{cases} \tag{17}$$

The parameter P_c denotes the crossover probability, v_j represents the j^{th} variable derived from the donor vector, and x_j corresponds to the j^{th} variable extracted from the target vector. Additionally, j_r is a randomly generated number, where $j_r \in (1,2,3,\dots,D)$ and D denotes the length of the parameters to be estimated. After performing crossover operator to each chromosome, the resulting vector is called trail vector.

The third DE operator is the selection operator, which keeps the population size fixed by performing greedy selection between the target vector and the trial vector to determine which one survives to the next generation. This greedy selection is done according to the fitness function. The greedy selection is performed using equation (18) where $x_{i,g+1}$ represents the solution that passes to the next generation using Equation (18):

$$x_{i,g+1} = \begin{cases} u_i, & f(u_i) \ge f(x_i) \\ x_i, & f(u_i) < f(x_i) \end{cases} \tag{18}$$

3.4 Artificial Bee Colony

ABC, developed by Karaboga and Basturk (2007), is a nature-inspired population-based metaheuristic algorithm which was inspired by the foraging behavior of honeybees. The ABC algorithm has three important components. The first component is food sources where a bee explores the neighborhood around the hive in search of food. A bee selects a food source, it evaluates several properties of the food such as its proximity to the hive, nectar amount, and ease of exploiting the food source. In the ABC algorithm, the food source represents a potential solution to the optimization problem and the nectar amount of the food source represents the quality of the food source which corresponds to the fitness function. The second component of the ABC algorithm is employed forager. An employed forager is a bee which has already visited a food source and exploiting it. It memorizes the location of the food source it has visited and returns to the hive carrying information about the food quality such as distance to the hive, direction and profitability of the food source. The third component is unemployed forager. An unemployed forager is a bee which is currently looking for food source to exploit. An unemployed forager can be an onlooker or a scout bee. An onlooker bee watches the employed bee in the dancing area where employed bees perform waggle dance to recruit other bees. This dance is a way to communicate and exchange information by the bees. Onlooker bees select a promising food source based on the information shared by the employed bees. The scout bees explore new food sources when the existing ones are exhausted.

In the ABC algorithm, the number of food sources is equal to the number of employed bees or onlooker bees. Each food source is exploited by one bee. In the first step of the algorithm is to randomly generate food source for each bee in D-dimensional space where D represents the number of parameters in the optimization problem. The objective function and the fitness value of each food source is evaluated and stored. The fitness value can be calculated Equation (19).

$$\begin{cases} \dfrac{1}{1+f_i}, & f_i \ge 0 \\ \dfrac{1}{1+|f_i|}, & f_i < 0 \end{cases} \tag{19}$$

where f_i is the objective function value of the i^{th} food source.

Pseudo Code for DE Algorithm

```
Inputs: Objective function, lower bound (lb), upper bound (up), Number of parameters (D), Population size (nPop),
Number of Iterations (T), Crossover Probability (pC), Scaling Factor (F)
    1. Initialize a random population (nPop)
    2. Calculate the objective function (f)
    3. Memorize the best solution
       Enter the evolution loop
       for t = 1: T
       for i = 1: nPop
               Perform mutation and obtain the donor vector
               Perform crossover and generate Trail vector
       end
       for i = 1: nPop
               Bound the Trail Vector
               Evaluate the fitness function of the trail vector (fu)
               Perform greedy selection between f and fu
               update nPop
       end
       Update best solution
       end
```

The employed bees explore new food source (solution) around the current food source (solution) using Equation (20).

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{20}$$

where ϕ_{ij} is random number between -1 and 1, $k \in (1,2,3,\ldots,N)$ and $i \neq k$.

The fitness value of the new solution is calculated. The current solution is compared with the new solution. A greedy selection is performed and the best of the two solutions is stored, and the other one is discarded. After all the employed bees complete their search, they share the fitness value of each food source to the onlooker bees. An onlooker bee selects a food source based on the probability value with that food source. The probability is calculated using Equation (21).

$$p_i = \left(0.9 \times \frac{f_i}{\max(f)}\right) + 0.1 \tag{21}$$

After selecting a food source, a new solution is generated around the selected food source using Equation (20).

The fitness value of the new food source is calculated and compared to the current one. If the new food source is better than the current one, the new food source is stored otherwise the current one is kept. If the quality of the food source is low, that food source is abandoned. A new food source is explored by a scout bee. In ABC algorithm, a food source is abandoned if the profitability of that food source cannot be improved in predefined number of trails.

Pseudo Code for ABC

Inputs: Objective Function, lower bound (lb), upper bound (up), Number of parameters (D), Population size (nPop), Number of Iterations (T), limit

```
1. Initialize a random population (nPop)
2. Calculate the objective function (f) and fitness function (fit)
3. Memorize the best solution
4. Set the trail counter to 0
   Enter the main loop
   for it = 1: T
   for i = 1: nPop
       Perform Employed Bee Phase
       Perform Greedy selection
       if fitnew < fit(i)
           Update the nPop(i)
           Update the f(i)
           Update the fit(i)
           Set trail(i) to zero
       else
           increment trail(i) by 1
       end
   end
   Determine the of Probability of each food source
   for i = 1: nPop
       Perform Onlooker Bee Phase
       Generate new solution
       Perform Greedy selection
       if fitnew < fit(i)
           Update the nPop(i)
           Update the f(i)
           Update the fit(i)
           Set trail(i) to zero
       else
           Increment trail(i) by 1
       End
   end
   Memorize the best food source
   Perform Scout Bee Phase
   if trail(i) > limit
       set trail(i) to zero
       Generate new food source
       Calculate the f and fit functions
   end
   Update the best food source
   end
```

3.5 Newton Raphson Algorithm

The Newton-Raphson optimization algorithm is an iterative technique used to find the critical points (local minima or maxima) for a real-valued function. The general steps of the Newton-Raphson method for multivariate optimization are as follows:

1. Initialize: Start with an initial guess for the optimal solution, denoted as

$$x^{(0)} = \left[x_1^{(0)}, x_2^{(0)}, \cdots, x_p^{(0)}\right]$$

   where p is the number of parameters.

2. Compute Gradient and Hessian: Calculate the gradient vector, $\nabla f\left(x^{(k)}\right)$, and the Hessian matrix, $\nabla^2 f\left(x^{(k)}\right)$, of the objective function $f(x)$ at the current iteration $x^{(k)}$, $k = 0,1,2,\cdots$. Here k represents the number of iterations.

$$\nabla f\left(x^{(k)}\right) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \cdots, \frac{\partial f}{\partial x_p}\right]^T$$

$$\nabla^2 f\left(x^{(k)}\right) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_p} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_p \partial x_1} & \frac{\partial^2 f}{\partial x_p \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_p^2} \end{bmatrix}$$

3. Update: Compute the update direction $d^{(k)}$ using Equation (22).

$$d^{(k)} = -\left[\nabla^2 f\left(x^{(k)}\right)\right]^{-1} \nabla f\left(x^{(k)}\right) \tag{22}$$

4. Iterate: Update the current iterate $x^{(k)}$ using Equation (23).

$$x^{(k+1)} = x^{(k)} + d^{(k)} \tag{23}$$

Repeat steps 2-4 until the convergence criteria are met, such as the magnitude of the gradient vector or the change in the function value being smaller than a specified tolerance.

4. Simulation Results

This section presents a comprehensive simulation study that aims to evaluate the performance of different parameter estimation algorithms for TWD. The transmutation parameter (λ) will be systematically varied from -1, -0.5, 0.5 to 1 to examine the behavior of the algorithms under different scenarios.

To examine the impact of sample size on parameter estimation accuracy, three sample sizes (50, 100, and 200) will be considered within each scenario. All other distribution parameters will remain constant throughout the simulation. The parameters for each algorithm were carefully selected to ensure effective optimization. For the GA, a population size (nPop) of 100 was chosen, along with 100 generations (T). Additionally, the crossover probability (pC), mutation rate (rM) and mutation step size (sM) were set to 0.8, 0.02 and 0.1, respectively. Similarly, the PSO algorithm utilized a swarm size (nPop) of 100, an inertia weight (w) of 0.8, and cognitive (c1) and social (c2) coefficients of 1.5, with 100 cycles (T). Furthermore, the DE algorithm adopted a population size of 100 and 100 generations (T), a crossover probability (pC) of 0.8 and a scaling factor (F) of 0.85. Finally, for the ABC algorithm, the swarm size was also set to 100, but it featured 50 food sources (F), a limit of 5, and a 100 number of cycles (T).

These parameters were chosen to balance computational efficiency and optimization performance. The performance of each algorithm will be assessed using bias, mean squared error (MSE) and Def as defined in Equations (24), (25) and (26) respectively. The algorithm with the lowest bias, MSE and Def will be considered as the best-performing algorithm. All simulations will be conducted using MATLAB R2022b. Each simulation will be repeated 2000 times.

$$\text{bias}\left(\hat{\beta}\right) = E\left(\hat{\beta}\right) - \beta \tag{24}$$

$$MSE\left(\hat{\beta}\right) = E\left[\left(\hat{\beta} - \beta\right)^2\right] = Var\left(\hat{\beta}\right) + \left[bias\left(\hat{\beta}\right)\right]^2 \tag{25}$$

$$Def = MSE(\hat{\eta}) + MSE(\hat{\sigma}) + MSE(\hat{\lambda}) \tag{26}$$

The results of the Monte Carlo simulations are presented in Tables 1-4. The results indicate that DE performs best in minimizing bias for the η and σ parameters when λ is negative (-0.5 and -1). NR, however, outperforms in estimating the λ parameter when $\lambda = -1$. For positive λ values, GA excels in minimizing bias for the λ parameter across all sample sizes and for the η parameter at small sample sizes ($n = 50$). NR shows superior performance for the η parameter at moderate to large sample sizes ($n = 100, 200$) and for the σ parameter at small to moderate sample sizes ($n = 50, 100$). At larger sample sizes ($n = 200$), PSO achieves the lowest bias for the σ parameter and consistently performs well for both η and λ parameters when $\lambda = 1$.

DE achieves the lowest MSE for η and σ parameters when λ is negative ($-0.5, -1$), highlighting its stability under such conditions. For the λ parameter, NR produces the smallest MSE, followed closely by DE. For positive λ values (0.5 and 1), GA demonstrates superior performance, achieving the lowest MSE for all parameters except when $\lambda = 1$, where NR has the smallest MSE for the σ parameter. Notably, GA struggles with negative λ values, showing the highest MSE across all sample sizes. Conversely, DE exhibits instability under positive λ values, particularly for η and σ parameters, while NR performs poorly in these scenarios due to higher bias and MSE values.

Based on the deficiency (Def) criterion, DE is the most efficient algorithm for estimating TWD parameters when the true λ value is negative, achieving the lowest Def values across all sample sizes. For positive λ values, GA emerges as the most efficient method, consistently outperforming other algorithms. These findings highlight the adaptability of DE and GA under different parameter conditions. Notably, NR is less reliable, as it frequently yields negative λ estimates regardless of the true λ value.

Table1: Simulated parameter values for transmuted Weibull distribution ($\eta = 2, \sigma = 2, \lambda = -1$)

n	Algorithm	η				σ				λ				Def
		Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	
50	**GA**	3.0870	**0.1342**	1.0870	1.3157	2.7963	0.0618	0.7963	0.6960	0.4023	0.1128	1.4023	2.0791	4.0909
	PSO	2.9104	0.2419	0.9104	1.0707	2.6977	0.1490	0.6977	0.6358	0.2123	0.3902	1.2123	1.8597	3.5663
	DE	2.4704	0.3071	**0.4704**	**0.5283**	2.2846	0.1410	**0.2846**	**0.2220**	-0.5106	0.3909	0.4894	0.6304	**1.3808**
	ABC	2.4817	0.3125	0.4817	0.5446	2.3002	0.1512	0.3002	0.2414	-0.4841	0.4216	0.5159	0.6878	1.4738
	NR	3.1036	0.1464	1.1036	1.3643	2.7626	**0.0422**	0.7626	0.6237	-0.9995	**0.0000**	**0.0005**	**0.0000**	1.9880
100	**GA**	3.0521	**0.0627**	1.0521	1.1695	2.8506	0.0412	0.8506	0.7648	0.4858	0.0854	1.4858	2.2930	4.2274
	PSO	2.8547	0.2002	0.8547	0.9307	2.7186	0.1539	0.7186	0.6703	0.2423	0.4247	1.2423	1.9681	3.5692
	DE	2.3898	0.2480	**0.3898**	**0.3999**	2.2756	0.1508	**0.2756**	**0.2268**	-0.5271	0.4350	0.4729	0.6586	**1.2853**
	ABC	2.4228	0.2606	0.4228	0.4394	2.3140	0.1735	0.3140	0.2721	-0.4620	0.4958	0.5380	0.7853	1.4968
	NR	3.0469	0.0691	1.0469	1.1651	2.7596	**0.0182**	0.7596	0.5953	-0.9992	**0.0001**	**0.0008**	**0.0001**	1.7605
200	**GA**	3.0328	**0.0364**	1.0328	1.1032	2.8890	0.0301	0.8890	0.8205	0.5483	0.0677	1.5483	2.4650	4.3887
	PSO	2.8334	0.1825	0.8334	0.8771	2.7343	0.1530	0.7343	0.6922	0.2707	0.4340	1.2707	2.0487	3.6180
	DE	2.3177	0.2222	**0.3177**	**0.3232**	2.2482	0.1548	**0.2482**	**0.2164**	-0.5688	0.4556	0.4312	0.6415	**1.1810**
	ABC	2.3564	0.2375	0.3564	0.3646	2.2898	0.1767	0.2898	0.2606	-0.4984	0.5198	0.5016	0.7714	1.3966
	NR	3.0110	0.0432	1.0110	1.0653	2.7665	**0.0234**	0.7665	0.6109	-0.9992	**0.0001**	**0.0008**	**0.0001**	1.6763

Table 2: Simulated parameter values for transmuted Weibull distribution ($\eta = 2, \sigma = 2, \lambda = -0.5$)

n	Algorithm	η				σ				λ				Def
		Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	
50	**GA**	2.3965	**0.1022**	0.3965	0.2593	2.3892	**0.0861**	0.3892	0.2376	0.1394	0.1560	0.6394	0.5648	1.0618
	PSO	2.3145	0.1276	0.3145	0.2265	2.4091	0.1764	0.4091	0.3438	0.1353	0.3362	0.6353	0.7398	1.3101
	DE	2.1411	0.1487	**0.1411**	**0.1686**	2.1234	0.1180	**0.1234**	**0.1332**	-0.2967	0.2629	**0.2033**	0.3042	**0.6060**
	ABC	2.1686	0.1472	0.1686	0.1756	2.1686	0.1366	0.1686	0.1650	-0.2288	0.2989	0.2712	0.3725	0.7130
	NR	2.4769	0.1166	0.4769	0.3441	2.5259	0.1151	0.5259	0.3917	-0.9990	**0.0001**	-0.4990	**0.2490**	0.9848
100	**GA**	2.3711	**0.0530**	0.3711	0.1907	2.4167	0.0787	0.4167	0.2523	0.1775	0.1556	0.6775	0.6146	1.0576
	PSO	2.2966	0.0793	0.2966	0.1672	2.4647	0.1758	0.4647	0.3918	0.2112	0.3398	0.7112	0.8456	1.4046
	DE	2.1434	0.1039	**0.1434**	**0.1244**	2.1717	0.1228	**0.1717**	**0.1523**	-0.2226	0.2809	**0.2774**	0.3578	**0.6345**
	ABC	2.1779	0.1025	0.1779	0.1341	2.2264	0.1396	0.2264	0.1908	-0.1384	0.3107	0.3616	0.4415	0.7664
	NR	2.4510	0.0588	0.4510	0.2621	2.5144	**0.0553**	0.5144	0.3199	-0.9986	**0.0002**	-0.4986	**0.2488**	0.8307
200	**GA**	2.3506	**0.0320**	0.3506	0.1550	2.4132	0.0731	0.4132	0.2438	0.1674	0.1527	0.6674	0.5980	0.9968
	PSO	2.2656	0.0582	0.2656	0.1287	2.4722	0.1872	0.4722	0.4101	0.2079	0.3681	0.7079	0.8692	1.4080
	DE	2.1457	0.0759	**0.1457**	**0.0972**	2.1976	0.1215	**0.1976**	**0.1606**	-0.1860	0.2780	**0.3140**	0.3766	**0.6343**
	ABC	2.1877	0.0696	0.1877	0.1048	2.2696	0.1384	0.2696	0.2111	-0.0782	0.3084	0.4218	0.4863	0.8022
	NR	2.4333	0.0379	0.4333	0.2256	2.5214	**0.0627**	0.5214	0.3346	-0.9992	**0.0001**	-0.4992	**0.2493**	0.8095

Table 3: Simulated parameter values for transmuted Weibull distribution ($\eta = 2, \sigma = 2, \lambda = 0.5$)

n	Algorithm	η				σ				λ				Def
		Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	
50	**GA**	2.0090	**0.0591**	**0.0090**	**0.0592**	1.8858	**0.0657**	-0.1142	**0.0787**	0.3013	0.1253	**-0.1987**	**0.1648**	**0.3027**
	PSO	1.9277	0.0876	-0.0723	0.0928	1.8863	0.1483	-0.1137	0.1612	0.2618	0.3328	-0.2382	0.3896	0.6436
	DE	1.7468	0.1184	-0.2532	0.1825	1.5754	0.1332	-0.4246	0.3135	-0.2674	0.3708	-0.7674	0.9597	1.4556
	ABC	1.7907	0.1149	-0.2093	0.1587	1.6467	0.1491	-0.3533	0.2739	-0.1455	0.4076	-0.6455	0.8242	1.2569
	NR	2.0321	0.0690	0.0321	0.0700	1.9121	0.0795	**-0.0879**	0.0872	-0.9990	**0.0001**	-1.4990	2.2470	2.4041
100	**GA**	1.9805	**0.0291**	-0.0195	**0.0294**	1.9186	**0.0542**	-0.0814	**0.0609**	0.3483	0.1130	**-0.1517**	**0.1360**	**0.2263**
	PSO	1.8949	0.0599	-0.1051	0.0710	1.9228	0.1478	-0.0772	0.1538	0.3050	0.3390	-0.1950	0.3770	0.6017
	DE	1.7384	0.0904	-0.2616	0.1588	1.6230	0.1431	-0.3770	0.2853	-0.1885	0.4029	-0.6885	0.8770	1.3210
	ABC	1.7898	0.0863	-0.2102	0.1305	1.7122	0.1576	-0.2878	0.2404	-0.0378	0.4296	-0.5378	0.7188	1.0897
	NR	1.9905	0.0374	**-0.0095**	0.0375	1.9264	0.0853	**-0.0736**	0.0908	-0.9991	**0.0001**	-1.4991	2.2474	2.3756
200	**GA**	1.9688	**0.0170**	-0.0312	**0.0180**	1.9403	**0.0477**	-0.0597	**0.0513**	0.3817	0.1074	**-0.1183**	**0.1214**	**0.1908**
	PSO	1.8921	0.0424	-0.1079	0.0540	1.9525	0.1317	**-0.0475**	0.1340	0.3523	0.3128	-0.1477	0.3346	0.5226
	DE	1.7622	0.0787	-0.2378	0.1353	1.6831	0.1448	-0.3169	0.2452	-0.0832	0.4160	-0.5832	0.7561	1.1367
	ABC	1.8277	0.0667	-0.1723	0.0964	1.7989	0.1468	-0.2011	0.1872	0.1104	0.3993	-0.3896	0.5511	0.8347
	NR	1.9836	0.0210	**-0.0164**	0.0213	1.9152	0.0483	-0.0848	0.0555	-0.9986	**0.0002**	-1.4986	2.2460	2.3228

Table 4: Simulated parameter values for transmuted Weibull distribution ($\eta = 2, \sigma = 2, \lambda = 1$)

n	Algorithm	η				σ				λ				Def
		Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	Mean	Variance	Bias	MSE	
50	**GA**	2.1115	**0.0640**	0.1115	**0.0764**	1.5059	**0.0371**	-0.4941	0.2813	0.1939	0.1229	-0.8061	**0.7728**	**1.1305**
	PSO	2.0363	0.0817	**0.0363**	0.0830	1.5484	0.0917	-0.4516	0.2956	0.2341	0.3239	**-0.7659**	0.9105	1.2891
	DE	1.8828	0.1103	-0.1172	0.1240	1.3196	0.0651	-0.6804	0.5280	-0.2451	0.2876	-1.2451	1.8378	2.4898
	ABC	1.9141	0.1053	-0.0859	0.1127	1.3698	0.0795	-0.6302	0.4766	-0.1420	0.3410	-1.1420	1.6450	2.2343
	NR	2.1564	0.0748	0.1564	0.0992	1.5749	0.0459	**-0.4251**	**0.2266**	-0.9986	**0.0002**	-1.9986	3.9947	4.3205
100	**GA**	2.0828	**0.0346**	0.0828	**0.0415**	1.5285	0.0327	-0.4715	0.2550	0.2343	0.1218	-0.7657	**0.7081**	**1.0045**
	PSO	2.0076	0.0542	**0.0076**	0.0543	1.5675	0.0874	-0.4325	0.2745	0.2649	0.3190	**-0.7351**	0.8593	1.1881
	DE	1.8817	0.0814	-0.1183	0.0954	1.3572	0.0663	-0.6428	0.4795	-0.1646	0.3028	-1.1646	1.6592	2.2341
	ABC	1.9294	0.0741	-0.0706	0.0791	1.4239	0.0763	-0.5761	0.4082	-0.0243	0.3315	-1.0243	1.3807	1.8680
	NR	2.1281	0.0404	0.1281	0.0568	1.5729	**0.0284**	**-0.4271**	**0.2107**	-0.9990	**0.0002**	-1.9990	3.9961	4.2637
200	**GA**	2.0661	**0.0186**	0.0661	**0.0229**	1.5320	**0.0315**	-0.4680	0.2505	0.2410	0.1166	-0.7590	**0.6927**	**0.9661**
	PSO	1.9988	0.0331	**-0.0012**	0.0331	1.5954	0.0860	**-0.4046**	0.2498	0.3162	0.3060	**-0.6838**	0.7736	1.0565
	DE	1.9156	0.0550	-0.0844	0.0622	1.3998	0.0594	-0.6002	0.4196	-0.0650	0.2666	-1.0650	1.4008	1.8825
	ABC	1.9580	0.0470	-0.0420	0.0487	1.4817	0.0755	-0.5183	0.3441	0.0989	0.3044	-0.9011	1.1163	1.5091
	NR	2.1050	0.0278	0.1050	0.0388	1.5811	0.0359	-0.4189	**0.2114**	-0.9983	**0.0002**	-1.9983	3.9934	4.2437

5. Real Data Applications

We consider two datasets to test the utility of metaheuristic algorithms in obtaining the maximum likelihood estimates for TWD parameters. The performance of the parameter estimation algorithms is compared using the log-likelihood values and Kolmogorov-Smirnov (KS) test and Akaike Information Criterion (AIC). KS is a non-parametric test which assesses the goodness-of-fit between the empirical distribution function $\hat{F}(x)$ of the data and the theoretical distribution function $F(x)$ of the distribution being evaluated. The KS statistic measures the maximum absolute difference between these two cumulative distributions. The null hypothesis (H_0) assumes that the data originates from the proposed distribution. Rejection of the null hypothesis at a significance level of α = 0.05 indicates a statistically significant difference between the data and the theoretical distribution. The K-S test and AIC equations are presented in Equations (27) and (28) respectively.

$$KS = \max |\hat{F}(x) - F(x)| \tag{27}$$

$$AIC = -2\log(l) + 2p \tag{28}$$

$log(L)$ represents the maximized log-likelihood of the model, n is the sample size, and p is the number of parameters in the model. The algorithm that produces the highest log-likelihood value or the lowest AIC value is considered the best algorithm.

Dataset 1

The first data set represents the lifetimes of Kevlar 49/epoxy strands subjected to constant sustained pressure at 90% stress level until the strand failure. This data is extracted from Barlow et al. (1984) and recently used by (Owoloko et al., 2015). The data are as follows:

0.0251	0.0886	0.0891	0.2501	0.3113	0.3451	0.4763	0.5650	0.5671
0.6566	0.6748	0.6751	0.6753	0.7696	0.8375	0.8391	0.8425	0.8645
0.8851	0.9113	0.9120	0.9836	1.0483	1.0596	1.0773	1.1733	1.2570
1.2766	1.2985	1.3211	1.3503	1.3551	1.4595	1.4880	1.5728	1.5733
1.7083	1.7263	1.7460	1.7630	1.7746	1.8275	1.8375	1.8503	1.8808
1.8878	1.8881	1.9316	1.9558	2.0048	2.0408	2.0903	2.1093	2.1330
2.2100	2.2460	2.2878	2.3203	2.3470	2.3513	2.4951	2.5260	2.9911
3.0256	3.2678	3.4045	3.4846	3.7433	3.7455	3.9143	4.8073	5.4005
5.4435	5.5295	6.5541	9.0960					

Table 5: The descriptive statistics for the Kevlar 49/epoxy data

n	Min	Max	Mean	Median	Variance	Skewness	Kurtosis
76	0.0251	9.096	1.959	1.736	2.477	2.0196	5.6

Table 5 provides the descriptive statistics for the Kevlar dataset, which consists of the lifetimes of Kevlar 49/epoxy strands under constant pressure. As shown in Table 6, the metaheuristic algorithms (GA, PSO, DE, ABC) yielded comparable maximum likelihood estimates for the TWD parameters, all achieving similar log-likelihood values and AIC scores. In contrast, the NR algorithm performed poorly, with significantly higher AIC values and lower log-likelihoods. The KS test confirmed that the parameter estimates from NR failed to adequately fit the data (p-value = 0.004), while the metaheuristic methods produced better fits (p-values > 0.45). Figure 2 illustrates the histogram and the fitted TWD PDFs. While the NR-based estimates deviate noticeably from the data distribution, all metaheuristic algorithms converged to identical parameter estimates (and thus identical PDF curves), consistently capturing the underlying distribution.

Table 6: The parameter estimates, p-value of K-S, Log-likelihood and AIC values for Kevlar 49/epoxy data

Algorithm	η	σ	λ	Log-likelihood	AIC	P-value
GA	1.431	2.941	0.711	-121.735	249.4706	0.460
PSO	1.431	2.942	0.711	-121.735	249.4706	0.458
DE	1.431	2.942	0.711	-121.735	249.4706	0.458
ABC	1.431	2.944	0.713	-121.735	249.4706	0.460
NR	1.492	3.976	0.954	-124.767	255.5342	0.004

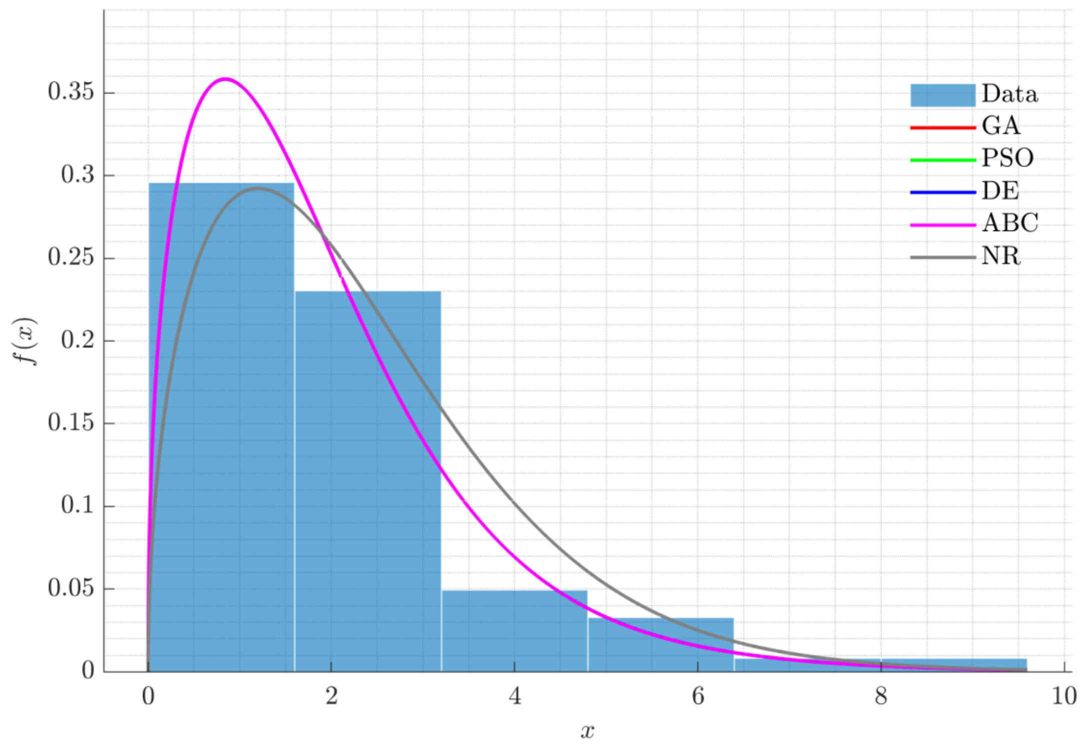

Figure 2. Histogram and fitted pdf of the TWD for Kevlar dataset.

Dataset 2

The second data set represents survival times (in days) of 72 guinea pigs infected with virulent tubercle bacilli. The data in reported in Bjerkedal (1960) and recently used by (Owoloko et al., 2016; Pobočíková et al., 2018). The data is as follows:

10	93	108	122	153	183	230	293
33	96	108	122	159	195	231	327
44	100	109	124	160	196	240	342
56	100	112	130	163	197	245	347
59	102	113	134	163	202	251	361
72	105	115	136	168	213	253	402
74	107	116	139	171	215	254	432
77	107	120	144	172	216	254	458
92	108	121	146	176	222	278	555

Table 7 summarizes the descriptive statistics for the guinea pig survival dataset, which includes survival times (in days) for 72 subjects infected with virulent tubercle bacilli. The estimated TWD parameters are shown in Table 8. While all algorithms, including GA, PSO, DE, ABC, and NR, provided similar results, the metaheuristic algorithms achieved marginally lower AIC values and higher log-likelihood scores compared to NR. This indicates a slight advantage in model parsimony and fit quality. However, the KS test p-values (ranging from 0.451 to 0.597) suggest that all methods fit the dataset well. Figure 3 illustrates the histogram and the fitted TWD PDFs, showing that all algorithms effectively modeled the survival times without significant discrepancies.

Table 7. Descriptive statistics for the guinea pigs survival times data

n	Min	Max	Mean	Median	Variance	Skewness	Kurtosis
72	10	555	176.894	149.5	10702.91	1.371	2.225

Table 8. The parameter estimates, p-value of K-S, Log-likelihood and AIC values for guinea pigs survival times data

Algorithm	η	σ	λ	Log-likelihood	AIC	P-value
GA	1.322	138.561	-0.937	-425.746	857.492	0.451
PSO	1.315	137.184	-0.969	-425.720	857.439	0.493
DE	1.353	139.624	-0.951	-425.684	857.369	0.586
ABC	1.352	139.487	-0.953	-425.684	857.369	0.586
NR	1.412	145.460	-0.895	-425.788	857.576	0.597

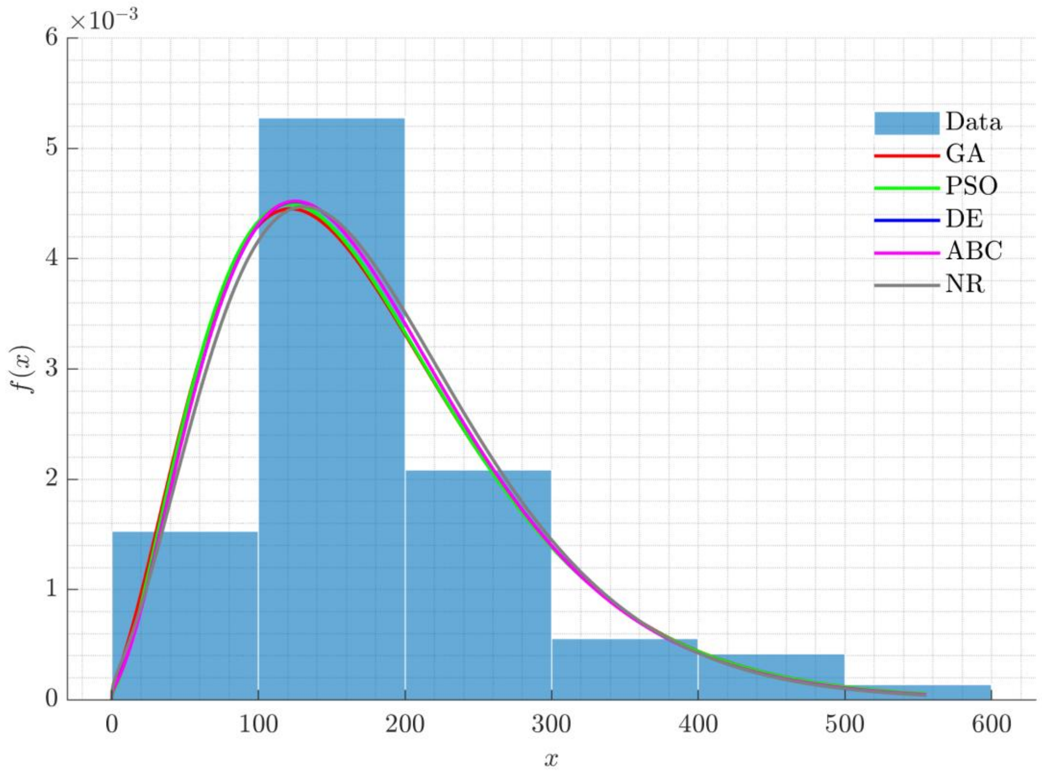

Figure 3. Histogram and fitted pdf of the TWD for guinea pigs survival times.

6. Conclusion

This study investigated the application of four metaheuristic algorithms including Genetic Algorithm, Particle Swarm Optimization, Differential Evolution, and Artificial Bee Colony for maximum likelihood estimation of the Transmuted Weibull Distribution. Through extensive Monte Carlo simulations and real-world applications, the performance of these algorithms was compared to the traditional Newton-Raphson optimization method.

The results demonstrated that metaheuristic algorithms offer significant advantages over NR in terms of accuracy, particularly in handling the nonlinear likelihood equations of TWD. Among the metaheuristics, DE and GA consistently provided superior parameter estimates under varying conditions, achieving lower bias, mean squared error, and deficiency values in the simulation study. The algorithms' ability to escape local optima and converge globally makes them reliable alternatives to traditional numerical methods such as NR algorithm. Specifically, the study suggested using DE for scenarios with negative λ and GA for positive λ, positioning these algorithms as superior alternatives to traditional approaches.

Applications to Kevlar 49/epoxy and guinea pig survival datasets further validated the effectiveness of metaheuristics, with improved log-likelihood, AIC, and Kolmogorov–Smirnov test results compared to NR. These findings underscore not only the theoretical relevance but also the practical significance of metaheuristic approaches for modeling in reliability engineering and survival analysis.

Overall, this study makes a significant methodological contribution by demonstrating that metaheuristic optimization is not only viable but superior to classical methods for MLE of the TWD — a distribution characterized by a complex, nonlinear likelihood function. Our findings empower researchers and practitioners in reliability and survival analysis with a more accurate, stable, and globally convergent estimation approach, paving the way for broader adoption of metaheuristics in parametric distribution fitting.

Despite their advantages, metaheuristic algorithms may require careful tuning of parameters such as population size, number of iterations, and any other user defined parameters to optimize performance. Future research could explore alternative metaheuristics, such as Ant Colony Optimization or Simulated Annealing, and assess their comparative efficacy.

References

Abbasi, B., Eshragh Jahromi, A. H., Arkat, J., & Hosseinkouchack, M. (2006). Estimating the parameters of Weibull distribution using simulated annealing algorithm. *Applied Mathematics and Computation*, *183*(1), 85–93. https://doi.org/10.1016/j.amc.2006.05.063

Aryal, G. R., & Tsokos, C. P. (2011). Transmuted Weibull Distribution: A Generalization of the Weibull Probability Distribution. *European Journal of Pure and Applıed Mathematıcs*, *4*(2), 89–102. www.ejpam.com

Barlow, R. E., Toland, R. H., & Freeman, T. (1984). A Bayesian analysis of stress-rupture life of kevlar 49/epoxy spherical pressure vessels. *Proc. Conference on Applications of Statistics', Marcel Dekker, New York*.

Bjerkedal, T. (1960). *Acquisition of Resistance in Guinea Pies infected with Different Doses of Virulent Tubercle Bacilli.*

Borges, P., & Campos, M. (2023). A Metaheuristic Approach to Parameter Estimation of a Flexible Parametric Mixture Cure Rate Model with Interval-Censored Data. *Trends in Computational and Applied Mathematics*, *24*(3), 535–555. https://doi.org/10.5540/tcam.2023.024.03.00535

Faouri, A. O., & Kasap, P. (2023). Maximum Likelihood Estimation for the Nakagami Distribution Using Particle Swarm Optimization Algorithm with Applications. *Necmettin Erbakan Üniversitesi Fen ve Mühendislik Bilimleri Dergisi*. https://doi.org/10.47112/neufmbd.2023.19

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan press Ann Arbor MI.

Jiang, H., Wang, J., Dong, Y., & Lu, H. (2015). Comprehensive assessment of wind resources and the low-carbon economy: An empirical study in the Alxa and Xilin Gol Leagues of inner Mongolia, China. In *Renewable and Sustainable Energy Reviews* (Vol. 50, pp. 1304–1319). Elsevier Ltd. https://doi.org/10.1016/j.rser.2015.05.082

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

Karakoca, A., & Pekgör, A. (2019). Maximum Likelihood Estimation of the Parameters of Progressively Type-2 Censored Samples From Weibull Distribution Using Genetic Algorithm. *Academic Platform Journal of Engineering and Science*, 1–1. https://doi.org/10.21541/apjes.452564

Kasap, P., & Faouri, A. O. (2024). Comparison of the Meta-Heuristic Algorithms for Maximum Likelihood Estimation of the Exponentially Modified Logistic Distribution. *Symmetry*, *16*(3). https://doi.org/10.3390/sym16030259

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Kılıç, M. B. (2022). Artificial Bee Colony and Genetic Algorithms for Parameters Estimation of Weibull Distribution. In *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems* (pp. 309–325). Springer.

Lai, C. D., Xie, M., & Murthy, D. N. P. (2003). A modified Weibull distribution. *IEEE Transactions on Reliability*, *52*(1), 33–37.

Lee, C., Famoye, F., & Olumolade, O. (2007). Beta-Weibull distribution: some properties and applications to censored data. *Journal of Modern Applied Statistical Methods*, *6*, 173–186.

Liu, F. J., Chen, P. H., Kuo, S. S., Su, D. C., Chang, T. P., Yu, Y. H., & Lin, T. C. (2011). Wind characterization analysis incorporating genetic algorithm: A case study in Taiwan Strait. *Energy*, *36*(5), 2611–2619. https://doi.org/10.1016/j.energy.2011.02.001

Mudholkar, G. S., Srivastava, D. K., & Kollia, G. D. (1996). A generalization of the Weibull distribution with application to the analysis of survival data. *Journal of the American Statistical Association*, *91*(436), 1575–1583.

Owoloko, E. A., Oguntunde, P. E., & Adejumo, A. O. (2015). Performance rating of the transmuted exponential distribution: an analytical approach. *SpringerPlus*, *4*(1), 1–15. https://doi.org/10.1186/s40064-015-1590-6

Owoloko, E. A., Oguntunde, P. E., & Adejumo, A. O. (2016). A Comparative Analysis on the Performance of the Convoluted Exponential Distribution and the Exponential Distribution in terms of Flexibility. *Journal of Mathematics and Statistics*, *12*(1), 59–64.

Pal, M., Ali, M. M., & Woo, J. (2006). Exponentiated weibull distribution. *Statistica*, *66*(2), 139–147.

Pobočíková, I., Sedliačková, Z., & Michalková, M. (2018). Transmuted Weibull distribution and its applications. *MATEC Web of Conferences*, *157*. https://doi.org/10.1051/matecconf/201815708007

Ravindra, K., Rao, R. S., & Narasimham, S. V. L. (2012). Wind distribution analysis incorporating Artificial Bee Colony Algorithm. *2012 International Conference on Advances in Power Conversion and Energy Technologies (APCET)*, 1–6. https://doi.org/10.1109/APCET.2012.6302073

Reddy, M. J., & Singh, V. P. (2014). Multivariate modeling of droughts using copulas and meta-heuristic methods. *Stochastic Environmental Research and Risk Assessment*, *28*(3), 475–489. https://doi.org/10.1007/s00477-013-0766-2

Shaw, W. T., & Buckley, I. R. C. (2009). *The alchemy of probability distributions: beyond Gram-Charlier expansions, and a skew-kurtotic-normal distribution from a rank transmutation map*. http://arxiv.org/abs/0901.0434

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.

Wadi, M., & Elmasry, W. (2021, March 23). Modeling of Wind Energy Potential in Marmara Region Using Different Statistical Distributions and Genetic Algorithms. *2021 International Conference on Electric Power Engineering - Palestine, ICEPE-P 2021*. https://doi.org/10.1109/ICEPE-P51568.2021.9423471

Wang, J., Huang, X., Li, Q., & Ma, X. (2018). Comparison of seven methods for determining the optimal statistical distribution parameters: A case study of wind energy assessment in the large-scale wind farms of China. *Energy*, *164*, 432–448. https://doi.org/10.1016/j.energy.2018.08.201

Xu, M., Droguett, E. L., Lins, I. D., & das Chagas Moura, M. (2017). On the q-Weibull distribution for reliability applications: An adaptive hybrid artificial bee colony algorithm for parameter estimation. *Reliability Engineering & System Safety*, *158*, 93–105. https://doi.org/https://doi.org/10.1016/j.ress.2016.10.012

Yalçınkaya, A., Şenoğlu, B., & Yolcu, U. (2018). Maximum likelihood estimation for the parameters of skew normal distribution using genetic algorithm. *Swarm and Evolutionary Computation*, *38*, 127–138. https://doi.org/10.1016/j.swevo.2017.07.007