Decision Analytics Journal 8 (2023) 100294

Contents lists available at ScienceDirect

Decision Analytics Journal

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

A comparative analysis of global optimization algorithms for surface electromyographic signal onset detection

Shafiq Alam [a,*], Xin Zhao [b], Imran Khan Niazi [c], Muhammad Sohaib Ayub [d], Muhammad Asad Khan [e]

[a] School of Management, Massey University, Auckland, New Zealand
[b] School of Computing, Whitireia Community Polytechnic, Auckland, New Zealand
[c] New Zealand College of Chiropractic,//AUT//AAU, Auckland, New Zealand
[d] Department of Computer Science, Lahore University of Management Sciences, Lahore, Pakistan
[e] Department of Telecommunication, Hazara University, Mansehra, Pakistan

ARTICLE INFO

Keywords:
Surface electromyography
Onset detection
Metaheuristic optimization algorithm
Genetic algorithm
Ant colony optimization
Particle swarm optimization
Tabu search
Simulated annealing

ABSTRACT

Surface Electromyography (sEMG) is a technique for measuring muscle activity by recording electrical signals from the surface of the body. It is widely used in fields such as medical diagnosis, human–computer interaction, and sports injury rehabilitation. The detection of the onset and offset of muscle activation is a longstanding challenge in sEMG analysis. This study pioneers the implementation, configuration, and evaluation of Particle Swarm Optimization (PSO) against other optimization algorithms for sEMG signal detection, including Genetic algorithms (GA), Simulated Annealing (SA), Ant Colony Optimization (ACO), and Tabu Search (TS). The results show that the PSO algorithm achieves the highest median accuracy and F1-Score and is the fastest among the selected algorithms but has lower stability compared to Genetic algorithms and Ant colony optimization. The design and value of the cost function had a significant impact on the results, with optimal results obtained when the cost value was between 0.1203 and 0.1384. The use of these algorithms improved detection efficiency and reduced the need for manual parameter adjustment. To the best of our knowledge, no published studies have utilized Simulated Annealing, Ant colony optimization, and Tabu search meta-heuristic algorithms to detect sEMG signal onsets.

1. Introduction

Surface Electromyography (sEMG) is a technique that records electrical signals from the surface of the human body to measure muscle activity [1]. It has a range of applications, such as medical diagnosis, human–computer interaction, and sports injury rehabilitation [2,3]. Despite its usefulness, identifying the onset and offset of muscle activation in sEMG analysis remains a challenge, which has been the focus of research for many years. To tackle this issue, researchers have developed automatic detection algorithms, such as the threshold algorithm and the maximum likelihood algorithm, which have been effective but are not fully automated and require manual input of parameters [4,5].

The threshold algorithm and maximum likelihood algorithm are widely used and established detection methods in the field of sEMG signal analysis [5–10], however, their accuracy is contingent on the correct input of parameters, which is traditionally accomplished through manual estimation by experts [4]. This manual input process detracts from the automation of the detection process and increases the risk of inaccurate results. Recently, the integration of optimization algorithms has opened up new avenues for automatic parameter estimation and improved the accuracy of the detection process.

Some studies have utilized meta-heuristic optimization algorithms in the detection process to determine the optimal parameters. This involves creating a cost function and then utilizing optimization algorithms to minimize it, resulting in the identification of the parameters that yield the lowest value of the cost function.

Recent studies have shown the efficacy of meta-heuristic optimization algorithms in detecting sEMG signal onsets and offsets. These algorithms, using random search, can handle a broad range of problems and identify optimal parameters by iteratively minimizing the objective function [11,12]. This reduces the manual effort involved in adjusting parameters and improves the detection efficiency of the algorithm. For instance, a study by Rashid et al. [5] combined Particle Swarm Optimization (PSO) and extended Double Threshold Algorithm

* Corresponding author.
E-mail addresses: salam1@massey.ac.nz (S. Alam), xin.zhao@whitireianz.ac.nz (X. Zhao), imran.niazi@nzchiro.co.nz (I.K. Niazi), 15030039@lums.edu.pk (M.S. Ayub), asadkhan@hu.edu.pk (M.A. Khan).

https://doi.org/10.1016/j.dajour.2023.100294
Received 13 May 2023; Received in revised form 22 July 2023; Accepted 26 July 2023
Available online 29 July 2023

(eDTA) to deduce global parameters and detect onsets/offsets pairs automatically. This resulted in the detection accuracy of over 95%, reducing the dependence on parameters estimated by experts. Another study by Magda et al. [4] used Genetic Algorithms (GA) to optimize the threshold detector, and it was observed that GA could obtain global parameters automatically.

However, these studies have only focused on the use of a single optimization algorithm and there has been limited comparison of the performance of different algorithms. It remains unclear whether there are differences between optimization algorithms when applied to sEMG signal analysis and what their advantages and disadvantages are in various scenarios.

The aim of this study is to compare and analyze the impact of different optimization algorithms on the detection of sEMG signal onsets and offsets. The optimization algorithms under examination are Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Tabu Search (TS), and Simulated Annealing (SA). These algorithms are used to determine the input parameters for the onsets/offsets detection algorithm.

The following are the main contributions of this research work.

1. A comprehensive review establishes the need for comparative evaluation and highlights research gaps in sEMG signal detection optimization algorithms.
2. The paper compares the detection quality of the optimization algorithms, including Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Tabu Search (TS), and Simulated Annealing (SA). The optimization algorithms provide similar quality with small median and interquartile range differences. Particle Swarm Optimization (PSO) outperforms others, and Simulated Annealing (SA) performs relatively poorer.
3. The paper evaluates algorithm efficiency based on runtime. Particle Swarm Optimization (PSO) is the fastest, showcasing its efficiency. Simulated Annealing (SA) is the slowest.
4. The paper assesses algorithm stability. Genetic Algorithms (GA) and Ant Colony Optimization (ACO) show higher stability (lower interquartile range (IQR)). Particle Swarm Optimization (PSO) and Tabu Search (TS) exhibit higher variability. Simulated Annealing (SA) has a higher runtime interquartile range (IQR) after setting a cost value threshold.
5. The paper identifies factors influencing algorithm performance. Muscle activity count < 21 leads to unacceptable accuracy. Different algorithms yield significantly different optimal parameters. Cost function design impacts results (Accuracy, F1-Score, Degree Over Detection (OD), Degree Under Detection (UD)).
6. The paper provides practical insights. Optimal detection results were achieved within the cost value range of 0.1203 to 0.1384. It helps to guide cost value selection for real-world detection tasks.

To assess the differences and benefits, and drawbacks of each algorithm, the detection results generated using their respective parameters are compared. Our comparison framework is based on a confusion matrix that takes into account three key metrics, i.e., quality, efficiency, and stability.

Section 2 of the rest of the paper introduces the background information, including the introduction to the sEMG signals, their detection, and the basic principles of heuristic optimization algorithms. Section 3 provides a review of the various algorithms used for onsets/offsets detection, with a focus on the selected optimization algorithms and their core processes. Section 4 details the experimental design, including the setup of the algorithms, the data source, the validation method, evaluation metrics, and tools used in the experiment. Sections 5 and 6 analyze and discuss the results of the experiment using various matrices, comparing the differences between the algorithms. Finally, the last section concludes the study, summarizing the findings and limitations and outlining future work.

2. Background

In this section, the background information related to the detection of onsets and offsets in Surface Electromyography (sEMG) signals is presented. The characteristics and applications of sEMG signals are discussed, as well as the technologies used for onset and offset detection. The commonly utilized meta-heuristic optimization algorithms in this research area are also covered. Additionally, the current state of the field is presented, emphasizing the challenges and limitations of current techniques and highlighting the gaps in research.

2.1. Surface Electromyography (sEMG)

EMG, or Electromyography, is a method for measuring and recording the electrical activity of muscles. This information can be used to identify neuromuscular abnormalities and gauge muscle performance. Specifically, surface EMG (sEMG) measures electrical impulses on the skin's surface to reflect the nervous system's control of shallow muscle contraction [1,13]. Surface Electromyography (sEMG) involves using sensors placed on the skin to monitor and document the electrical activity in the muscles beneath. The collected signals can be easily analyzed to reveal details about the pattern of muscle contractions, their intensity, and the general state of muscle function. Owing to its accessibility and usefulness, the analysis of sEMG signals has become a widely studied area in rehabilitation and exercise science [4,14]. For instance, in the field of rehabilitation medicine, sEMG signals are utilized to determine movement intentions and assist individuals with disabilities in regaining movement capabilities [15].

The analysis of Surface Electromyography (sEMG) signals can be difficult due to the weak nature of these signals, which typically have amplitudes of less than 5 mV and a frequency range between 6–500 Hz, with the most common range being 20–150 Hz [16]. The accuracy of analysis results can be impacted by the presence of noise sources such as power line interference, radio transmission, crosstalk from other muscle signals, and the electrical conductivity of skin tissue and fat [4,17]. The significance of utilizing effective tools and algorithms is emphasized by the difficulties faced in analyzing sEMG signals. This study aims to detect the onset and offset, and the algorithms utilized in this process are discussed in the literature review section.

2.2. sEMG onsets/offsets detection

Accurate detection of the beginning and end of muscle contractions, known as onsets and offsets, is important in areas such as motion control and muscle ability assessment. Onsets and offsets are typically indicated by a change in the voltage value of the signal and correspond to the start and end of muscle activity [4,15]. The duration between the onsets and offsets represents the length of muscle activity. Examples of its significance include proper analysis of normal and abnormal swallowing processes [18].

The detection of onsets and offsets in sEMG signals is of utmost importance in the field of human health, where accuracy, reliability, and efficiency are critical factors. The threshold detection algorithm and the maximum likelihood detection algorithms are two common methods for onsets/offsets detection [2,6]. However, these algorithms often require expert input for the initial parameters, which can limit their automation and accuracy. With recent advancements in optimization algorithms, there is a growing interest in using these methods to automatically determine the parameters and maximize the automation of onsets/offsets detection [19–21].

2.3. Optimization algorithms and metaheuristic optimization algorithms

Optimization is a method used to find the best solution among multiple options for a given problem. It involves transforming the problem into a mathematical model, where variables, constraints, and an objective function are defined. Optimization algorithms then aim to find a solution that minimizes the value of the objective function [12]. Examples of optimization problems include finding the shortest route for a pipe-laying project and selecting parameters to minimize project costs [22].

The field of optimization algorithms encompasses a diverse set of classification methods, and there is a lack of agreement among academic experts. To provide an overview of the meta-heuristic optimization algorithms used in this study, we adopt the categorization method proposed by Yang [23]. The algorithms can be divided into two broad categories based on the presence of randomness: deterministic and stochastic [23]. Deterministic algorithms operate under the assumption that a unique solution to an optimization problem can be found through differentiation or enumeration [24]. The calculation process does not involve any random elements, meaning that the same input always results in the same output [12]. The gradient descent algorithm falls under this category, for example. However, deterministic algorithms are limited in their application and are not suitable for solving complex problems due to the requirement that the problem has a clear mathematical expression and that the objective function is continuous and differentiable [24].

Random optimization algorithms, such as meta-heuristic algorithms, incorporate random elements into the algorithm, for instance, by randomly generating the starting point or adding random movements to increase the search area [25]. Meta-heuristic optimization algorithms use randomness to broaden the search space and produce varied solutions, preventing them from becoming trapped in local optimality [26]. They also incorporate a local search method, balancing the diversity of random solutions with a concentrated search, allowing them to find the best possible solution at a reasonable cost [27]. Unlike deterministic optimization algorithms, meta-heuristic optimization algorithms do not necessitate a precise mathematical expression of the objective function and do not rely on its continuous and differentiable properties [24]. Therefore, they can be employed to solve nearly any optimization problem. However, they do not guarantee that the obtained solution is globally optimal, as opposed to deterministic algorithms, which provide a unique optimal solution [26].

Previous research has acknowledged the sensitivity of onset/offset detection algorithms to their parameters, with the detection accuracy being contingent upon the accuracy of the input parameters [4]. However, there is a lack of a mathematical theory to guide the selection of these parameters [2]. Given that the detection algorithms lack an explicit analytical expression and are not continuous and differentiable, the use of a meta-heuristic algorithm appears to be an appropriate solution.

Some of the well-known meta-heuristic algorithms are Genetic Algorithms (GA) based on Darwinian evolution, the Simulated Annealing (SA) algorithm modeled after the process of metal annealing, the Ant Colony Optimization (ACO) algorithm, the Particle Swarm Optimization (PSO) algorithm that imitates the behavior of natural animal groups, and the Tabu Search (TS) algorithm that mirrors human memory [28,29]. These meta-heuristic optimization algorithms have been widely used and have been demonstrated to possess various benefits [30,31]. This study focuses on GA, Simulated Annealing (SA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Tabu Search (TS), and a comprehensive examination of their applications, advantages, and disadvantages is presented in the literature review section.

3. Existing research

Existing research on the automated detection of Surface Electromyography (sEMG) signal onsets and offsets has resulted in the creation of multiple algorithms, which can be grouped into three categories based on their role in the detection process. The first category encompasses algorithms specifically designed for onsets and offsets detection, such as the Threshold Algorithm and the Maximum Likelihood Algorithm. The second category encompasses noise reduction algorithms such as the Teager–Kaiser Energy Operator and the Wavelet Transform. The third category consists of optimization algorithms, which are used to determine input parameters for the onsets and offsets detection function.

3.1. Detection algorithms

Manual observation by experienced experts was previously used for the detection of onsets/offsets. However, this method is not reliable or efficient due to the dependence on the expert's experience and knowledge [32]. Consequently, researchers have concentrated their efforts on creating automated techniques for identifying the start and end of sEMG signals. The two widely employed algorithms for this purpose are the threshold detector algorithm and the maximum likelihood (ML) algorithm.

3.1.1. Threshold detector

The threshold detection algorithm relies on the idea that when muscle activity starts, the amplitude of the muscle signal exceeds a set threshold [6]. The first single threshold algorithm was proposed in a study by Hodges and Bui [6] which compared and analyzed 27 sEMG processing methods and determined the optimal parameter combination and threshold for onset detection through statistical analysis.

The concept of a double threshold detection algorithm was first introduced in the research by Hodges and Bui [6] and further improved upon by Micera et al. [9]. The work of Severini et al. [8] further improved the double threshold algorithm, and the latest improvement is the extended Double Threshold Algorithm (eDTA) proposed by Rashid et al. [5]. The eDTA is capable of adapting to changes in the signal-to-noise ratio (SNR) and adjusts the detection parameters accordingly, enhancing the performance of the double-threshold detection algorithm.

Threshold detection algorithms have the benefit of being simple and straightforward to implement. However, the manual specification of detection parameters by experts before detection can lead to inconsistencies in the results, as the quality of these parameters depends on the expert's knowledge and experience [2].

3.1.2. Maximum Likelihood (ML) algorithm

The Maximum Likelihood algorithm is a statistical approach that determines the distribution of the signal model first and then uses the baseline sample to estimate the model parameters [2]. Gaussian and Laplace distributions are commonly used as conventional models. The GLR algorithm was proposed by Micera et al. [9]. It is less sensitive to low sEMG activity than the threshold algorithm but still relies on the muscle activity and baseline activity's probability density functions and signal characteristics [10].

On the other hand, Selvan et al. [10] proposed the PLM algorithm based on Laplace Probability Distribution Function (PDF), which overcomes some of the limitations of the ML algorithm. However, ML algorithms require more information about the signal compared to the threshold algorithms, which are straightforward and simple to implement [2].

3.2. Auxiliary algorithms

Auxiliary algorithms are frequently employed before carrying out the detection process. This is because preprocessing the signal has been shown to enhance detection performance [2]. Two common preprocessing techniques are the Teager–Kaiser energy operator and wavelet template matching. The Teager–Kaiser algorithm increases the separation between the muscle activity signals and any background noise [33], while wavelet template matching separates the muscle activity signals from the noise. These algorithms have been shown to lead to improved detection results and have become essential components in the sEMG signal detection process, typically being applied before conducting any analysis.

3.2.1. Teager–Kaiser Energy Operator (TKEO)

The Teager–Kaiser Energy Operator (TKEO) was first introduced in the 1990s for processing audio signals, but it has since been applied in sEMG analysis [34]. The TKEO calculates the energy of muscle signals during contraction based on fluctuations in their amplitude and frequency, which improves muscle activity analysis [33].

TKEO has been employed in various ways to enhance sEMG analysis, such as reducing sEMG noise and boosting the energy of muscle activity movement units [35]. Additionally, it has been utilized to identify the onset of sEMG signals [18]. In a study, TKEO was integrated into three commonly used detection algorithms: visual detection, threshold method, and approximated generalized likelihood ratio [33].

The results indicated that incorporating TKEO into popular sEMG onset detection methods improved their performance without increasing their complexity. The authors, therefore, recommended using TKEO as a standard component in sEMG onset detection algorithms.

3.2.2. Wavelet template matching

The authors of Restrepo-Agudelo et al. [18] combined the Teager–Kaiser Energy Operator (TKEO) and wavelet template matching to eliminate sEMG signal noise for infrahyoid analysis and then applied the threshold algorithm to extract muscle activity signals. This approach demonstrated promising results in enhancing the performance of sEMG onset detection.

3.3. Optimization method applied in sEMG onset detection

Selection and adjustment of parameters play a crucial role in the accuracy of the detection results obtained from the algorithms. Threshold detection algorithms are highly sensitive to these parameters and require experts to set their values beforehand. On the other hand, statistical algorithms such as ML functions have internal parameters that need to be determined. Typically, researchers use test or simulation data to determine the optimal values of these parameters through repeated verification [2].

Recent advancements in machine learning theory have paved the way for the use of optimization algorithms to determine the input parameters of detection algorithms. This is done by minimizing the value of the objective function, improving the accuracy of the detection results.

3.3.1. Particle Swarm Optimization (PSO) algorithm

The Particle Swarm Optimization (PSO) algorithm, introduced by Eberhart and Kennedy [36], is a meta-heuristic optimization method that mimics the predatory behavior of birds to find the optimal solution. The algorithm begins with a group of random particles and updates their position and velocity vectors in each iteration based on the best solution found by each particle (pBest) and the best solution found by the entire swarm (gBest). In a study, Rashid et al. [5] used the Particle Swarm Optimization (PSO) algorithm in conjunction with an eDTA to automatically detect onsets/offsets pairs and determine the global detection parameters. This research showed that utilizing PSO reduced the dependence on parameters set by experts and improved the accuracy of the detection results to over 95%.

The basic pseudo code of PSO is given in Algorithm 1, which works in an iterative manner as follows. Initially, every particle p_i is randomly initialized (Line 1) and the location and velocity are assigned to p_i (Line 3). Let T be the count of total iterations (Line 4), and in each iteration t, the fitness value of p_i is calculated based on fitness function f (Line 7). Let p_i^{opt} be the best fitness value achieved by p_i in previous iterations, and if in iteration t, the fitness value of $p_i(t)$ is better than p_i^{opt}, we update p_i^{opt} to $p_i(t)$ (Line 9). Similarly, we update p_{opt}, the optimal value achieved by any of the particles in the swarm (Line 11). In the end, the position and velocity of each particle are updated (Line 12, Line 13).

Procedure 1 Particle Swarm Optimization

1: $P \leftarrow \{p_1, p_2, \ldots, p_N\}$ ▷ population of randomly initialized N particles
2: **for each** $p_i \in P$ **do**
3: initialize v_i
4: let $T \leftarrow$ iteration count
5: **while** $t < T$ **do**
6: **for each** $p_i \in P$ **do**
7: calculate $f\left(p_i(t)\right)$
8: **if** $f\left(p_i(t)\right) > f(p_i^{opt})$ **then**
9: $p_i^{opt} \leftarrow p_i(t)$
10: **if** $f\left(p_i(t)\right) > f(p_{opt})$ **then**
11: $p_{opt} \leftarrow p_i(t)$
12: $p_i(t+1) \leftarrow p_i(t) + v_i(t+1)$
13: update v_i

3.3.2. Genetic Algorithm (GA)

The Genetic Algorithm (GA), introduced by Holland [25], is a global search and optimization method that emulates the biological process of natural selection. It starts with an initial population and performs selection, reproduction, crossover, and mutation operations to generate new generations and find the optimal solution. In a study by Magda et al. [4], GA was used to optimize the threshold detector. The results showed that the GA was able to find optimal global parameters automatically.

The pseudo-code of the GA algorithm is shown as Algorithm 2. The algorithm starts by choosing the initial population and initializes it (Line 1). The fitness of each individual is evaluated to find the overall fitness of the population (Line 2). Survival of the fittest is done by performing multiple iterations on the initially selected population (Line 3). The best-ranking individuals are selected as parents to reproduce (Line 4). Crossover and mutation operations are performed to breed new generation P^* (Line 5). The fitness of each individual in P^* is calculated (Line 6). Worst-ranked individuals of the population (Line 7) are replaced by the new generation P^* until the best individuals remain in the population (Line 8).

Procedure 2 Genetic Algorithms

1: $P \leftarrow$ ChooseInitialPopulation()
2: Fitness(P)
3: **while** termination condition not met **do**
4: $B \leftarrow$ BestIndividuals(P)
5: $P^* \leftarrow$ CrossOver(B) & Mutation(B)
6: Fitness(P^*)
7: $W \leftarrow$ WorstRanked(P)
8: $P^+ \leftarrow$ Replace(P^*, W)

The genetic algorithm is better than predictable artificial intelligence in terms of robustness. Dissimilar to older artificial systems, they do not break easily even if the inputs are altered somewhat, huge state-space, or in the occurrence of reasonable noise. Moreover, in searching

an n-dimensional surface or multi-modal state-space, a GA may deliver significant benefits over the more usual search of optimization techniques, e.g., praxis, heuristic, breath-first, linear programming, and depth-first.

3.4. Machine learning and optimization algorithms

Several recent studies have addressed various aspects of optimization algorithms and machine learning techniques. Singh et al. proposed a load balancing and service discovery approach using Docker Swarm for microservice-based big data applications [37]. Slathia et al. presented a performance evaluation of a situational-based fuzzy linear programming problem for job assessment [38]. Pooja et al. conducted an analysis of clustering algorithms for facility location–allocation problems [39]. Anupong et al. utilized deep learning algorithms to generate photovoltaic renewable energy in saline water analysis via an oxidation process [40]. Mekala et al. proposed an efficient LiDAR-trajectory affinity model for autonomous vehicle orchestration [41].

In the field of sentiment analysis, Vyas et al. developed RUemo, a classification framework for Russia-Ukraine war-related societal emotions on Twitter through machine learning [42]. Shanmuganathan et al. reviewed state-of-the-art load-balancing algorithms for the mist-fog-cloud-assisted paradigm [43]. Sharma et al. conducted software-based sentiment analysis of clinical data for the healthcare sector [44]. Dhiman et al. proposed a blockchain-based covert software information transmission method for Bitcoin [45].

In the domain of optimization algorithms, Nayak et al. presented a comprehensive review of particle swarm optimization over the past 25 years [46]. Zhen et al. proposed an intelligent-based ensemble deep learning model for security improvement in real-time wireless communication [47]. Dhiman and Kumar introduced the spotted hyena optimizer and emperor penguin optimizer, novel bio-inspired metaheuristic techniques for engineering applications [48,49]. Kaur et al. proposed the tunicate swarm algorithm, a new bio-inspired metaheuristic paradigm for global optimization [50]. Dhiman and Kaur presented STOA, a bio-inspired optimization algorithm for industrial engineering problems [51].

Other studies focused on specific applications. Kumar and Dhiman conducted a comparative study of fuzzy optimization through fuzzy number [52]. Chatterjee reviewed the relationship between artificial intelligence and patentability [53]. Vaishnav et al. performed an analytical review analysis for screening COVID-19 [54]. Gupta et al. developed a crime tracking system using machine learning approaches for people's safety in India [55]. Sharma et al. applied transfer learning and convolutional neural networks for breast cancer image classification [56]. Shukla et al. proposed a self-aware execution environment model for improving the performance of multicore systems [57]. Dhiman and Kumar proposed the Seagull Optimization Algorithm for large-scale industrial engineering problems [58], while Dhiman et al. introduced the Rat Swarm Optimizer for global optimization [59]. In another work, Dhiman et al. presented BEPO, a novel binary emperor penguin optimizer for automatic feature selection [60]. Dhiman developed ESA, a hybrid bio-inspired metaheuristic optimization approach for engineering problems [61].

Overall, these papers contribute to advancing optimization algorithms and various machine learning application domains, providing valuable insights and novel solutions to their respective research areas.

3.5. Meta-heuristic optimization algorithms

As evident from the successful implementation of PSO and GA in sEMG onsets/offsets detection, the process of determining the value of parameters for the onsets/offsets detector can be considered an optimization problem. Motivated by the successful application of PSO and GA, this project aims to study and compare the feasibility of three other commonly used heuristic search algorithms – Simulated Annealing (SA), Ant Colony Optimization (ACO), and Tabu Search (TS) – in sEMG signal detection. In this section, we examine the key concepts, underlying processes, as well as the pros and cons of these candidate algorithms.

3.5.1. Simulated Annealing (SA)

The Simulated Annealing (SA) algorithm, introduced by Kirkpatrick et al. [62], is a heuristic global optimization technique that simulates the annealing process of solid materials. Just as in the heating and cooling process, the particles in a solid become organized from disorder, leading to a reduction in internal energy to its minimum.

The Simulated Annealing (SA) algorithm operates by starting with a high temperature and gradually decreasing it, using Metropolis rules to randomly search for the optimal global solution of the target function within the solution space. The algorithm enables the acceptance of solutions that are worse than the current one, allowing it to escape local extrema and reach the global optimum [62].

Simulated Annealing (SA) is a stochastic optimization algorithm that is used to find a global optimum of a given cost function. It is inspired by the physical annealing process and works by mimicking the cooling of a material to reach a state of low energy. The algorithm generates candidate solutions using random perturbations and accepts or rejects them based on a probabilistic acceptance rule. The acceptance probability is determined by the current temperature and the difference between the current solution and the candidate solution. The temperature is gradually reduced, leading to a decrease in the acceptance of new solutions that are far from the current solution until a final state is reached. Simulated Annealing (SA) is used in various optimization problems, including the detection of signal onsets/offsets in sEMG signals.

Procedure 3 Simulated Annealing

$T \leftarrow T_{max}$
$best \leftarrow$ INIT()
while $T > T_{min}$ **do**
  $next \leftarrow$ NEIGHBOUR($T, best$)
  $\Delta E \leftarrow$ ENERGY($next$) − ENERGY($best$)
  if $\Delta E < 0$ **then**
    $best \leftarrow next$
  else if RANDOM() < ACCEPT($T, \Delta E$) **then**
    $best \leftarrow next$
  $T \leftarrow$ COOLING($T, best$)

Simulated Annealing (SA) has been applied in the analysis of sEMG signals for parameter estimation. In [63], the Simulated Annealing (SA) algorithm was used to optimize parameters in a human–computer interaction model, leading to the development of a lower limb rehabilitation robot based on sEMG signals.

3.5.2. Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) algorithm, introduced by Dorigo et al. [64], is a heuristic global optimization method inspired by the foraging behavior of ants. It simulates the behavior of ants searching for food, where each ant leaves a trail of pheromones on the path. Other ants in the colony then follow the path with the highest pheromone concentration, leading to a positive feedback loop where the most traveled path has the highest pheromone concentration. This ultimately results in the entire colony finding the shortest route to the food source [64].

The way the ants can explore the shortest trails between food sources and their nest is an intriguing phenomenon. In the real world, each ant interacts indirectly with the others through its pheromone, a chemical substance left behind when they move. More pheromones being emitted along a certain route makes that path more desirable. So, more ants follow this way and finally, it is chosen as the best path

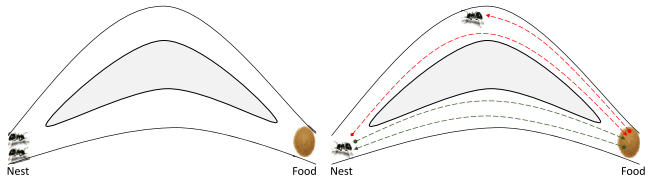

Fig. 1. Path selections by ants using pheromone.

between the source of food and the nest [65]. As shown in Fig. 1, two ant scouts choose two different paths when they start to forage from their nests. Initially, no pheromone is produced on the two paths, so these trails are selected with the same probability. However, the scout who chooses the shorter path takes less time and returns to the nest earlier than the other. Also, the pheromone accumulated on the shorter path is more than the pheromone on the longer path in the same period. Therefore, the shorter path is more likely to be chosen by other ants in the nest. As a result, the phenomenon on the shorter path is accumulated more quickly than the other, and the final shortest path from the nest to the food is determined [65].

The pseudo-code of the Ant Colony Optimization (ACO) algorithm for combinatorial problems is shown as Algorithm 4. First, parameters and pheromone trails are initialized (Line 1). Available ants generate possible solutions using the available pheromone and heuristics information (Line 3). These possible solutions can optionally be improved using a local search phase (Line 4). Finally, pheromone trails are globally updated as per the pheromone search of ants (Line 5). The above three steps are repeated until the optimal solution is achieved (Line 2).

Procedure 4 Ant Colony Optimization

```
1: Initialization of parameters and pheromone trails
2: while termination condition not met do
3:     ACOSolutionsConstruction( )
4:     PossibleDaemonAction( )          ▷ For example, Local Search
5:     GlobalPheromoneUpdate( )
```

So far, no studies have been conducted to demonstrate the use of Ant Colony Optimization (ACO) for parameter estimation in sEMG onset/offset detection. However, Ant Colony Optimization (ACO) has been utilized in other aspects of sEMG analysis.

In [66], an ACO-based classifier was developed by using a Gaussian Mixture Model to cluster sEMG signals into burst and non-burst segments, then feeding these results into the ACO algorithm to extract classification rules. These rules were able to effectively detect the activation time of skeletal muscles from the sEMG signal.

Ant Colony Optimization (ACO) was also used for sEMG feature selection in [67]. The authors utilized the ACO algorithm for the selection and classification of sEMG signals.

3.5.3. Tabu Search (TS)

The Tabu Search (TS) algorithm, introduced by Glover [25], is a global optimization method that uses heuristics and a "Tabu List" to keep track of the search history and avoid previously performed operations. It starts with an initial feasible solution and uses heuristics to choose a series of search directions. The best change in the objective function is then selected for movement. The core of the TS algorithm is using a "Tabu List" to keep track of the search history, avoid previously performed operations, and incentivize desirable states based on certain criteria. The "Tabu List" serves as a short-term memory, reducing the search space and computational costs in comparison to other optimization algorithms, such as the Genetic Algorithm (GA) [68]. The basic process of the TS algorithm is depicted in Fig. 5.

In [68], a hybrid algorithm was created for parameter estimation by combining the Tabu Search (TS) algorithm with the Particle Swarm Optimization (PSO) algorithm. This hybrid algorithm was applied to the control of prosthetic hands, which is a crucial application in sEMG signal analysis. However, to date, no published studies have utilized the TS algorithm solely to detect sEMG signal onsets/offsets.

3.6. Summary of existing work

To summarize the existing work, there are three categories of algorithms used for sEMG onset/offset detection: detection algorithms, auxiliary algorithms, and optimization methods. Detection algorithms include threshold detection algorithms and maximum likelihood (ML) algorithms. Auxiliary algorithms, such as the Teager–Kaiser Energy Operator (TKEO) and wavelet template matching, are used for noise reduction and signal preprocessing of sEMG signals. Optimization methods, such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), are employed to determine the optimal parameters for detection algorithms. Additionally, meta-heuristic optimization algorithms like Simulated Annealing (SA), Ant Colony Optimization (ACO), and Tabu Search (TS) show potential for parameter estimation in sEMG analysis.

The core of the detection process involves using onsets/offsets detection algorithms, which can either be machine learning (ML) or threshold-based. However, these algorithms typically require manual parameter adjustment by experts. To overcome this challenge, various researchers have incorporated meta-heuristic optimization algorithms into the detection process to determine the necessary parameters automatically. These algorithms use random search to construct a cost function and then optimize it to find the parameters that result in the minimum value.

The Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) have already been effectively utilized in sEMG onsets/offsets detection. While the Simulated Annealing (SA), Ant Colony Optimization (ACO), and Tabu Search (TS) algorithms are also meta-heuristic optimization methods, there have not yet been any published studies on their use in sEMG onset detection. Despite their different principles, strengths, and weaknesses, all the algorithms discussed in this section are potential solutions for sEMG detection. The design and implementation methods, as well as the comparison metrics, are discussed in the following section.

4. Experiments design

The aim of this study is to evaluate and compare the effectiveness and efficiency of the global optimization algorithms for the automatic detection of onsets/offsets in sEMG signals. The procedures and variables of these algorithms, the cost functions, the detection algorithm, the dataset, and the evaluation metrics used for comparison, are presented in this section.

4.1. Implementation details of optimization algorithms

This section aims to outline the key elements of the selected global heuristic optimization algorithms, PSO, GA, Simulated Annealing (SA), Ant Colony Optimization (ACO), and TS, including their implementation methods, crucial steps, and necessary parameters.

4.1.1. PSO implementation details

The PSO function in the MATLAB Global Optimization Toolbox is used in this study. The updating method for particle velocity and position is the primary method that affects the optimization result of the PSO algorithm. The speed and position update formulas are shown in Eq. (1) [36].

$$V_{id}^{k} = \omega V_{id}^{k-1} + AC_1\,\mathrm{Rand}()\left(p\ \mathrm{Best}_{id} - X_{id}^{k-1}\right) + AC_2\,\mathrm{Rand}()\left(g\ \mathrm{Best}_{d} - X_{id}^{k-1}\right)$$

Table 1
Parameters of PSO.

Swarm size (Population)	Max iteration	ω	AC_1	AC_2
24	1600	[0.1, 1.1]	1.49	1.49

$$X_{id}^{k} = X_{id}^{k-1} + V_{id}^{k-1} \quad (1)$$

The algorithm uses several parameters to regulate the solution search space, including ω as the inertia weight, AC_1 and AC_2 as acceleration constants, and the random function Rand(). The particles' best positions, both individually (pBest) and globally (gBest), are also considered in the algorithm. The velocity and position V_{id} and X_{id} of each particle are updated based on these parameters in each iteration.

The value of ω is established through an adaptive random adjustment process in MATLAB, with a range from 0.1 to 1.1 as described in [69]. The optimal position of six neighboring particles, rather than the global optimal position, is used to obtain the gBest value. Other parameters of the PSO algorithm include the swarm size and the maximum number of iterations. The swarm size is set to three times the number of parameters in the detection function, which has eight input parameters, resulting in a particle swarm of 24 particles. The maximum iteration value is set to 200 times the number of parameters, equaling 1600. The crucial parameters are summarized in Table 1.

The chosen swarm size of 24 is justified by striking a balance between exploration and exploitation, enabling effective exploration of the solution space. This balance ensures that the algorithm has sufficient diversity to explore different regions while converging toward promising solutions. The maximum iteration limit of 1600 is justified by providing ample iterations for the algorithm to refine and improve solutions. A higher iteration count allows for a more comprehensive exploration of the solution space, increasing the chances of finding optimal solutions. The range of the inertia weight (ω) from 0.1 to 1.1 is justified by achieving a balance between global and local search abilities. This range allows the particles to explore a wide range of solutions initially and gradually converge toward promising regions as the algorithm progresses. Setting the acceleration coefficients (AC_1 and AC_2) to 1.49 is justified by maintaining a balanced exploration–exploitation trade-off. These coefficients control the influence of personal and global best positions on particle movements, enabling effective exploration while exploiting promising solutions. These parameter choices aim to optimize the performance of the PSO algorithm, but further fine-tuning may be required to accommodate specific problem characteristics and achieve optimal results.

4.1.2. GA implementation details

The GA algorithm used in this study was taken from the MATLAB Global Optimization Toolbox. The process of parent selection, crossover, and mutation greatly impacts the optimization results [70]. For this reason, "Stochastic Uniform", "Scattered", and "Gaussian" were used as the default functions for selection, crossover, and variation, respectively [71]. Another crucial factor is the "Elite Count", which refers to the number of fittest chromosomes that are allowed to survive and move on to the next generation. The default value for the Elite Count is set to 5% of the population size.

As seen in Table 2, the population size and the maximum number of generations were set to the default parameters of GA in MATLAB. The population size was set to 50 if the number of parameters was less than 5 and to 200 if it was more than 5. In this case, the number of parameters in the detection function was 8, so the initial number of chromosomes was set to 200. The maximum number of generations was set to 100 times the number of parameters of the objective function, resulting in a value of 800.

The chosen population size of 200 chromosomes facilitates diverse exploration, enhancing the likelihood of discovering a favorable solution. By allowing for a maximum of 800 generations, the genetic algorithm (GA) can thoroughly explore the solution space, potentially identifying superior solutions. The utilization of Stochastic Uniform selection aids in maintaining genetic diversity within the population by probabilistically selecting individuals based on their fitness values. Scattered crossover contributes to the preservation of diversity by facilitating the exchange of genetic material at multiple random points. Gaussian mutation introduces incremental modifications, enabling the exploration of neighboring solutions. Lastly, the preservation of the top 10 performing individuals in each generation ensures the retention and potential improvement of the most promising solutions. These parameter choices were made to optimize the GA's ability to achieve desirable outcomes and adapt to various problem scenarios.

Table 2
Parameters of GA.

Number of chromosomes (Population)	Max generations	Selection function	Crossover function	Mutation function	Elite count
200	800	Stochastic uniform	Scattered	Gaussian	10

Table 3
Parameters of Simulated Annealing (SA).

Initial temperature	Max iteration	New solution generating function	Temperature updating schedule	Acceptance function
100	4000	'annealingfast'	'temperatureexp'	'acceptancesa'

4.1.3. Simulated Annealing (SA) implementation details

The Simulated Annealing (SA) algorithm applied in this study is sourced from the Global Optimization Toolbox in MATLAB. The key factors affecting the optimization results include the initial temperature, solution generation rules, acceptance criteria for new solutions, and cooling method [22].

The input parameters for the Simulated Annealing (SA) algorithm are set to their default values in MATLAB. The initial temperature is set to 100, and the maximum number of iterations is 4000, which is calculated by multiplying the number of objective function parameters by 500.

The 'annealingfast' function, based on the Student t-distribution, is used to generate new solutions, and the 'acceptancesa' function, based on the Boltzmann probability density, is used to determine whether the new solution is acceptable or not [71]. These functions are built into the Simulated Annealing (SA) algorithm in MATLAB. The basic formula is as Eq. (2).

$$P_k = \frac{1}{1 + e^{-\frac{\Delta E_k}{T}}} \quad (2)$$

The acceptance of a new solution in the Simulated Annealing (SA) algorithm is based on both the current temperature, T, and the difference in the objective function values, ΔE_k, between the old and new solutions. If ΔE_k is negative, the new solution is readily accepted as the starting point for the next iteration. However, if ΔE_k is positive, a probability, P_k, is calculated using the Boltzmann probability density formula (as shown in Eq. (2)). The new solution is only accepted if P_k is greater than a random number generated between 0 and 1.

The temperature function applied in the Simulated Annealing (SA) algorithm is 'temperatureexp', which follows an exponential annealing schedule. The values for the temperature function and other relevant parameters are outlined in Table 3.

The initial temperature of 100 balances exploration and exploitation. The maximum iteration limit of 4000 allows for sufficient iterations to refine solutions. The 'annealingfast' new solution generating function quickly generates diverse solutions. The 'temperatureexp' temperature updating schedule gradually reduces randomness.

Table 4
Parameters of Ant Colony Optimization (ACO).

Ant colony size (Population)	Max iteration	Selection pressure (q)	Deviation-Distance ratio (ξ)
40	800	0.5	0.85

The 'acceptancesa' acceptance function balances exploration and exploitation. These choices optimize the Simulated Annealing (SA) algorithm's performance, but fine-tuning may be needed for specific problem characteristics and optimal results.

4.1.4. Ant Colony Optimization (ACO) implementation details

The Ant Colony Optimization (ACO) algorithm used in this study is sourced from the Global Optimization Toolbox in MATLAB. Originally, the ACO algorithm was used to solve discrete optimization problems but was later optimized for the continuous domain by Socha and Dorigo [72]. This optimized version is called ACOR and is well-suited for this study as the input parameters of the detection algorithm, eDTA, have continuous value distributions. The implementation of the ACOR algorithm in this study is based on the open-source code "Ant Colony Optimization for Continuous Domains (ACOR)" published in the MATLAB Central File Exchange [73].

The ACOR algorithm stores the pheromone information of the starting solution path in an archive table. The solutions are ranked based on their objective function values, with the best solution being at the top. The concentration of pheromones is calculated using Eq. (3) from [72], where "q" stands for "Selection Pressure" and "k" is the rank of each solution. The formula dictates that solutions with smaller objective function values will have a higher concentration of pheromones. The roulette algorithm is utilized in ACOR to select the path for each ant. The probability calculation in the algorithm relies on ω_l.

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{3}$$

Generating a new solution based on the selected path is another crucial step in the ACOR algorithm. The normal distribution is employed for sampling, and the standard deviation is calculated using Eq. (4) from [72]. "ξ" represents the deviation-distance ratio, and "S_i" signifies the i_{th} variable of the solution.

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{|s_e^i - s_l^i|}{k-1} \tag{4}$$

These above parameters are shown in Table 4.

The parameters chosen for the Ant Colony Optimization (ACO) algorithm, as shown in Table above, were carefully selected to ensure an effective balance between exploration and exploitation. A population size of 40 ants enables sufficient exploration of the solution space while maintaining computational efficiency. With a maximum iteration of 800, the algorithm has ample iterations to refine solutions without excessive computational cost. A selection pressure of 0.5 strikes a balance between exploiting the best-known solutions and exploring alternative paths. Additionally, a deviation-distance ratio of 0.85 emphasizes exploration moderately by considering both deviation and distance components in the pheromone update equation. These parameter choices were made to optimize the performance of the Ant Colony Optimization (ACO) algorithm, taking into account the specific characteristics of the problem at hand.

4.1.5. Tabu Search (TS) implementation details

The Tabu Search (TS) algorithm applied in this research paper is a modified version of the code presented by Xu et al. in [22] for optimizing continuous functions. The initial solution is generated randomly within the range of values for each variable. In the first half of the iteration process, candidate solutions are generated by randomly updating a single variable. During the second half of the iteration, all variables are updated randomly through the application of mutation probability.

Table 5
Parameters of Tabu search (TS).

Number of candidate solutions	Max iteration	Tabu length	Distance, Function values difference	Mutation probability
40	800	200	0.5, 0.05	0.75

Table 6
Input parameters of "extended Double Threshold Algorithm" (eDTA) algorithm.

Parameter	Description
L_b	Length of the baseline segment
K_{bth}	The rank of moving average value
N_{sd}	Number of standard deviations of the baseline segment
T_{on}	Time for detecting an onset
T_{off}	Time for detecting an offset
$T's$	Time for the shortest sEMG burst
N_{nt}	Number of standard deviations of the root means square (RMS) values of the detected bursts
T_j	Time for the window in which two or more sEMG bursts are joined into a single burst.

The core of the algorithm lies in the updating rules of the Tabu table, as described by Xu et al. in [22]. The Tabu table holds 200 solutions, and with each iteration, the best solution from the candidate set is added to the table while the oldest solution is removed to signify its expiration.

A vital step in the algorithm is determining if the candidate solution is in the Tabu table. If the candidate solution is close to a solution in the Tabu table (calculated using 2-Norm) and their difference is not significant, then the candidate solution is considered as being in the Tabu state. The relevant parameters in the Tabu Search (TS) algorithm are listed in Table 5.

The parameter selection for the Tabu Search (TS) algorithm, as depicted in Table 5, was carefully made to optimize its performance in balancing exploration and exploitation. The choice of 40 candidate solutions allows for sufficient exploration of the solution space while maintaining computational efficiency. With a maximum iteration of 800, the algorithm has an adequate number of iterations to refine solutions without excessive computational cost. The tabu length of 200 ensures that previously visited solutions are temporarily prohibited, promoting diversification in the search process. By considering both distance and function values difference, with values of 0.5 and 0.05, respectively, the algorithm can balance between exploring new solutions and exploiting the best-known ones. Finally, a mutation probability of 0.75 introduces a moderate level of randomness to the search, facilitating the exploration of different regions in the solution space. These parameter choices aim to optimize the performance of the Tabu Search (TS) algorithm, considering the specific characteristics of the problem at hand.

4.2. eDTA implementation details

The detection algorithm used in this study is based on the eDTA algorithm from [5] and involves threshold detection. It requires eight input parameters, including the sEMG signal, as listed in Table 6. The algorithm outputs onset/offset pairs. These parameters can either be manually adjusted by experts or obtained through optimization. The comparison of the optimization algorithms in this study is based on the estimation of these eight parameters.

4.3. Cost functions

In our experiments, optimization algorithms were utilized to determine the optimal input parameters by reducing the cost function value

through iterative processes [74]. In order to examine the effect of cost function variations on optimization algorithms, two cost functions were employed to calculate the detection results. These cost functions were proposed in [5].

The first cost function is based on the comparison of the number of onset/offset pairs identified by the detection algorithm and the actual number. The formula for this cost function is represented as Eq. (5) followed by its constraints and referred to as $\mathbb{C}1$ in this study [5].

$$\min_{P} \|n\left(\Lambda\left(P_i, X\right)\right) - N_e\|_2 + \frac{N_A}{N} + \frac{E_B}{E} \tag{5}$$

with constraints:

$$B^l \leqslant P \leqslant B^u$$

The second cost function used in this study, referred to as $\mathbb{C}2$, is based on expert-identified onsets and offsets, and is described by the formula in Eq. (6) [5]. The formula employs the following variables: Λ, representing the eDTA detection algorithm with input parameters P_i; B^l and B^u, the lower and upper boundaries of P_i respectively; 'n (.)', denoting the number of onsets/offsets pairs obtained by the eDTA detector; N_e, the number of onsets/offsets pairs identified by the expert; X, the input signal samples; N_A to N, representing the ratio of the number of samples between onset and offset to the total signal samples; and E_B to E, representing the ratio of the energy of the samples excluding muscle burst to the total signal energy. The energy E_B and E are calculated using the Teager–Kaiser operator.

$$\min_{P} \left|1 - C\left(\Lambda\left(P_i, X\right), R\right)\right|_2 \tag{6}$$

with Constraints:

$$B^l \leqslant P \leqslant B^u$$

In Eq. (5), the objective function aims to minimize the Euclidean norm of the difference between 1 and the output of the function $C\left(\Lambda\left(P_i, X\right), R\right)$. The function C takes the inputs $\Lambda(P_i, X)$ and R. The variable P is the variable being optimized.

The constraints specify the lower bound (B^l) and upper bound (B^u) on the variable P, ensuring that its values fall within the specified range. These constraints restrict the feasible solutions in the optimization problem.

The variable refers to concordance, which is determined by comparing the onsets/offsets identified by the detection algorithm (Λ) with the expert-identified results (R). The method of calculating concordance is the same as accuracy and is described in Section 4.6.1. The parameters (P) necessary for the detection algorithm can be obtained by optimizing these two cost functions. The assessment of these optimization algorithms mainly focuses on comparing the results of the algorithm using the parameters obtained from different optimizations.

In the study, two cost functions, $\mathbb{C}1$ and $\mathbb{C}2$, were used to evaluate and optimize the input parameters of a detection algorithm. The cost value of $\mathbb{C}1$ represents the discrepancy between the detected onset/offset pairs and the actual number, considering additional signal characteristics. The cost value of $\mathbb{C}2$ measures the agreement between the detection results and expert-identified onsets and offsets. The optimization process aims to minimize these cost values, which serve as a unitless measure of the error or discrepancy between the detected results and the ground truth or expert-identified values. The study compares different optimization algorithms based on their ability to minimize these unitless cost values and improve the accuracy and agreement with the reference data.

4.4. Dataset description

The study utilized sEMG data from 20 sets of joint activities and foot dorsiflexion from 20 participants [75]. The sEMG signals were recorded at a frequency of 500 Hz, and their amplitude was included in the dataset. To evaluate the performance of each optimization algorithm, the study compared it with expert manual identification results, which were also part of the dataset. The number of muscle activities in each dataset is shown in Table 7, providing a detailed breakdown.

Table 7
Dataset and activities.

Dataset ID	$\mathbb{D}1$	$\mathbb{D}2$	$\mathbb{D}3$	$\mathbb{D}4$	$\mathbb{D}5$	$\mathbb{D}6$	$\mathbb{D}7$	$\mathbb{D}8$	$\mathbb{D}9$	$\mathbb{D}10$
No. of activities	50	50	50	50	50	50	21	51	50	30
Dataset ID	$\mathbb{D}11$	$\mathbb{D}12$	$\mathbb{D}13$	$\mathbb{D}14$	$\mathbb{D}15$	$\mathbb{D}16$	$\mathbb{D}17$	$\mathbb{D}18$	$\mathbb{D}19$	$\mathbb{D}20$
No. of activities	16	30	14	30	25	30	18	12	22	50

4.5. Cross-validation

We used cross-validation to determine the optimal parameters of the model. Cross-validation is a technique aimed at evaluating the performance of a model on new data and avoiding overfitting, as explained in [76]. The training dataset, composed of sEMG signals and expert-labeled onsets/offsets from 5 participants, was utilized to identify the suitable input parameters (P_i). The cross-validation design followed the method outlined in [5] and implemented leave-p-out cross-validation with $p = 4$.

The cross-validation process involved removing one dataset from the training set and using different optimization algorithms to obtain the P_i parameters. The cost function values were then calculated for the remaining four datasets using these parameters, and the mean of these values was determined. The process was repeated 5 times, and the P_j vector that produced the lowest mean cost value was used as input parameters for eDTA, which was applied to 15 test datasets. The detection results were compared and analyzed against expert-identified outcomes, as illustrated in Fig. 2.

The method of leave-p-out cross-validation was applied to obtain the P_i parameters, with $p = 4$. The first step involved removing one dataset from the training set and using different optimization algorithms to obtain the P_i parameters. In the second step, the cost function values were calculated for the remaining four datasets using the P_i parameters as input. Finally, the mean of these cost values was determined. The cost function utilized in this process is shown in $\mathbb{C}2$.

4.6. Comparative analysis

Quality, efficiency, and reliability are standard metrics for evaluating optimization algorithms [77]. This study uses these three indicators as the evaluation framework and employs specific sub-indicators to compare the performance of the selected meta-heuristic optimization algorithms.

4.6.1. Quality metrics

The evaluation of predictive models often involves the use of a confusion matrix, a widely accepted method in data mining theory [78]. Confusion matrices have been used to assess the performance of onsets/offsets detection algorithms [5,18,79]. In this study, these are used to compare and analyze the quality of various optimization algorithms for EMG onsets/offsets detection. The accuracy of expert identification is assumed, and the results of the optimization algorithm are compared to the expert results to construct the confusion matrix, which is shown in Fig. 3.

In this study, the quality of different optimization algorithms for EMG onsets/offsets detection is evaluated using quality metrics based on the confusion matrix, including Accuracy, F1-Score, Degree Under Detection (UD), Degree Over Detection, Area Under Curve (AUC), and Detection Rate. These metrics are selected to compare the performance of the optimization algorithms.

Accuracy. Accuracy refers to the proportion of correctly identified signals, calculated by dividing the sum of true positive (TP) and true negative (TN) results by the total number of signals tested. The formula for accuracy is given in Eq. (7) in Refs. [80,80]. The term "Concordance" in Section 4.3 is equivalent to accuracy.

$$\text{Accuracy} = \frac{TPs + TNs}{TNs + FNs + FPs + TPs} \tag{7}$$

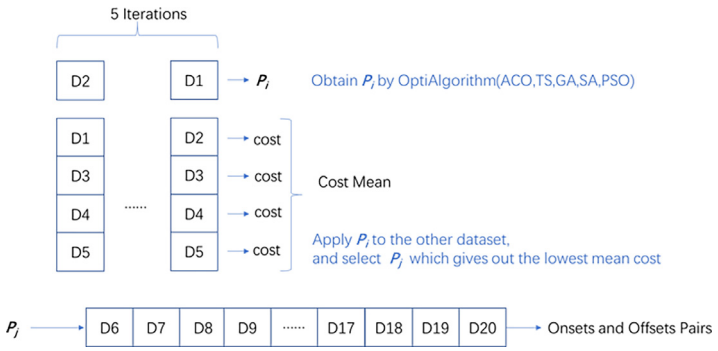

Fig. 2. Cross-validation scheme.

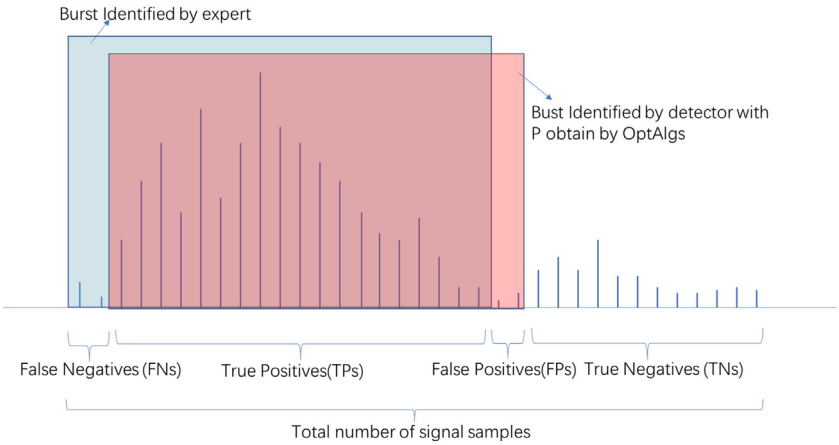

Fig. 3. Confusion matrix for sEMG.

F1-score. Besides accuracy, the Recall and Precision indices, derived from the confusion matrix, can evaluate the correctness and accuracy of positive class recognition. These two metrics are often in opposition to each other [80]. The study in [5] suggests that F1-Score, which is the harmonic mean of Recall and Precision, is appropriate for evaluating the performance of onset/offset detection algorithms. F1-Score is defined in Eq. (8) in [80].

$$\text{F1-Score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

$$\text{Recall} = \frac{TPs}{FNs + TPs}$$

$$\text{Precision} = \frac{TPs}{FPs + TPs} \tag{8}$$

Recall indicates the proportion of correctly identified bursts compared to the total actual bursts and is calculated as the ratio of true positive (TP) results to the sum of false negative (FN) and true positive results. Precision reflects the accuracy of correctly recognized burst samples relative to all samples identified as bursts and is defined as the ratio of true positive results to the sum of false positive (FP) and true positive results [80].

Degree Under Detection (UD) and degree Over Detection (OD). Degree Under Detection (UD) and Degree Over Detection (OD) are metrics introduced in [5] to measure the extent to which an algorithm fails to recognize or over-detects events. UD and OD are defined in Eqs. (9) and (10) in [5].

$$OD = \frac{FPs}{FNs + TPs} \tag{9}$$

$$UD = \frac{FNs}{TNs + FPs} \tag{10}$$

Area Under Curve (AUC). The Area Under the Receiver Operating Characteristic (ROC) curve (AUC) is a commonly used metric for classifier selection and performance evaluation [80]. In the context of this study, the detection of onsets/offsets in sEMG signals is treated as a binary classification problem, i.e., the recognition of muscular activity or non-muscular activity. The AUC is used to evaluate the performance of various detectors, which are based on different optimization algorithms, and indirectly reflects the performance of the optimization algorithms.

The ROC plot takes False Positive Rate (FPR) as the X-axis and True Positive Rate (TPR) as the Y-axis [80]. TPR and FPR are defined in Eqs. (11) and (12).

$$TPR = \frac{TP}{TP + FN} \tag{11}$$

$$FPR = \frac{FP}{FP + TN} \tag{12}$$

The AUC is calculated as the area under the ROC curve, as defined in Eq. (13). The closer the AUC value is to the upper left corner (0,1), the better the performance of the classifier, as a larger AUC value indicates better performance [80].

$$AUC = \frac{TPR - FPR + 1}{2} \tag{13}$$

In this experiment, a point on the ROC graph is calculated for the detection results of each dataset, and the AUC value is obtained.

Detection Rate: Detection Rate compares the number of onset/offset pairs detected by the optimization algorithm to the number of pairs manually identified by an expert [5]. It is expressed as the ratio of these two numbers, with a value closer to 1 indicating a better detection result.

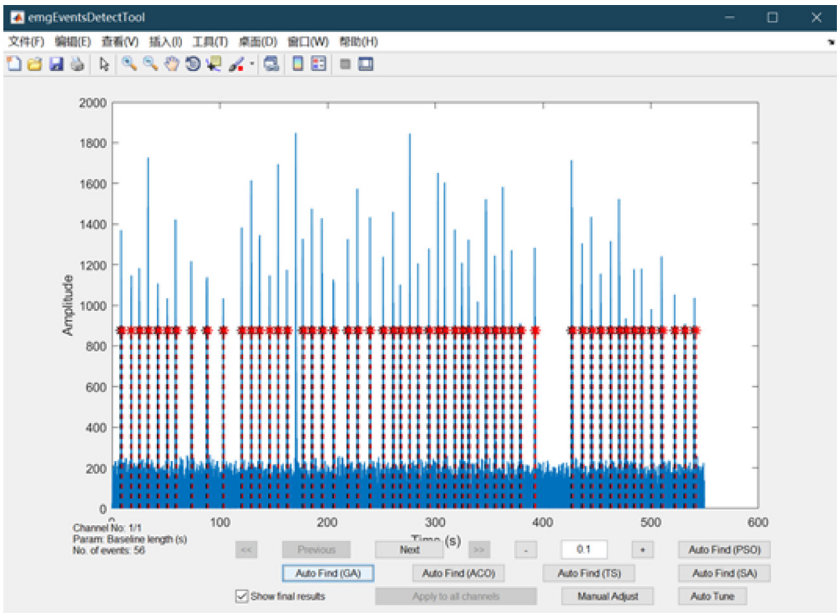

Fig. 4. ACO, TS, GA and SA optimization algorithms added to EMG detector.

4.6.2. Efficiency

Besides quality, the execution efficiency of the algorithm also impacts its practical use. For instance, real-time detection results are crucial in wearable sEMG signal detectors [81]. To measure efficiency in this study, the runtime is used as the metric [77].

Runtime: Runtime is the time taken to complete an optimization, where "completion" refers to obtaining a feasible solution [77]. This study considers a cost function value threshold as a criterion for feasibility, and the running time is recorded when the cost value falls within this range. The cost value threshold is determined during testing.

4.6.3. Reliability

Meta-heuristic algorithms do not always guarantee optimal results [12]. To evaluate their performance, reliability is often used as a metric, which is expressed as the success rate or the probability of the optimization algorithm solving the problem successfully [77]. In this study, reliability is used to measure the probability of obtaining feasible optimization parameters that allow for accurate detection results. Feasibility is defined as the precise accuracy of the detection results.

Success Rate: Success criteria is the value of the objective function or the distance between the solution and the minimum value [77]. In the detection scenario of sEMG onsets/offsets, the success rate is calculated as the number of successful optimization executions divided by the total number of executions. The specific value of the F1-Score is determined during testing.

4.7. Environment and tools

The study used MATLAB R2017b software to implement and run algorithms. The code was based on the emgGo project [5] and included the PSO algorithm. Four additional algorithms (ACO, TS, GA, SA) were added to the published emgEventsDetectTool and can be triggered by the designed interface as shown in Fig. 4 labeled AutoFind (GA), AutoFind (ACO), AutoFind (TS), and AutoFind (SA).

4.8. Experimental validation

In our study, experimental validation refers to the process of optimizing input parameters using optimization algorithms and evaluating their performance. The experiments conducted in our study are considered valid for several reasons. Firstly, we provide the details of already available code used for optimization algorithms, cost functions, and evaluation metrics, ensuring transparency and reproducibility. Secondly, the use of sEMG data from 20 participants, provided by our lab and stated in [75] along with expert-identified onsets and offsets, enhances the accuracy and reliability of the experiments. Lastly, the adoption of cross-validation using a leave-p-out approach ensures unbiased evaluation and robust assessment of the algorithms' performance across multiple datasets. These factors contribute to the validity of our experiments and allow for replication and verification by the research community.

5. Results and analysis

This section reports on the experimental process and results in detail. It discusses data preprocessing and the removal of datasets that could not be detected by the optimized algorithms. Then it explains cross-validation to obtain input parameters. The quality, efficiency, and stability of the test results using different optimization algorithms applied to the onsets/offsets scenario is also discussed. This section also discusses the pros and cons of each selected algorithm.

5.1. Data preprocessing

The preliminary analysis shows that the detection results from the selected optimization algorithms are significantly different from expert identification when the number of muscle activities is $\leqslant 21$. This results in low Accuracy and F1-Score values ($< 60\%$). Fig. 5 shows the detection results using the PSO algorithm on $\mathbb{C}1$, which also have low Accuracy and F1-Score for datasets $\mathbb{D}7$, $\mathbb{D}11$, $\mathbb{D}13$, and $\mathbb{D}18$ when the number of muscle activities is $\leqslant 21$ as shown in Fig. 5.

Fig. 6 shows the detection results using the Ant Colony Optimization (ACO) algorithm on $\mathbb{C}2$. Similar to the PSO algorithm, when the number of muscle activities is $\leqslant 21$, the results are poor. This pattern is also seen in the detection results from the other three optimization algorithms, Simulated Annealing (SA), GA, and TS.

Based on the above results, it can be concluded that none of the selected optimization algorithms are effective for onsets/offsets detection when the number of muscle activities is $\leqslant 21$. Thus, the datasets $\mathbb{D}7$, $\mathbb{D}11$, $\mathbb{D}13$, $\mathbb{D}17$, and $\mathbb{D}18$ are excluded from further comparative analysis.

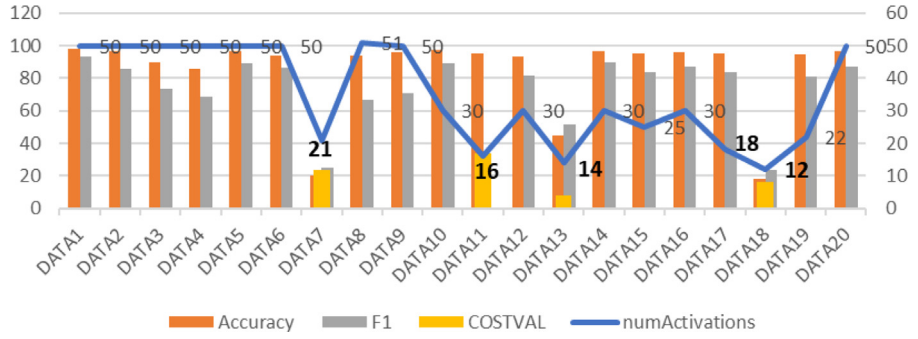

Fig. 5. Detection quality using PSO algorithm.

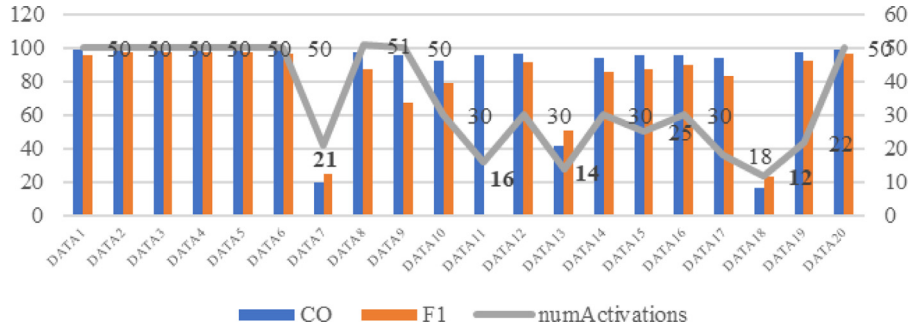

Fig. 6. Detection quality using ASO algorithm.

Table 8
Detection parameters obtained by the optimization algorithms.

Algorithm	Parameters								Cost value
	L_b	K_{bth}	N_{sd}	T_{on}	T_{off}	Ts	N_{nt}	T_j	
PSO	0.7460	3.0000	40.0000	0.0100	0.1400	0.5940	5.0000	0.6540	0.0086
ACO	0.5660	2.0000	31.0000	0.0120	0.1460	0.3400	5.0000	2.8080	0.0081
SA	0.9500	1.0000	47.0000	0.0100	0.0680	0.1980	3.0000	0.0240	0.0104
GA	0.4580	2.0000	16.0000	0.0120	0.1360	0.4160	5.0000	2.6080	0.0085
TS	0.3620	2.0000	70.0000	0.0120	0.1420	0.4100	6.0000	0.6800	0.0085

5.2. Detection parameters

Cross-validation was used to obtain the detection parameters, as described in Section 4.5. The training set consists of 5 datasets, which are used to obtain the parameters using the optimization algorithms. These parameters are then used to detect the remaining four training datasets. The $\mathbb{C}2$ value is calculated for each detection result by comparing it to the expert-identified result.

Fig. 7 shows the results of the optimization algorithms in obtaining parameters via cross-validation. The black line represents the cost function value from each training dataset, and the red line is the average cost function value of the remaining training datasets. The detection parameter vector with the lowest mean value is chosen as the optimal vector. The PSO, GA, and Tabu Search (TS) algorithms obtain the optimization parameters from detecting $\mathbb{D}3$, while the Ant Colony Optimization (ACO) and Simulated Annealing (SA) algorithms obtain the parameters from detecting $\mathbb{D}2$.

The optimal parameter vectors obtained by each optimization algorithm are recorded in Table 8. The cost values corresponding to these optimal parameters are between 0.0081 and 0.0104. The parameters obtained by each algorithm are not similar, so they have no interpretable meaning.

5.3. Quality measurement

This section explains the quality matrices of selected optimization algorithms i.e., Accuracy, F1-Score, Degree Under Detection (UD), Degree Over Detection (OD), Detection Rate, and Area Under the Curve (AUC).

5.3.1. Accuracy, F1-score, degree Under Detection (UD), degree Over Detection (OD), detection rate

Table 9 shows the quality metrics of the discussed optimization algorithms, including Accuracy, F1-Score, UD, OD, and Detection Rate. The median, maximum, minimum, and interquartile range (IQR) values are displayed for each metric.

Each optimization algorithm utilized $\mathbb{C}1$ and $\mathbb{C}2$ as optimization targets to compute each metric. The comparison among the optimization algorithms is based on the assumption that the cost function is identical.

The comparison of the results of the optimization algorithms that used $\mathbb{C}1$ as the optimization objective is shown in Fig. 8. Although there was a slight variation in the maximum, minimum, and median values of Accuracy, F1-Score, OD, and UD among the chosen algorithms, the Simulated Annealing (SA) algorithm performed better, with higher Accuracy and F1-Score and lower Degree Under Detection (UD) and Degree Over Detection (OD) values.

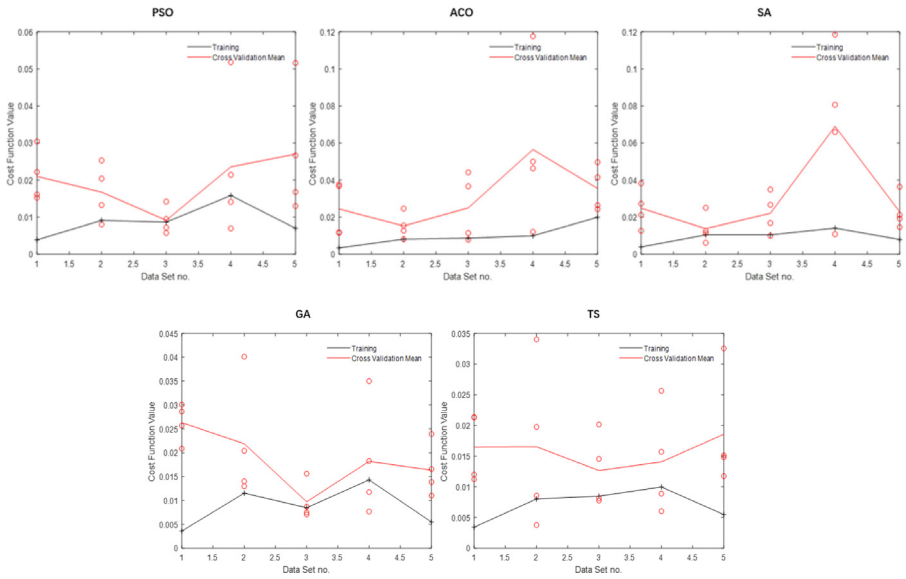

Fig. 7. Obtaining input parameters for detector using cross-validation.

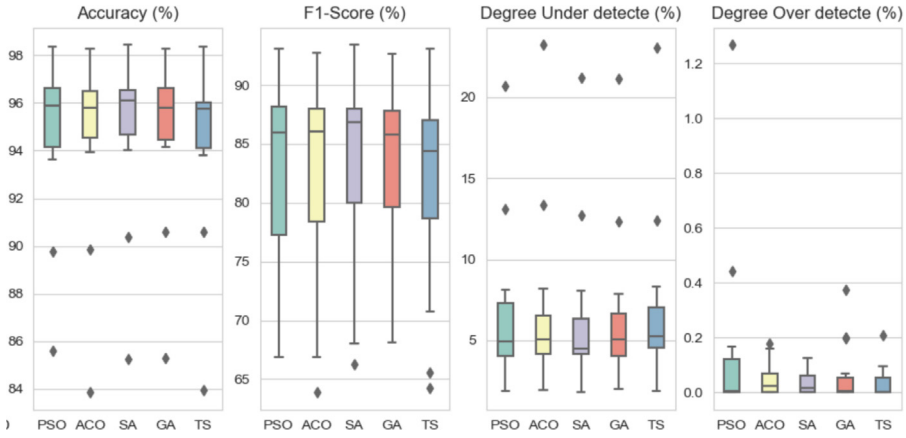

Fig. 8. Comparison of optimization detection results of $\mathbb{C}1$.

Table 9
Quality measurement of the optimization algorithms involved in the detection process.

Measure		Accuracy		F1-Score		UD		OD		Detection rate	
Cost function		$\mathbb{C}1$	$\mathbb{C}2$	$\mathbb{C}1$	$\mathbb{C}2$	$\mathbb{C}1$	$\mathbb{C}2$	$\mathbb{C}1$	$\mathbb{C}2$	$\mathbb{C}1$	$\mathbb{C}2$
PSO	Median	95.91	98.33	85.99	95.67	4.96	0.38	0	5.01	100	100
	IQR	2.51	1.46	10.88	3.54	3.31	0.68	0.12	6.55	0	3
	Max	98.36	99.36	93.08	97.97	20.67	9.27	1.27	28.4	100	108
	Min	85.62	91.42	66.92	7.51	1.86	0.05	0	0.53	100	4
ACO	Median	95.78	97.24	86.02	90.21	5.07	0.34	0.02	12.98	100	98
	IQR	1.97	3.33	9.57	9.06	2.36	0.7	0.07	19.4	0	4
	Max	98.27	99.29	92.7	97.79	23.22	2.38	0.18	43.93	100	100
	Min	83.85	93.3	63.86	81.86	1.95	0.02	0	1.2	100	91
SA	Median	96.1	96.74	86.81	88.63	4.48	0.23	0.02	21.45	100	120
	IQR	1.85	3.28	8.01	8.98	2.2	0.67	0.06	19.4	0	38
	Max	98.43	99.26	93.41	97.77	21.2	2.02	0.13	55.8	100	163
	Min	85.25	92.11	66.26	78.11	1.78	0.001	0	3.04	100	100
GA	Median	95.78	96.8	85.75	91.39	5.07	0.31	0.01	12.99	100	98
	IQR	2.19	2.79	8.13	8.25	2.6	0.67	0.05	16.2	0	4
	Max	98.25	99.34	92.63	97.95	21.14	2.31	0.37	41.31	100	103
	Min	85.29	94.01	68.13	79.17	1.98	0.02	0	1.3	100	95
TS	Median	95.75	97.22	84.4	89.14	5.23	0.37	0	13.07	100	100
	IQR	1.9	2.41	8.35	6.72	2.49	0.73	0.06	14.61	0	3
	Max	98.36	99.34	93.07	97.95	23.05	2.13	0.21	37.24	100	107
	Min	83.96	94.7	64.2	81.54	1.87	0.02	0	1.3	100	92

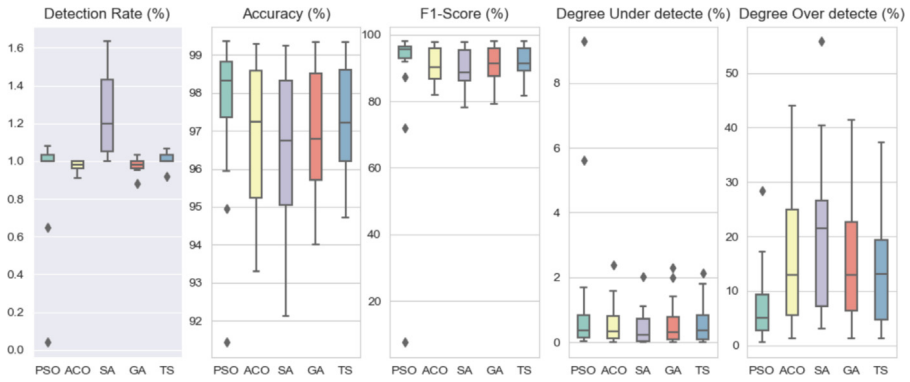

Fig. 9. Comparison of optimization detection results of $\mathbb{C}2$.

Table 10
The AUC measurement result.

Cost function	Method	PSO	ACO	SA	GA	TS
$\mathbb{C}1$	Average	85.39	85.48	86.02	85.86	85.01
	Median	87.71	87.78	88.35	87.53	86.51
$\mathbb{C}2$	Average	93.82	96.40	96.64	96.51	96.79
	Median	98.28	96.58	96.92	97.56	97.52

Table 11
F1-Score value corresponding to the cost value mean.

Cost Value	F1-Score
$\leqslant$0.1169 (Max)	>90%
$\leqslant$0.1203 (Median)	>80%
$\leqslant$0.1384 (Max)	>80%

In summary, the results of the detection process using $\mathbb{C}1$ indicate that all the optimization algorithms have a 100% detection rate, whereas, Simulated Annealing (SA) outperforms the others with higher Accuracy and F1-Score, and lower UD and OD. However, the PSO algorithm has a higher interquartile range (IQR) value and a higher maximum value of OD, suggesting that its results are less stable compared to the other algorithms.

Fig. 9 compares the optimized detection results using $\mathbb{C}2$. It can be seen that the Simulated Annealing (SA) algorithm detected more onsets/offsets than the actual number and had a wider range of detection results compared to the other optimization algorithms. Simulated Annealing (SA) also had a lower median accuracy, F1-Score, and Degree Over Detection (OD) results compared to its results when using $\mathbb{C}1$.

The comparison between optimization algorithms when using $\mathbb{C}2$ shows that PSO has better Accuracy, F1-Score, UD, and Degree Over Detection (OD) but with more outliers than the other algorithms, suggesting instability in its detection results. The Simulated Annealing (SA) algorithm produced more onsets/offsets and had a wider range of detection results, with poorer median values for Accuracy, F1-Score, and Degree Over Detection (OD) compared to other algorithms.

5.3.2. Area Under the Curve (AUC)

Table 10 shows the comparison of the optimization algorithms in terms of their AUC scores, with results showing that each algorithm performed better with $\mathbb{C}2$ compared to $\mathbb{C}1$.

When comparing the performance of the optimization algorithms using $\mathbb{C}2$, it is observed that the median AUC value of the PSO algorithm is higher than that of the other algorithms, although its average value is lower. This indicates that the results obtained by the PSO algorithm have more variability than those obtained by the other algorithms.

5.4. Summary of quality measurement results

In conclusion, the results of the comparison between the optimization algorithms suggest that when the same cost function is used, the results of detection can be similar with respect to the median and maximum values of Accuracy and F1-Score. However, the stability of the results varies between algorithms. The PSO algorithm showed a higher interquartile range (IQR) value when using $\mathbb{C}1$ and more outlier values when using $\mathbb{C}2$. This is reflected in the lower mean AUC value for PSO compared to the other algorithms. These results indicate that the PSO algorithm is more random and variable than the other algorithms.

When $\mathbb{C}1$ is used as the cost function, the Simulated Annealing (SA) algorithm shows better performance in terms of Accuracy, F1-Score, Degree Over Detection (OD), UD, and AUC compared to other optimization algorithms, but with slight differences. On the other hand, when $\mathbb{C}2$ is used, the Simulated Annealing (SA) algorithm detects more onsets/offsets than the actual amount and returns a higher detection rate. On the other hand, the other algorithms' results match the exact number. PSO algorithm is more random and variable in terms of AUC, with a higher median but lower mean value compared to other algorithms.

5.5. Runtime & success rate for efficiency and reliability measurements

The impact of cost function value on test results was demonstrated in the experiment. The optimal cost value was found to be between 0.0081 and 0.0104 (per Section 4.2). To compare the efficiency and stability of different optimization algorithms, a cost function threshold was established, and a successful process was defined as one with a cost function value less than or equal to the threshold. The runtime of the optimization algorithm was recorded in this scenario.

The relationship between F1-Score and cost value is shown in Table 11 for 5 datasets, $\mathbb{D}1$, $\mathbb{D}2$, $\mathbb{D}3$, $\mathbb{D}4$, and $\mathbb{D}5$, using $\mathbb{C}1$. F1-Score is above 90% when the cost value is 0.1169 or lower and above 80% when the cost value is 0.1384 or lower. The median cost value for F1-Score above 80% is 0.1203, and 0.1169, 0.1203, and 0.1384 are selected as the cost value threshold.

The optimization process using $\mathbb{C}1$ stops when the cost value is less than or equal to the threshold. This value is considered the input parameter for the detection algorithm and is deemed a successful process. If the optimization process does not reach the cost value threshold by the time the termination condition is met, it is considered a failure.

In each of the ten runs, the optimization algorithms were applied with the same cost value threshold and datasets $\mathbb{D}1$, $\mathbb{D}2$, $\mathbb{D}3$, $\mathbb{D}4$, and $\mathbb{D}5$ were used. The results of the optimization process, including the median Accuracy and F1-Score of the detection result and the Runtime, are presented in Table 12.

The results demonstrate that for the same dataset, a smaller cost value leads to higher Accuracy. For $\mathbb{D}1$, the best results in Accuracy and F1-Score are obtained when the cost value falls between 0.1203 and 0.1384. A comparison of the runtime of the selected optimization

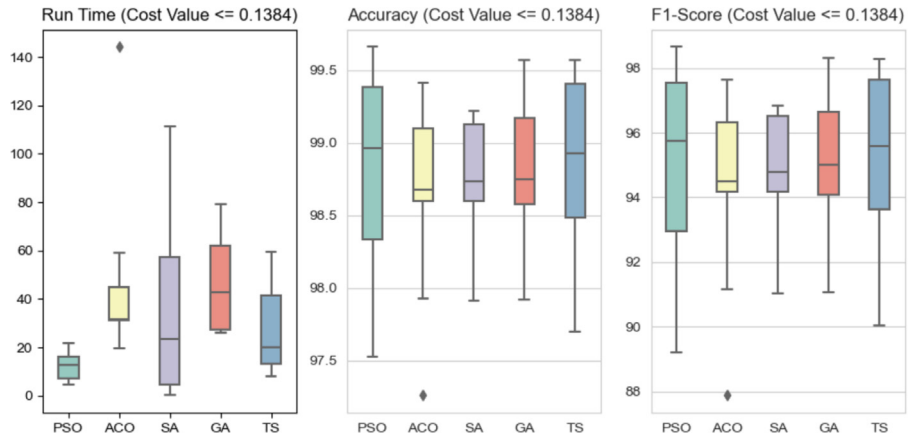

Fig. 10. Distribution comparison of runtime, accuracy, and F1-Score values of the optimization algorithms under the condition of cost value $\leqslant 0.1384$.

Table 12
Measurement of runtime and success rate under different cost value thresholds.

Cost value	Algorithm	Accuracy (%)	F1-Score (%)	Runtime (s)	Success rate
$\leqslant 0.150$	PSO	98.61	94.26	6.85	100%
	ACO	98.15	92.19	32.15	100%
	SA	98.66	94.78	9.96	100%
	GA	98.91	95.56	63.38	100%
	TS	98.35	93.11	12.45	100%
$\leqslant 0.1384$	PSO	98.96	95.74	12.8	100%
	ACO	98.67	94.5	31.39	100%
	SA	98.73	94.78	23.43	100%
	GA	98.75	95	42.55	100%
	TS	98.92	95.6	19.91	100%
$\leqslant 0.1203$	PSO	98.9	95.47	22.48	100%
	ACO	98.66	94.43	35.24	100%
	SA	98.84	95.23	33.46	100%
	GA	98.71	94.64	51.8	100%
	TS	98.65	94.38	141.73	100%
$\leqslant 0.1169$	PSO	98.56	93.99	52.3	80%
	ACO	98.38	93.18	60.87	80%
	SA	98.4	93.57	333.31	40%
	GA	98.44	93.45	214.02	80%
	TS	98.48	93.64	2147.6	90%

Table 13
Summary of comparisons.

Measurements		$\mathbb{C}1$					$\mathbb{C}2$				
		PSO	ACO	SA	GA	TS	PSO	ACO	SA	GA	TS
Quality	Accuracy			↑			↑		↓		
	F1-Score			↑			↑		↓		
	UD and OD			↑			↑		↓		
	Detection rate	↑	↑	↑	↑	↑			↓		
	AUC						↑	↑	↑	↑	↑
Efficiency	Runtime	↑		↓			↑		↓		
Stability	Success rate								↓		
	Stability (IQR)	↓	↑	↓	↑	↓	↓	↑	↓	↑	↓

algorithms in this cost value range is crucial to determine the best algorithm for onsets/offsets detection. Additionally, Table 12 shows that the PSO algorithm has the best median values of Accuracy, F1-Score, and Runtime when the cost value is less than 0.1384.

From the boxplots in Fig. 10, it is evident that PSO has the best median value for Runtime and a more compact distribution range compared to the other algorithms. On the other hand, PSO's Accuracy and F1-Score values exhibit more variability as compared to the other four algorithms, which aligns with the previously derived conclusion from the Quality comparison.

When the cost value threshold was further reduced to 0.1384, the Success Rate of all algorithms decreased further. The PSO algorithm had the highest Success Rate of 80%, while the other algorithms had a Success Rate ranging from 50% to 60%. This comparison clearly shows that PSO had the best performance in terms of Success Rate and that the Success Rate of the other algorithms was significantly lower. The results indicate that PSO is the best algorithm in terms of reliability in the onset/offset detection scenario when the cost value threshold is set to 0.1384.

In conclusion, the experimental results demonstrate that the PSO algorithm achieves the best median values for both Runtime and efficiency in terms of Accuracy and F1-Score. Table 12 shows that the PSO algorithm consistently outperforms other algorithms in terms of Runtime when the cost value threshold is set to 0.1384. Moreover, Fig. 10 confirms that the PSO algorithm exhibits a more compact distribution range for Runtime compared to the other algorithms. These findings emphasize the efficiency of the PSO algorithm, highlighting its potential for efficient onset/offset detection.

6. Discussion

This section discusses the performance of the selected optimization algorithms in terms of quality, efficiency, and stability metrics.

6.1. Quality, efficiency, and reliability matrices

This study evaluated the performance of the optimization algorithms discussed in this study based on quality, efficiency, and stability metrics. The results were summarized in Table 13 where ↓ represents lower performance compared to others, and ↑ indicates higher performance in comparison.

Firstly, the quality of detection results was the focus of this study. When the running time was not taken into account, results showed that similar quality was obtained from each optimization algorithm using $\mathbb{C}1$. Differences between the median index and interquartile range (IQR) values were small. On the other hand, when $\mathbb{C}2$ was used, the PSO algorithm produced better quality results with a median value better than the other algorithms. On the other hand, the Simulated Annealing (SA) algorithm performed relatively poorly.

Secondly, the efficiency of the optimization algorithms was also evaluated. Specifically, the runtime was used to measure the efficiency. The results showed that the PSO algorithm was the fastest among the chosen optimization algorithms. However, the Simulated Annealing (SA) algorithm had the slowest runtime and was not as efficient as the other algorithms.

Finally, The GA and Ant Colony Optimization (ACO) algorithms are more stable than PSO, Tabu Search (TS), and Simulated Annealing (SA). The PSO and TS algorithms have higher interquartile range

(IQR) values for certain measurement indices, indicating lower stability compared to the GA and Ant Colony Optimization (ACO) algorithms. Meanwhile, Simulated Annealing (SA)'s instability is reflected in its higher interquartile range (IQR) value of runtime after setting a cost value threshold.

In addition to the above, the following conclusions can be made. The experiments found that the performance of optimization algorithms may vary depending on factors such as the number of muscle activities, the optimal parameters obtained, and the design of the cost function. When the number of muscle activities is less than 21, the accuracy of the results is unacceptable. Additionally, the optimal parameters obtained by different optimization algorithms may vary greatly, even with the same cost function value, potentially losing their business meaning. The design of the cost function has a significant impact on the detection results, as the same cost function leads to similar quality results across algorithms, but different cost functions result in different measures such as Accuracy, F1-Score, Degree Over Detection (OD), and Degree Under Detection (UD).

The cost function value plays a role in determining the quality of the detection results, but a lower cost value does not necessarily indicate better quality. Based on the study results, the optimal detection results are obtained when the cost value falls within the range of 0.1203 to 0.1384. Thus, it is crucial to experimentally determine a reasonable cost value when using optimization algorithms in practice.

The optimization algorithms implemented in this work for onset/offset detection (PSO, Ant Colony Optimization (ACO), Simulated Annealing (SA), GA, and Tabu Search (TS)) can also be effectively applied to real-time scenarios where timely detection of the beginning or end of events is crucial. For example, in environmental monitoring, the algorithms can analyze real-time sensor data to detect the onset of pollutant levels exceeding acceptable thresholds or the offset of abnormal environmental conditions. In industrial settings, the algorithms can process continuous data streams from instruments such as oscilloscopes or spectrometers to identify the start and end of specific process states or anomalies.

Furthermore, the algorithms can be employed in the Internet of Things (IoT) devices, where they can analyze real-time sensor data from smart home devices, wearables, or industrial IoT devices, enabling the detection of events such as motion patterns, abnormal energy consumption, or equipment failure. In healthcare, the algorithms can leverage real-time data from medical devices like heart rate monitors or EEG devices to detect the onset or offset of critical health events or abnormal patterns. Additionally, in imaging applications, the algorithms can process real-time image or video streams captured by surveillance cameras, traffic monitoring systems, or medical imaging devices, facilitating the detection of objects, anomalies, or changes in visual data.

Some of the limitations of these optimization algorithms are as follows. Particle Swarm Optimization (PSO) may suffer from premature convergence and sensitivity to noise or outliers in the data. Ant Colony Optimization (ACO) can have high computational complexity and may require a large number of iterations to converge. Simulated Annealing (SA) is susceptible to getting trapped in local optima, and its efficiency depends on the choice of the cooling schedule. Genetic Algorithms (GA) can become computationally expensive, especially with large populations or high-dimensional search spaces, and may suffer from premature convergence. Tabu Search (TS) is sensitive to the tabu length parameter and can have a higher computational complexity.

7. Conclusions and future work

Optimization algorithms are valuable for onsets/offsets detection in Surface Electromyography (sEMG) signals, as they improve efficiency and reduce dependence on manual parameter adjustment while still obtaining high-quality results. This paper performed a comparative analysis of the performance of the meta-heuristic global optimization algorithms, Particle Swarm Optimization (PSO), Genetic algorithms (GA), Simulated Annealing (SA), Ant Colony Optimization (ACO), and Tabu Search (TS) for the onsets/offsets detection process in sEMG signals, yielding a median Accuracy of over 95%. These algorithms reduced the dependence on manual parameter adjustment and improved detection efficiency. The Particle Swarm Optimization algorithm had the best median Accuracy and F1-Score compared to the other algorithms. It also showed efficient runtime. However, its results were less stable, with a wider range of interquartile range (IQR) values compared to the Genetic algorithms (GA) and Ant Colony Optimization (ACO) algorithms. This paper provides insights for choosing an algorithm, the trade-off between quality, efficiency, and stability. Our work primarily emphasizes the utilization of individual algorithms rather than their hybridization. However, it is worth exploring the possibility of incorporating additional optimization algorithms and hybrid optimization algorithms in future research to enhance the obtained outcomes.

Declaration of competing interest

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

Data availability

No data was used for the research described in the article

Funding

No specific funding was obtained for this work.

References

[1] A. Merlo, I. Campanini, Technical aspects of surface electromyography for clinicians, Open Rehabil. J. 3 (2010) 1.
[2] G. Staude, C. Flachenecker, M. Daumer, W. Wolf, Onset detection in surface electromyographic signals: a systematic comparison of methods, EURASIP J. Adv. Signal Process. 2001 (2) (2001) 867853.
[3] F. Karimi, J. Kofman, N. Mrachacz-Kersting, D. Farina, N. Jiang, Detection of movement related cortical potentials from EEG using constrained ICA for brain-computer interface applications, Front. neurosci. 11 (2017) 356, http://dx.doi.org/10.3389/fnins.2017.00356.
[4] M. Magda, A. Martinez-Alvarez, S. Cuenca-Asensi, MOOGA parameter optimisation for onset detection in EMG signals, in: International Conference on Image Analysis and Processing, . , Cham, 2017, pp. 171–180.
[5] U. Rashid, I.K. Niazi, N. Signal, D. Farina, D. Taylor, Optimal automatic detection of muscle activation intervals, J. Electromyogr. Kinesiol. 48 (2019) 103–111, http://dx.doi.org/10.1016/j.jelekin.2019.06.010.
[6] P.W. Hodges, B.H. Bui, A comparison of computer-based methods for the determination of the onset of muscle contraction using electromyography, Electroencephalogr. Clin. Neurophysiol./Electromyogr. Mot. Control 101 (6) (1996) 511–519.
[7] P. Bonato, T. D'Alessio, M. Knaflitz, A statistical method for the measurement of muscle activation intervals from surface myoelectric signal during gait, IEEE Trans. biomed. eng. 45 (3) (1998) 287–299.
[8] G. Severini, S. Conforto, M. Schmid, T. D'alessio, Novel formulation of a double threshold algorithm for the estimation of muscle activation intervals designed for variable SNR environments, J. Electromyogr. Kinesiol. 22 (6) (2012) 878–885.
[9] S. Micera, A.M. Sabatini, P. Dario, An algorithm for detecting the onset of muscle contraction by EMG signal processing, Med. Eng. Phys. 20 (3) (1998) 211–215.
[10] S.E. Selvan, D. Allexandre, U. Amato, G.H. Yue, Unsupervised stochastic strategies for robust detection of muscle activation onsets in surface electromyogram, IEEE Trans. Neural Syst. Rehabil. Eng. 26 (6) (2018) 1279–1291.
[11] H. Salimi, Stochastic fractal search: a powerful metaheuristic algorithm, Knowl.-Based Syst. 75 (2015) 1–18.
[12] X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2010.
[13] T. Moritani, D. Stegeman, R. Merletti, Basic physiology and biophysics of EMG signal generation, Electromyogr. Physiol. Eng. Noninvasive Appl. (2004) 1–20.
[14] P. Ahmadian, S. Cagnoni, L. Ascari, How capable is non-invasive EEG data of predicting the next movement? a mini review, Front. hum. neurosci. 7 (2013) 124.
[15] H. Huang, T.A. Kuiken, R.D. Lipschutz, A strategy for identifying locomotion modes using surface electromyography, IEEE Trans. Biomed. Eng. 56 (1) (2008) 65–73.

[16] K. Peter, ABC of EMG, a practical introduction to kinesiological electromyography, 2005.
[17] H.K. Hameed, W.Z. Hasan, S. Shafie, S.A. Ahmad, H. Jaafar, An amplitude independent muscle activity detection algorithm based on adaptive zero crossing technique and mean instantaneous frequency of the semg signal, in: IEEE Regional Symposium on Micro and Nanoelectronics, RSM, IEEE, 2017, pp. 183–186.
[18] S. Restrepo-Agudelo, S. Roldan-Vasco, L. Ramirez-Arbelaez, S. Cadavid-Arboleda, E. Pe-rez Giraldo, A. Orozco-Duque, Improving surface EMG burst detection in infrahyoid muscles during swallowing using digital filters and discrete wavelet analysis, J. Electromyogr. Kinesiol. 35 (2017) 1–8.
[19] M. Dorigo, T. St'utzle, Ant colony optimisation: overview and recent advances, in: Handbook of metaheuristics, 2019, pp. 311–351, http://dx.doi.org/10.1007/978-3-319-91086-4_10.
[20] L.M. Rios, N.V. Sahinidis, Derivative-free optimisation: a review of algorithms and comparison of software implementations, J. Glob. Optimisation 56 (3) (2013) 1247–1293.
[21] W. Elloumi, El Abed, H. Abraham, A.M.A. Alimi, A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP, Appl. Soft Comput. 25 (2014) 234–241.
[22] G. Xu, H. Zhao, Optimisation Method and Its MATLAB Implementation, Vol. 978, Beihang University Press, Beijing, 2018.
[23] X.S. Yang, Optimisation and metaheuristic algorithms in engineering, Metaheuristics water, geotech. transp. eng. (2013) 1–23.
[24] M. Gilli, an Introduction To Optimisation Heuristics, In Seminar university of cyprus department of public and business administration, 2004.
[25] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. oper. res. 13 (5) (1986) 533–549.
[26] X.S. Yang, Metaheuristic optimisation, Scholarpedia 6 (8) (2011) 11472.
[27] C. Blum, A. Roli, Metaheuristics in combinatorial optimisation: Overview and conceptual comparison, ACM comput. surv. (CSUR) 35 (3) (2003) 268–308.
[28] M. Sedghi, A. Ahmadian, M. Aliakbar-Golkar, Assessment of optimisation algorithms capability in distribution network planning: Review, comparison, and modification techniques, Renew. Sustain. Energy Rev. 66 (2016) 415–434.
[29] Z. Beheshti, S.M.H. Shamsuddin, A review of population-based meta-heuristic algorithms, Int. J. Adv. Soft Comput. Appl. 5 (1) (2013) 1–35.
[30] Zedong Zheng, Shengxiang Yang, Yinan Guo, Xiaolong Jin, Rui Wang, Meta-heuristic techniques in microgrid management: A survey, Swarm Evol. Comput. (2023) 101256, http://dx.doi.org/10.1016/j.swevo.2023.101256.
[31] Zhongqiang Ma, Guohua Wu, Ponnuthurai N Suganthan, Aijuan Song, Qizhang Luo, Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms, Swarm Evol. Comput. (2023) 101248.
[32] R.P. Di Fabio, Reliability of computerised surface electromyography for determining the onset of muscle activity, Phys. Ther. 67 (1) (1987) 43–48.
[33] S. Solnik, P. Rider, K. Steinweg, P. DeVita, T. Hortobágyi, Teager–Kaiser energy operator signal conditioning improves EMG onset detection, Eur. J. Appl. Physiol. 110 (3) (2010) 489–498.
[34] J.F. Kaiser, Some useful properties of teager's energy operators, in: International Conference on Acoustics, signal processing, 3, IEEE speech, 1993, pp. 149–152, http://dx.doi.org/10.1109/ICASSP.1993.319457.
[35] X. Li, P. Zhou, A.S. Aruin, Teager–kaiser energy operation of surface EMG improves muscle activity onset detection, Ann. Biomed. Eng. 35 (9) (2007) 1532–1538.
[36] R. Eberhart, J. Kennedy, A new optimiser using particle swarm theory, in: International Symposium on Micro Machine and Human Science, IEEE, 1995, pp. 39–43.
[37] Neelam Singh, Yasir Hamid, Sapna Juneja, Gautam Srivastava, Gaurav Dhiman, Thippa Reddy Gadekallu, Mohd Asif Shah, Load balancing and service discovery using docker swarm for microservice based big data applications, J. Cloud Comput. 12 (1) (2023) http://dx.doi.org/10.1186/s13677-022-00358-7.
[38] Shivali Slathia, Rakesh Kumar, Mudassir Lone, Wattana Viriyasitavat, Amandeep Kaur, Gaurav Dhiman, A performance evaluation of situational-based fuzzy linear programming problem for job assessment, in: Lecture Notes in Networks and Systems, Springer Nature Singapore, 2023, pp. 655–667, http://dx.doi.org/10.1007/978-981-19-9228-5_56.
[39] Pooja, Rakesh Kumar, Wattana Viriyasitavat, Kusum Yadav, Gaurav Dhiman, Analysis of clustering algorithms for facility location allocation problems, in: Proceedings of Third International Conference on Advances in Computer Engineering and Communication Systems: ICACECS 2022, Springer, 2023, pp. 597–605.
[40] Wongchai Anupong, Abolfazl Mehbodniya, Julian L. Webber, Ali Bostani, Gaurav Dhiman, Bharat Singh, Murali Dharan A.R., Deep learning algorithms were used to generate photovoltaic renewable energy in saline water analysis via an oxidation process, J. Water Reuse and Desalination (2023) http://dx.doi.org/10.2166/wrd.2023.071.
[41] M.S. Mekala, Gaurav Dhiman, Wattana Viriyasitavat, Ju H. Park, Ho-Youl Jung, Efficient lidar-trajectory affinity model for autonomous vehicle orchestration, IEEE Trans. Intell. Transp. Syst. (2023) 1–11, http://dx.doi.org/10.1109/tits.2023.3242900.
[42] Piyush Vyas, Gitika Vyas, Gaurav Dhiman, Ruemo—The classification framework for Russia-Ukraine war-related societal emotions on Twitter through machine learning, Algorithms 16 (2) (2023) 69, http://dx.doi.org/10.3390/a16020069.
[43] Subhranshu Sekhar Tripathy, Kaushik Mishra, Diptendu Sinha Roy, Kusum Yadav, Ali Alferaidi, Wattana Viriyasitavat, J. Sharmila, Gaurav Dhiman, Rabindra K. Barik, State-of-the-art load balancing algorithms for mist-fog-cloud assisted paradigm: A review and future directions, Arch. Comput. Methods Eng. 30 (4) (2023) 2725–2760, http://dx.doi.org/10.1007/s11831-023-09885-1.
[44] Vimal Shanmuganathan, Victor Hugo C. de Albuquerque, Paulo C.S. Barbosa, Marcello Carvalho dos Reis, Gaurav Dhiman, Mohd Asif Shah, Software based sentiment analysis of clinical data for healthcare sector, IET Softw. (2023) http://dx.doi.org/10.1049/sfw2.12115.
[45] Gaurav Dhiman, Marcello Carvalho dos Reis, Paulo C.S. Barbosa, Victor Hugo C. de Albuquerque, Sandeep Kautish, Blockchain-based covert software information transmission for bitcoin, IET Softw. (2023) http://dx.doi.org/10.1049/sfw2.12120.
[46] Janmenjoy Nayak, H. Swapnarekha, Bighnaraj Naik, Gaurav Dhiman, S. Vimal, 25 Years of particle swarm optimization: Flourishing voyage of two decades, Arch. Comput. Methods Eng. 30 (3) (2022) 1663–1725, http://dx.doi.org/10.1007/s11831-022-09849-x.
[47] S Zhen, R. Surender, Gaurav Dhiman, K. Radha Rani, K.M. Ashifa, Faheem Ahmad Reegu, Intelligent-based ensemble deep learning model for security improvement in real-time wireless communication, Optik 271 (2022) 170123, http://dx.doi.org/10.1016/j.ijleo.2022.170123.
[48] Gaurav Dhiman, Vijay Kumar, Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications, Adv. Eng. Softw. 114 (2017) 48–70, http://dx.doi.org/10.1016/j.advengsoft.2017.05.014.
[49] Gaurav Dhiman, Vijay Kumar, Emperor penguin optimizer: A bio-inspired algorithm for engineering problems, Knowl.-Based Syst. 159 (2018) 20–50, http://dx.doi.org/10.1016/j.knosys.2018.06.001.
[50] Satnam Kaur, Lalit K. Awasthi, A.L. Sangal, Gaurav Dhiman, Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization, Eng. Appl. Artif. Intell. 90 (2020) 103541, http://dx.doi.org/10.1016/j.engappai.2020.103541.
[51] Gaurav Dhiman, Amandeep Kaur, STOA: A bio-inspired based optimization algorithm for industrial engineering problems, Eng. Appl. Artif. Intell. 82 (2019) 148–174, http://dx.doi.org/10.1016/j.engappai.2019.03.021.
[52] Rakesh Kumar, Gaurav Dhiman, A comparative study of fuzzy optimization through fuzzy number, Int. J. Mod. Res. 1 (1) (2021) 1–14.
[53] Ishita Chatterjee, Artificial intelligence and patentability: Review and discussions, Int. J. Mod. Res. 1 (1) (2021) 15–21.
[54] Pankaj Kumar Vaishnav, Sunil Sharma, Prashant Sharma, Analytical review analysis for screening COVID-19, Int. J. Mod. Res. 1 (1) (2021) 22–29.
[55] Vishan Kumar Gupta, Surendra Kumar Shukla, Ramesh Singh Rawat, et al., Crime tracking system and people's safety in India using machine learning approaches, Int. J. Mod. Res. 2 (1) (2022) 1–7.
[56] Tripti Sharma, Rajit Nair, S. Gomathi, Breast cancer image classification using transfer learning and convolutional neural network, Int. J. Mod. Res. 2 (1) (2022) 8–16.
[57] Surendra Kumar Shukla, Vishan Kumar Gupta, Kireet Joshi, Ankit Gupta, Mukesh Kumar Singh, Self-aware execution environment model (SAE2) for the performance improvement of multicore systems, Int. J. Mod. Res. 2 (1) (2022) 17–27.
[58] Gaurav Dhiman, Vijay Kumar, Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems, Knowl.-Based Syst. 165 (2019) 169–196, http://dx.doi.org/10.1016/j.knosys.2018.11.024.
[59] Gaurav Dhiman, Meenakshi Garg, Atulya Nagar, Vijay Kumar, Mohammad Dehghani, A novel algorithm for global optimization: Rat swarm optimizer, J. Ambient Intell. Humaniz. Comput. 12 (8) (2020) 8457–8482, http://dx.doi.org/10.1007/s12652-020-02580-0.
[60] Gaurav Dhiman, Diego Oliva, Amandeep Kaur, Krishna Kant Singh, S. Vimal, Ashutosh Sharma, Korhan Cengiz, BEPO: A novel binary emperor penguin optimizer for automatic feature selection, Knowl.-Based Syst. 211 (2021) 106560, http://dx.doi.org/10.1016/j.knosys.2020.106560.
[61] Gaurav Dhiman, ESA: A hybrid bio-inspired metaheuristic optimization approach for engineering problems, Eng. Comput. 37 (1) (2019) 323–353, http://dx.doi.org/10.1007/s00366-019-00826-w.
[62] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimisation by simulated annealing, Science 220 (4598) (1983) 671–680.
[63] H. Wang, X. Zhang, J. Chen, Y. Wang, Realisation of human-computer interaction of lower limbs rehabilitation robot based on sEMG, in: IEEE International Conference on Cyber Technology in Automation, Control and Intelligent, 2014, pp. 491–495, http://dx.doi.org/10.1109/CYBER.2014.6917513.
[64] Marco Dorigo, Gianni Di Caro, Ant colony optimization: a new meta-heuristic, in: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), 2, IEEE, 1999, pp. 1470–1477.
[65] David Martens, Bart Baesens, Tom Fawcett, Editorial survey: swarm intelligence for data mining, Mach. Learn. 82 (1) (2011) 1–42.
[66] A. Naseem, M. Jabloun, O. Buttelli, P. Ravier, Detection of sEMG muscle activation intervals using gaussian mixture model and ant colony classifier, Eur. Signal Process. Conf. (2016) 1713–1717.
[67] H. Huang, H.B. Xie, J.Y. Guo, H.J. Chen, Ant colony optimisation-based feature selection method for surface electromyography signals classification, Comput. Biol. Med. 42 (1) (2012) 30–38.

[68] A. Sebastian, M.P. Schoen, Hybrid Particle Swarm: Tabu Search Optimisation Algorithm for Parameter Estimation, Dynamic Systems and Control Conference. American Society of Mechanical Engineers Digital Collection, 2014.
[69] S. Iadevaia, Y. Lu, F.C. Morales, G.B. Mills, P.T. Ram, Identification of optimal drug combinations targeting cellular networks: integrating phospho-proteomics and computational network analysis, Cancer Res. 70 (17) (2010) 6704–6714.
[70] W.Y. Lin, W.Y. Lee, T.P. Hong, Adapting crossover and mutation rates in genetic algorithms, J. Inf. Sci. Eng. 19 (5) (2003) 889–903.
[71] MATLAB, Documentation: Global optimisation toolbox, 2019, matlab, URL https://ww2.mathworks.cn/help/gads/index.html.
[72] K. Socha, M. Dorigo, Ant colony optimisation for continuous domains, European J. Oper. Res. 185 (3) (2008) 1155–1173.
[73] Yarpiz / Mostapha Heris, Ant colony optimization for continuous domains (ACOR), 2023, [Accessed 27-07-2023], https://www.mathworks.com/matlabcentral/fileexchange/52860-ant-colony-optimisation-for-continuous-domains-acor.
[74] J. Watt, R. Borhani, A. Katsaggelos, Machine Learning Refined: Foundations, Algorithms, and Applications, Cambridge University Press, 2016, http://dx.doi.org/10.1017/CBO9781316402276.
[75] M. Jochumsen, M.S. Navid, U. Rashid, H. Haavik, I.K. Niazi, EMG-versus EEG-triggered electrical stimulation for inducing corticospinal plasticity, IEEE Trans. Neural Syst. Rehabil. Eng. 27 (9) (2019) 1901–1908.
[76] R.R. Picard, R.D. Cook, Cross-validation of regression models, J. Amer. Statist. Assoc. 79 (387) (1984) 575–583.
[77] V. Beiranvand, W. Hare, Y. Lucet, Best practices for comparing optimisation algorithms, Optimisat. Eng. 18 (4) (2017) 815–848.
[78] X. Deng, Q. Liu, Y. Deng, S. Mahadevan, An improved method to construct basic probability assignment based on the confusion matrix for classification problem, Inform. Sci. 340 (2016) 250–261.
[79] J. Jubany, R. Angulo-Barroso, An algorithm for detecting EMG onset/offset in trunk muscles during a reaction-stabilisation test, J. back musculoskelet. rehabil. 29 (2) (2016) 219–230.
[80] D.M. Powers, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, J. Mach. Learn. Technol. 2 (2011) 1.
[81] G.L. Cerone, A. Botter, M. Gazzoni, A modular, smart, and wearable system for high density sEMG detection, IEEE Trans. Biomed. Eng. 66 (12) (2019) 3371–3380.