Imperial College London

Thesis for Master of Research
MRes Neurotechnology

Optimization of Multi-channel Intramuscular EMG (iEMG) Signal Simulator

Author: Dijun Liao
CID: 01987082
Supervisor: Professor Dario Farina
September 2021

Department of Bioengineering, Imperial College London
Words count: 5997

Abstract

Recent research by Dr. Konstantin and colleagues proposed a novel multi-channel iEMG simulator (IEEE Trans Biomed Eng. 2020;67(7):2005-2014). This simulator included the relevant biophysical and physiological parameters for simulating the activity of motor units (MU). However, some critics have pointed out that the model lacks authenticity since the signal it simulated was not compared with clinical iEMG signals. Based on this, we proposed a method to optimize model parameters such as the electrode position and recruitment thresholds to best match experimental data (i.e., inverse modelling by applying real iEMG data to test the model). Since generating the iEMG from a motor unit action potential (MUAP) is very simple, we focused in this project on generating the MUAP. The optimization model consists of two parts: first, we adjusted the electrode position to fit the best MUAP by comparing four common models: gradient descent, adaptive moment estimation (ADAM) and two quasi-Newton methods, DFP and BFPS. The results showed that ADAM had the best performance. Second, we optimized the basic MU information via recruitment threshold parameters. Since the optimization is a stochastic programming problem, we applied a modified genetic algorithm to search the optimum. To improve the computational performance, the model was simplified with a distribution generating methodology. After searching by applying the optimization model, a series of MUAPs similar to actual recorded MUAPs were generated by this simulator. Furthermore, we can confirm that the iEMG signal simulated from this model is similar to experimental data.

Contents

Chapter 1

Introduction

1.1 Physiological Background

A motor unit (MU) is a basic structure consisting of a motor neuron (MN) and the muscle fibres (MF) it innervates [1],[2]. A general view of this physiological structure is shown in figure 1.1. The MU receives excitations from the upper-level neural system. When the excitation is larger than a specific value called the recruitment threshold (RT), MU will be recruited and generate an electrical signal called an action potential (AP), which is conducted along the body of the neuron [3],[4],[5].At end of the axon, AP causes the releasing of chemical signals, the neurotransmitter acetylcholine (ACh) [4],[5],[6]. Activation of ACh receptors on the post-synaptic, electrical signals will initiate on the muscle fibre, and will be conducted along it. This signal, called a single fibre action potential (SFAP), will cause the contraction of MFs to generate force, macroscopically generating movement [5],[7],[8]. A motor unit action potential (MUAP) is the spatial and temporal summation of SFAPs generated by MFs in the same MU [9],[10]. Normally, a MF will receive more than one excitation during a period of time. A 0/1-time sequence called a spike train can be generated by including all the excitations that produced APs [10],[11].

At the macroscopic level, muscle force is determined by muscle length and velocity during a contraction [12]. A single AP can cause a contraction, called a twitch. Mathematically, the total contraction force can be represented by a summation of spike trains convoluted with corresponding twitch waveform that are non-linear, rate-dependent and coefficient-weighted [4],[13],[14]. At the cellular level, several mechanisms contribute to muscle control: recruitment of a higher number of MUs or larger MUs to super-position their effect or repeated stimulation of a single MU to accumulate the contraction effect [2],[3],[15]. Theoretically, these mechanisms can be described as the firing rate, which is the average number of APs per unit time [4],[13],[16],[17]. Smaller MUs have larger input resistances and smaller RTs due to their smaller size; so, when firing increases, larger MUs will be recruited [4],[18],[19],[20].

1.2 EMG Signals and Models

1.2.1 Types of EMG Signals

Electromyography (EMG) is a technique for recording electrical activity in muscle [21],[22]. Electrical activity initiated in a MF when it produces an electrical field that surrounds the MF and extends onto the skins [21],[23]. This electrical field can be detected by a metal electrode. Two

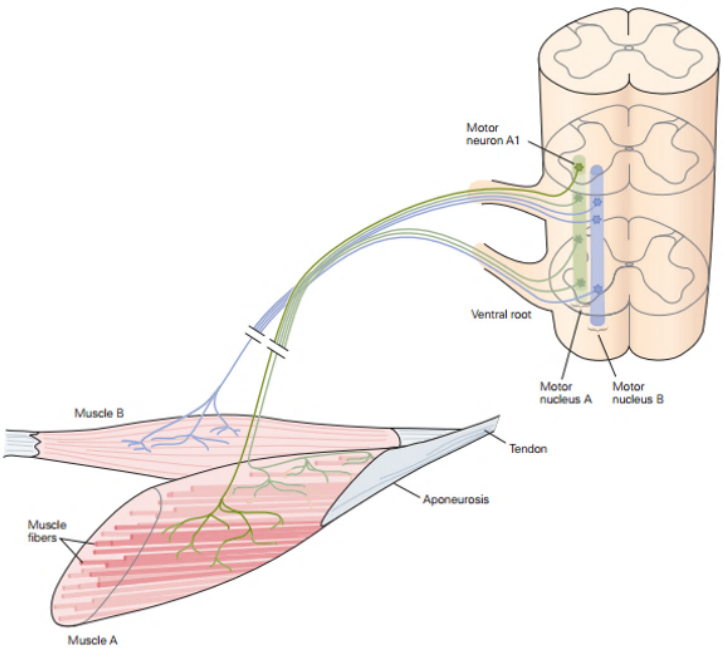


Figure 1.1: The Physiological Structure of the Motor Unit and Related Structure (Spinal Cord and Muscles) [1] As shown here, two nerve clusters (green and blue) to control the two muscles. Motor nucleus A (green nerve cluster) includes all MNs that innervate fibres in muscle A, and motor nucleus B (blue nerve cluster) includes all MNs that innervate fibres in muscle B. Muscle B only shows one MU here.

types of EMG signals assess clinically and are classified by the method of measurement: the surface EMG (sEMG) signal and the intramuscular EMG (iEMG) signal. To record surface EMG signals, a membrane/patch electrode or an electrode array is applied to the skin of a patient [24],[25]. Since this recording method is a non-invasive method, electromagnetic interference from different muscle tissues or the external environment will be higher; as a result, this method can only be used to detect the global activity of a muscle to a shallow depth [4],[26],[27],[28]. The intramuscular EMG uses a needle electrode inserted into a specific muscle area to avoid extra noise; as a result, this method yields higher measurement accuracy [4],[29],[30]. Thus, the iEMG electrode is more suitable for recording the activity of a muscle or a specific area of a muscle [4],[29],[31]. While the surface EMG and intramuscular EMG involve different modelling methods, we will only focus on the iEMG model in this project.

1.2.2 Mathematical Models of the iEMG Signal

1.2.2.1 Phenomenological iEMG Model

Many iEMG models have been proposed [32],[33],[34],[35],[36],[37],[38],[39],[40],[41],[42],[43],[44],[45]. Professor Dario Farina and colleagues proposed an iEMG decomposition model based on Professor De Luca's work in processing and EMG decomposition [39],[40],[46],[47]. Their approach convoluted the summation of experimental MUAPs in a detectable area and the experimental or simulated spike trains of these MUAPs [4],[48]. This approach is concise and can be rewritten into a hidden Markov model by transferring the spike train to the corresponding discrete sawtooth sequence [49],[50]. However, one drawback is that this method does not adequately model the physiological structures of MUs [4]. The model can be expressed as the following equation (1.1) [4],[48],[51].

$$Y_{(t)} = \sum_{i=1}^{N} H_i(t) * U_i(t) + w(t) \tag{1.1}$$

Here, N is the total number of MUs in a detectable area, $H_i(t)$ is the MUAP of i-th MU, $U_i(t)$ is the sparse vector consisting of a Dirac function sequence used to represent the spike train of i-th MU and $w(t)$ is the noise.

1.2.2.2 Biophysical iEMG Model

a) Simulation of Motor Units and Muscle Fibres

Recruitment Thresholds and Motor Neuron Size

According to the Henneman size principle, the recruitment and de-recruitment of MUs will occur in order from the smallest MU to the largest MU and will be distributed exponentially [4],[52],[53]. Furthermore, since the RTs of MUs are proportional to their sizes, the MU size can be obtained when their RT has been indicated [4],[53]. Dr. Konstantin developed a novel approach to generate RTs by applying the relationships between the maximum and the minimum RT, based on the work of A. J. Fuglevand and De Luca [54]. This approach considered both recruitment range and maximum RT whilst other methods only considered the recruitment range. The following equations can be used to represent this procedure; Equation (1.2) indicates the relationship between RTs, and Equation (1.3) represents the relationship between RT and MU size [4].

$$r_n = \frac{M}{R} \exp \frac{n-1}{N-1} \ln R \tag{1.2}$$

$$s_n = \frac{r_n}{r_N} \tag{1.3}$$

Here r_n is the RT of n-th MU, s_n is the size of n-th MU; N is the number of MUs, M is the maximum RT in those MUs, R is the range between maximum RT and minimum RT of those MUs.

Distribution of Motor Units Pool and Muscle Fibres Pool

All MFs are considered to be parallel to each other in a simulated cross-section, MUs and MNs generated in a muscle section are distributed uniformly with respect to their diameters. Previous models like [55],[56],[57]. did not consider the territory of each MU or the size of each MF, causing unevenness in the distributions and producing muscle fibre densities that were not physiologically relevant [4],[52],[55],[58],[59]. In this simulator, the farthest point sample (FPS) method is applied. As a type of sampling algorithm family, FPS generates points by adding a point at the maximum distance from former points. MUs are generated quasi-randomly whilst MFs are selected from a Gaussian distribution, taking into consideration the diameter of each MF; a larger MU will generate MFs that are also larger [60],[61],[62],[63]. Thus, the MU pool and the MF pool are created randomly and uniformly by taking into consideration the size of MUs and MFs.

Assignment of Muscle Fibres to Motor Units

After generating the MU pool and MF pool, it is necessary to assign each MF to a specific MN that innervates the MF by applying a weight consisting of three probabilities: a priory probability (P_n^a) given the size of the MN, a probability (P_n^g) donate the MN territory centre using a symmetrical two-dimensional Gaussian distribution and an indicator (P^d) that demonstrates whether any close neighbours of an MF have been assigned (if yes, return to 0; if no, return to 1), assuming circular MU territories [64],[65]. The expression of this probability can be represented as the following equation (1.4) [4],[52]:

$$P_f(n) \sim P_n^a \cdot P_n^g(x_f, y_f) \cdot P^d(n, f, n_c) \tag{1.4}$$

b) Simulation of the Neuromuscular Junction

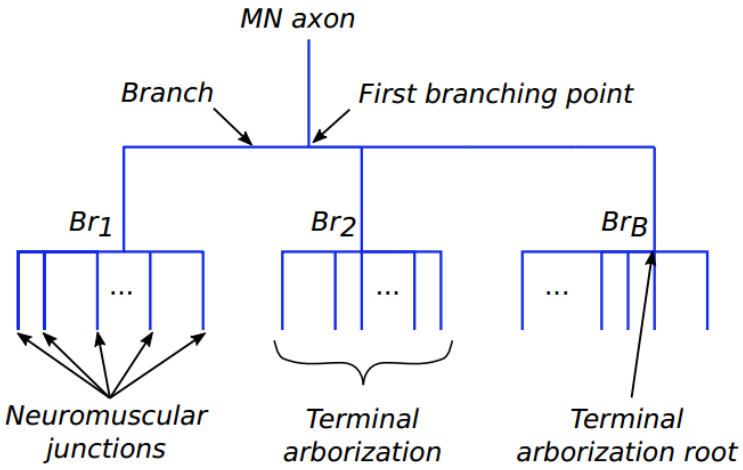
The Neuromuscular Junction and the Assignment of Muscle Fibre to Axon Branches

The NMJ is a tree-root structure that splits into several branches at the end of axons; each branch controls one MF. In this simulator, to simplify the structure but maintain physiological accuracy, the end of axon for each MU will only split twice. Assuming that the number of branches for each MU is proportional to the MU's size, the following equation (1.5) can be used to calculate the number of branches in each MU [4].

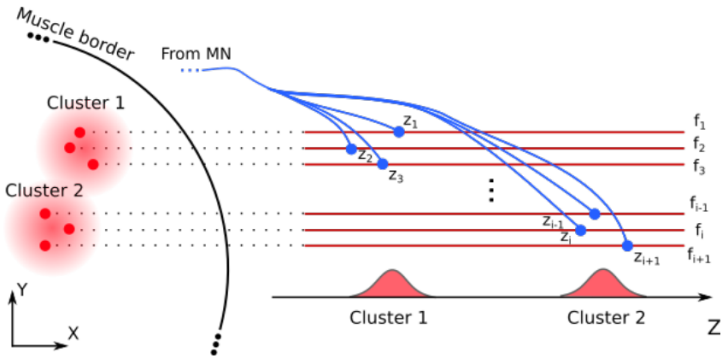
$$B(n) = 1 + \lfloor \ln \frac{s_n}{s_1} \rceil \tag{1.5}$$

Here, s_n is the size of n-th MU and $\lfloor \cdot \rceil$ is to find the nearest integer. Figure 1.2 shows splitting at the end of the axon.

To generate the coordinates of the neuromuscular junction, we used the following model: the x and y coordinates are the same as the x and y coordinates of the muscle fibre, and the z coordinate will be generated from two Gaussian distributions [4],[52].



(a) the structure simulation of NMJ



(b) the simulation of generating z-coordinate of NMJ

Figure 1.2: Structure and coordinate modelling of MU branches [4] (a) The MU axon branches shown here are modelled as a bifurcation tree, where the root is the axon and the leaves are organized into terminal arborizations Br_i. Each cluster was classified using a k-mean algorithm. (b) The z-coordinate of branches were generated by choosing from two Gaussian distributions where the first Gaussian distribution was used to confirm the first branch point, and the second distribution was used to confirm the NMJ in each cluster.

Motor Neuron Action Potential Delay

The motor neuron action potential (MNAP) propagation delay can be calculated separately by considering the different conduct velocities in different parts [4],[52]. Furthermore, the conduction velocities in each branch are assumed to be the same and much smaller than the conduction velocity in the axon.

c) Generation of the Motor Unit Action Potential

Generation of the Single Fibre Action Potential

The MUAP is the summation of SFAPs in a MU taking into account their MNAP delay [4],[52]. In this model, the SFAP is calculated with the convoluted transmembrane current density, which is the second derivative of transmembrane potential along the z-axis and the elementary current potential modulated in a time-space domain [66]. The following equation (1.6) is used to calculate the transmembrane potential, where A is $96mV \cdot mm^3$, B is the resting potential, which is $90mV$ [4],[52],[66].

$$V_m(z) = \begin{cases} Az^3 \exp(-z) + B & z > 0 \\ 0 & z \leq 0 \end{cases} \tag{1.6}$$

Equation (1.7) represents the transmembrane current density; d_f is the diameter of the muscle fibre, and z_{0f} is the z-coordinate of NMJ in f-th fibre. L_f and R_f are the distances from the NMJ in the f-th fibre to the left and right end of the fibre. v_f is the fibre conduction velocity [57],[67]. Note that the function $\psi(z) = \frac{dV_m(z)}{dz}$, p_{L_f} and p_{R_f} are two indicators equal to 1, where are $-\frac{L_f}{2} \leq z \leq \frac{L_f}{2}$ and $-\frac{R_f}{2} \leq z \leq \frac{R_f}{2}$ [4].

$$I_f(z,t) = d_f \frac{\partial}{\partial z}[\psi(z - z_{0f}) - v_f \cdot t) \cdot p_{L_f}(z - z_{0f} - \frac{L_f}{2}) - \psi(-z + z_{0f}) - v_f \cdot t) \cdot p_{R_f}(z - z_{0f} - \frac{R_f}{2})] \tag{1.7}$$

Equation (1.8) describes the SFAP in an observing point $P(x_p, y_p, z_p)$. Here ϕ_{fp} is the potential of a single fibre, the κ_r and κ_z are the conductivity along the radius and the axon and I_e is the elementary current source [54],[68].

$$\phi_{fp} = \frac{1}{4\kappa_r} \cdot \frac{I_e}{\sqrt{r_{fp}^2 \cdot \frac{\kappa_r}{\kappa_z} + (z_p - z)^2}} = h_{fp}(z_p - z) I_e \tag{1.8}$$

By using the transmembrane current density, the above equation can be rewritten as the following equation (1.9) [54]:

$$\phi_{fp}(t) = \int_{-\infty}^{+\infty} h_{fp}(z_p - z) I_f(z,t) dz \tag{1.9}$$

From Single Fibre Action Potential to Motor Unit Action Potential

MUAPs are generated by summation of all SFAPs in the MU, taking into consideration the MNAP propagation delays. The following equation (1.10) demonstrates that how to generate MUAP from SFAPs [57],[69].

$$\Phi_{np}(t) = \sum_{f=1}^{N} \widetilde{\phi}_{fp}(t - d_f - \zeta) \tag{1.10}$$

Here, F_n is the number of fibres innervated by MU, d_f is the delay of MNAP propagation and ζ is the delay cause by the NMJ jitter.

d) Generation of the iEMG

Target Profile

In neural coding, excitation is generated by external input from other sensory organs [70]. After this information has been inputted, the brain will process this information and send a series of excitations to the next level of the neural system. This is a dynamic procedure that involves adjustment of the movement in real time. In this model, a target profile has been used to generate the basic movement form and a PID was used to provide negative feedback for the closed-loop control [70],[71],[72]. The excitation series can be generated according to the target profile and after adjustment of the PID. The recruit and de-recruit procedure can then be modelled by comparing the excitations with RTs. The excitations are also used to generate the spike train since it the excitations are necessary to generate APs.

Generation of multi-channel iEMG

An intramuscular EMG signal is generated by a convoluted MUAP and the associated spike train [4],[52]. By considering all detectable MUs at all observing points and electrodes in a channel, a single channel iEMG signal can be obtained. Simulating multi-channel iEMG signals simply requires summing signal-channel iEMG signals in all channels.

The detectable MUs of this model are generated by excluding MUs that are much lower than instrumental noise and including MUs that are much higher than noise [4].

1.3 Objectives

Though this model is well-simulated, physiological and biophysical information is needed to generate an iEMG signal. Several criticisms have questioned the authenticity of this model since it has not been tested by applying experimental data. Thus, the main objective of this project was to test the model and search for appropriate parameters using optimization methods. Furthermore, the optimization model can also be used to test the performance of a single channel for an iEMG signal. The two intersections of the green and the cyan circle in figure 1.3 shows the points that can used for testing.

All of the parameters used in this model were selected from the experimental data of Feinstein's paper, which simulated the first dorsal interosseous of a 22-year-old man [73].

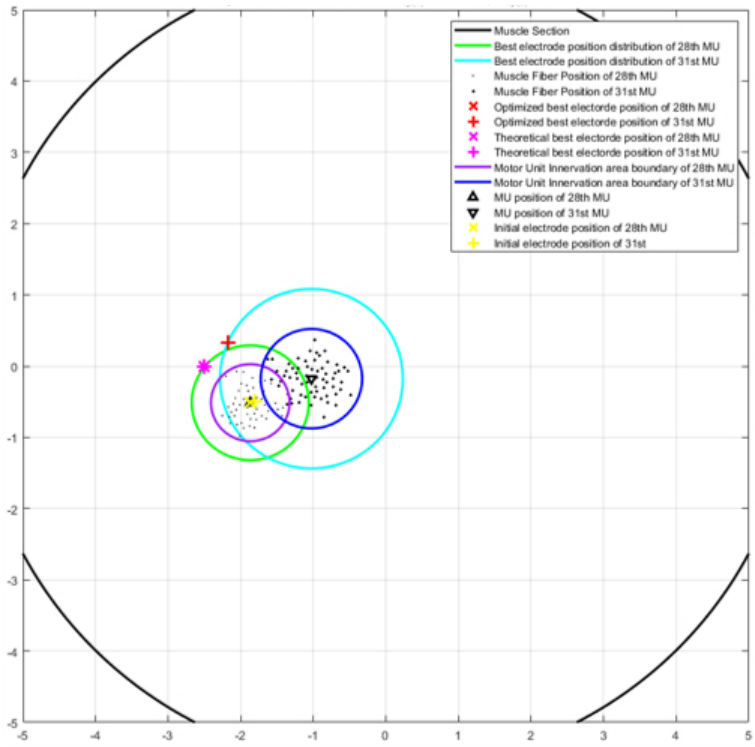


Figure 1.3: A sketch for finding points that can test the performance of a single channel iEMG signal When two similar MUAPs are obtained, theoretically the electrode surrounding the MU should be get MUAPs similar since the MFs distance between electrodes and MFs can regards as same. In this figure, the purple circle and the blue circle indicate two MUs, and the black dots are the MFs centres. Electrodes placed in the green and cyan circles will obtain the same MUAPs, and in the two intersections of the two circles (green and blue), it is hard to identify which can be used to test the single channel decomposition algorithm. The red and magenta points are the optimized electrode positions.

Chapter 2

Methodology and Result

2.1 Overview

According to Professor Dario Farina's model, generating iEMG signals from MUAPs simply requires convoluting MUAPs and their associated spike trains [4],[51]. Our optimization model will search the appropriate parameters to best match experimental MUAP data in this project. Due to the different independent procedures of this model, the entire optimization procedure can be viewed as two black boxes with various inputs and outputs. Figure 2.1 is a sketch of the entire optimization procedure. The first optimization is to adjust the electrode position under stationary muscle morphology and tissue distribution conditions. Another optimization is determining a suitable MU innervation. Since all physiological structure parameters are generated according to the MU size, which is obtained from the RT in this model, we optimized the RT parameters.

Since the procedure to generate an iEMG from a MUAP is simple, this project focused on optimising real MUAP shapes only.

2.2 Optimization of the Electrode Position

Optimizing the electrode position is performed independent of any other procedure in this simulator. By only adjusting the electrode position, the shape of the MUAP can also be adjusted. There are many stochastic processes in this simulator: the assignment of MFs to MUs and specific branches, the generation of the diameters of the MFs. These processes make the simulator complex and hard to optimize. However, this part has been fixed all these stochastic processes according to fixed all related parameters, making it simple to apply conventional algorithms like gradient descent and Newton method to the optimization process. Furthermore, this optimization model can also evaluate the performance of a single channel iEMG decomposition algorithm by adjusting the electrode position to generate two similar MUAPs in a single channel iEMG signal.

2.2.1 Loss function

In this section, we expect to find that a simulated MUAP has a shape that is close to the real experimental MUAP. We used a common method to describe the differences between the real MUAP and the simulated MUAP — the loss function. Since both the real MUAP and the simulated MUAP are time sequences, we can first construct a square loss function between each

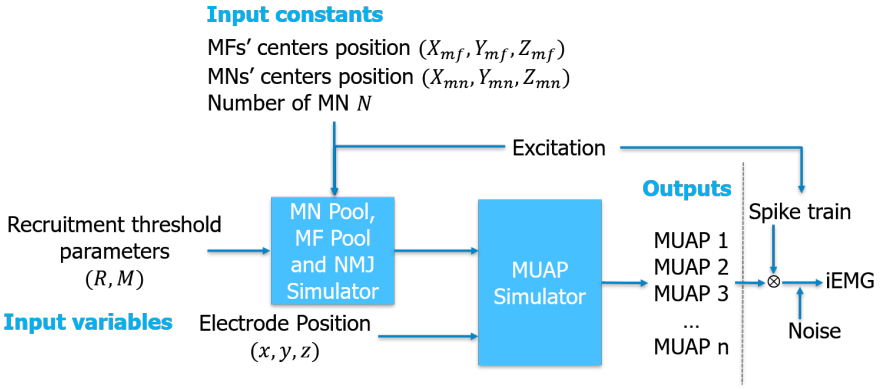


Figure 2.1: The structure of optimization procedure The two blue boxes are two black boxes, when the first optimization (i.e. optimize electrode position) has been processed, the outputs of the left black box are regards as the input constant. Input variables in this optimization are only the electrode position. The result of this part is only one MUAP. For the second optimization (i.e. optimize MU innervation territories and assign method via RT parameters), the RT parameters are input variables but electrode positions are constants. To simplify this model, all positions of MUs and MFs are fixed as constants. In this part, the output is the MUAPs, and for the optimization, it is a multi-objective optimization problem.

time step of these two signals and then add these loss function elements. The function obtained from this procedure can then be optimized. The square loss function is easy to deal with and has symmetry that is similar to the variance [74]. The following equation (2.1) shows the loss function of this optimization step:

$$L = \sum_{t=1}^{T_s} (\Phi_{real}(t) - \Phi_{sim}(t - d_f, x_p, y_p, z_p))^2 \tag{2.1}$$

Here, T_s is the total time step that MUAP lasts. A single time step is considered to be 10^{-5} s. Φ_{real} is the real MUAP recorded from experimental data, and Φ_{sim} is the simulated MUAP generated by applying this simulator. x_p, y_p, z_p are three coordinates of the electrode position.

2.2.2 Methodology

By introducing the square loss function, the optimization problem can write the following expression (2.2). Since the electrode position only depends the SFAPs to the MUAP generating procedure, this part is simply a function of the position of the electrode without any stochastic processes. Results can be obtained by only applying the common first-order or second-order methods.

$$\begin{aligned} \min_{x_p, y_p, z_p} \quad & L \\ s.t. \quad & x_p^2 + y_p^2 \leq R_{muscle}^2 \\ & 0 \leq z_p \leq L_{muscle} \end{aligned} \tag{2.2}$$

where L is the loss function between the real MUAP and the simulated MUAP, which is shown in equation (2.1). The electrode position is represented as a point $P(x_p, y_p, z_p)$. To ensure that

the electrode is in the muscle, the optimization was processed under the condition that the x-coordinate and y-coordinate not exceed the radius of the cross-section of the muscle (i.e., a circle has its origin at the centre, and the radius is R_{muscle}). In addition, the z-coordinate of it must not exceed the length of muscle L_{muscle}.

2.2.2.1 The First-Order Method

We chose the traditional first and second-order methods because of the stable simulator values. To compare and identify the optimization method with the best performance, we compared four different methods: the gradient descent (GD), the adaptive moment estimation (ADAM) and two quasi-Newton methods. The gradient descent and ADAM are first-order methods that only use the gradient information to search the minimum. Assume that an objective function $f(x, y)$ is convex and existing global minimum. Its minimum can be found by searching along the inverse direction of the gradient. An iteration of GD procedure can be represented as equation (2.3), where $\boldsymbol{x^{(k+1)}}$ is the position of the searching point in the next iteration, $\boldsymbol{x^{(k)}}$ is the current iteration point, α is the learning ratio (i.e., the step size for each iteration) and $\boldsymbol{d^{(k)}}$ is a vector that represents searching. It is the reverse of gradient $\boldsymbol{g^{(k)}}$ at the current point [75].

$$\boldsymbol{x^{(k+1)}} = \boldsymbol{x^{(k)}} + \alpha \cdot \boldsymbol{d^{(k)}} \tag{2.3a}$$

$$\boldsymbol{d^{(k)}} = -\boldsymbol{g^{(k)}} \tag{2.3b}$$

GD is utilized as the baseline here. It is necessary to select an appropriate value since the result may diverge if it is too large and may be slow to converge if too small [76]. In this project, we used the line search to find a best value at $\alpha = 0.1$.

GD has many weaknesses; the searching step will be too small when approaching optimum which cause its converge too slow. The orthogonal direction of adjacent searching causing the Zigzag descent. Methods like AdaGard, RMSProp and ADAM avoid these by keeping the learning ratio adaptable and the searching directions as straight as possible in each iteration. Since ADAM methods have all the advantages of AdaGard and RMSProp, we applied this method here to search for a suitable electrode position. Equations (2.4) show the iteration equations of ADAM [75]:

Biased decaying momentum

$$\boldsymbol{v^{(k+1)}} = \gamma_v \boldsymbol{v^{(k)}} + (1 - \gamma_v)\boldsymbol{g^{(k)}} \tag{2.4a}$$

Biased decaying square gradient

$$\boldsymbol{s^{(k+1)}} = \gamma_s \boldsymbol{s^{(k)}} + (1 - \gamma_s)(\boldsymbol{g^{(k)}} \odot \boldsymbol{g^{(k)}}) \tag{2.4b}$$

Corrected decaying momentum

$$\boldsymbol{\hat{v}^{(k+1)}} = \frac{\boldsymbol{v^{(k)}}}{1 - \gamma_v} \tag{2.4c}$$

Corrected decaying square gradient

$$\boldsymbol{\hat{s}^{(k+1)}} = \frac{\boldsymbol{s^{(k)}}}{1 - \gamma_s} \tag{2.4d}$$

Next iteration

$$\boldsymbol{x^{(k+1)}} = \boldsymbol{x^{(k)}} - \frac{\alpha \hat{\boldsymbol{v}}^{(\boldsymbol{k+1})}}{\epsilon + \sqrt{\hat{\boldsymbol{s}}^{(\boldsymbol{k+1})}}} \tag{2.4e}$$

Compared to other existing methods, ADAM has the best performance in most situations since it is highly efficient and requires lower memory. The hyper-parameters of ADAM can be explained very intuitively and require fewer parameter adjustments. Here, we found suitable parameters by applying dichotomies and starting from the optimization parameters mentioned in [77] $\alpha = 0.001, \gamma_v = 0.9, \gamma_s = 0.999 and \epsilon = 10^{-8}$. We find that the most suitable parameters were the following: $\alpha = 0.001, \gamma_v = 0.99, \gamma_s = 0.999 and \epsilon = 10^{-4}$.

2.2.2.2 The Second-Order Method

Another approach to improve the performance of the first-order method is to include more descending information of the objective function. As a further development of the gradient descent, the Newton method allows the first two terms in the Taylor series of an objective function whilst the gradient descent only applies to the first one term. The Newton method considers both the variety tendencies and the gradient of an objective function instead of the local properties of an objective function in the current iteration point to confirm the searching direction and the accuracy of the convergence speed [78]. However, the Newton methods require more computation to calculate the Hessian matrices [78]. The Hessian matrix of an objective function must be positive-definite when applying the Newton method. Equation (2.5) is the iteration equation of the Newton method [75]. Here $\boldsymbol{x^{(k)}}$ and $\boldsymbol{x^{(k+1)}}$ are the point in the current and next iteration, $\boldsymbol{H^{(k)}}$ is the Hessian matrix of the objective function and $\boldsymbol{g^{(k)}}$ is the gradient descent.

$$\boldsymbol{x^{(k+1)}} = \boldsymbol{x^{(k)}} - (\boldsymbol{H^{(k)}})^{-1} \boldsymbol{g^{(k)}} \tag{2.5}$$

Due to the massive amount of computation required to apply the original method to calculating the Hessian matrix, here we chose two quasi-Newton methods (DFP and BFGS) to estimate the inverse Hessian matrix directly. The following equation (2.6) shows these two quasi-Newton methods with the initial $\boldsymbol{Q}$ matrix is the unit matrix and $\gamma^{(k+1)} = \boldsymbol{g^{(k+1)}} - \boldsymbol{g^{(k)}}$ and $\delta^{k+1} = \boldsymbol{x^{(k+1)}} - \boldsymbol{x^{(k)}}$.

DFP Quasi-Newton Method

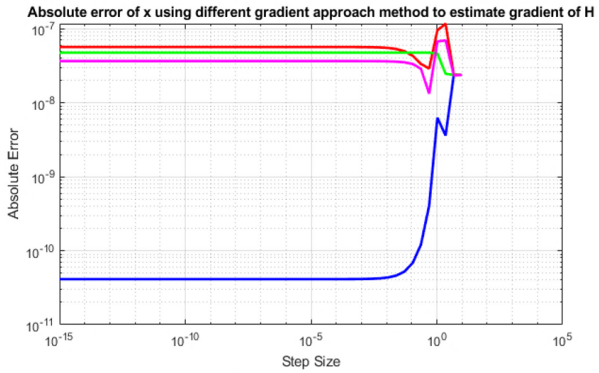
$$\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \frac{\boldsymbol{Q}\gamma\gamma^T\boldsymbol{Q}}{\gamma^T\boldsymbol{Q}\gamma} + \frac{\boldsymbol{Q}\delta\delta^T}{\delta^T\gamma} \tag{2.6a}$$

BFGS Quasi-Newton Method

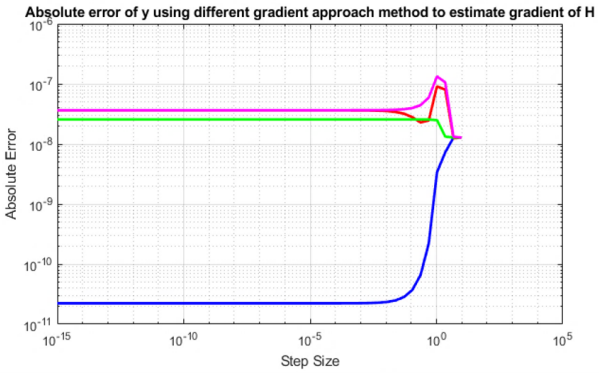
$$\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \frac{\delta\gamma^T\boldsymbol{Q} + \boldsymbol{Q}\gamma\delta^T}{\delta^T\gamma} + (1 + \frac{\gamma^T\boldsymbol{Q}\gamma}{\delta^T\gamma})\frac{\delta\delta^T}{\delta^T\gamma} \tag{2.6b}$$

2.2.2.3 Computation of Gradient

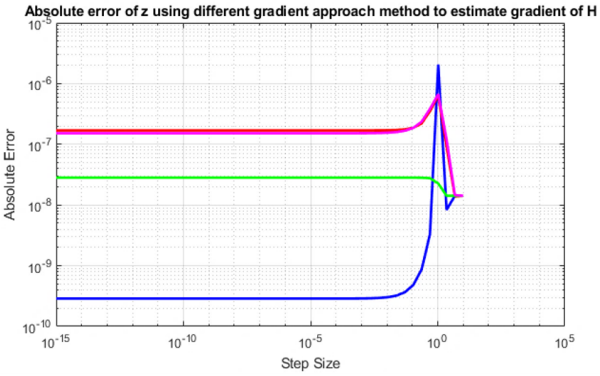
Here, we computed the gradient using the complex step method to calculate the gradient by taking into an imaginary direction. This method is the most accurate method since it bypasses the effect of subtractive cancellation by using a single function evaluation [75]. Figure 2.2 shows a comparison of absolute errors between the complex step method and other reality step methods (i.e., forward, central and backward step method).



(a) the absolute error of x for this MUAP simulator



(b) the absolute error of y for this MUAP simulator



(c) the absolute error of z for this MUAP simulator

Figure 2.2: The absolute error compared between the complex step method and the three reality step method The red line indicates the absolute errors of the forward difference method, the green line is the absolute errors of the centre difference method, the magenta line is the absolute errors of the backward difference method and the blue line is the absolute errors of the complex step method. The absolute errors of the complex step method are significantly smaller compared to the other three methods, especially when the step size is much smaller than 10^0.

2.2.3 Result

2.2.3.1 The Comparison of the Four Methods

Here we compare the above four methods: gradient descent, ADAM, DFP and BFGS quasi-Newton. The following table 2.1 shows the of loss function in final iteration and related optimization information.

Table 2.1: The results of above these four optimization methods

Method	Final Loss Function	Iteration	Optimization time
Gradient Descent	0.28	2070	503.884s
ADAM	0.00105	3547	1062.324s
Quasi-Newton (DFP)	8.1898	36	11.550s
Quasi-Newton (BFGS)	14.9304	8	2.9761s

Here, we compare the four methods: gradient descent, ADAM, DFP and BFGS quasi-Newton methods. Table 2.1 shows the loss function in the final iteration across these four methods. As shown here, ADAM has the best performance at a loss function of 0.00105, which is lower than the limitation of 0.01 although it requires more time to converge. For the gradient descent method, it is terminated on a small interval due to the fixed learning ratio. For the two quasi-Newton methods, the results are too large. The optimization was terminated because the parameters were out-of-boundary (i.e., divergent).

2.2.3.2 The Result for Optimizing the Electrode Position

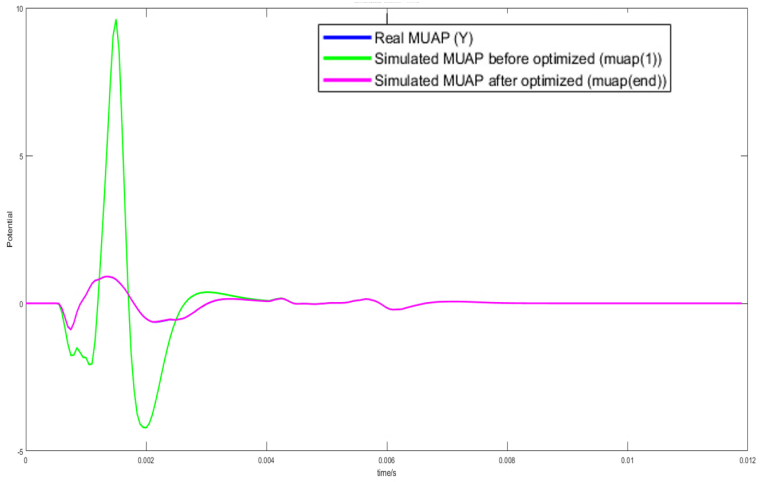
Figure 2.3 shows the results of the ADAM optimization; and the final position after optimization was $(-1.8, 0.5, 15)$,The optimization lasted 1062.324 seconds and iterated 3547 times, converging at the final position $(-2.49, -0.01, 14.98)$. The optimization last 1062.324 seconds and iterate 3547 times, converge at the final position $(-2.49, -0.01, 14.98)$ where the loss function was 0.00105.

2.3 Optimization of Motor Unit Territories and Muscle Fibre Innervation via Recruitment Threshold Parameters

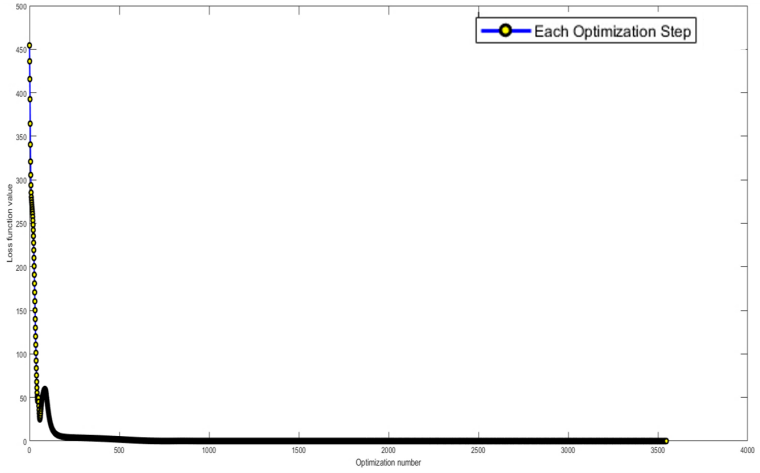
Another focus of this simulator is simulating the physiological structure of muscles. All physiological structures in this simulator are generating related to the sizes of MUs. The larger the MU is, the greater the number and stronger the MFs innervated by it will be. This also influences the MNAP delay since a stronger MF can conduct APs faster due to its lower input resistance. RTs display a distribution that can be selected from an exponential function. Thus, the simulator simulates the RTs first. Here, we consider RTs to be the initial inputs of the optimization procedure in this part.

2.3.1 Loss function

A muscle is controlled by several MUs. The iEMG signal we simulated here is the MVC iEMG, which contains MUAPs generated by all the MUs that control this muscle in a maximum voluntary contraction (MVC) theoretically [79]. However, in a real scenario, the iEMG recording



(a) Comparison of the Real MUAP, Initial Simulated MUAP and Simulated MUAP after Optimization



(b) Convergence of Loss Function for each Iteration

Figure 2.3: The result of optimizing the electrode positions by applying the ADAM algorithm (a) The blue line indicates the real MUAP and is almost entirely covered by a simulated MUAP after optimization, which is shown by the magenta line. The green line indicates the simulated MUAP before iteration. (b) The convergence of ADAM is shown here; it falls fast at first and finally converges at a value of nearly 0.

equipment and other tissues may generate noise. A considerable number of MUAPs with maximum peak values that were much smaller than the baseline noise could not be recorded clearly and were considered to be instrumental noise in the recording (that is, the iEMG electrodes had a detectable area that could only effectively record the MUAPs in that area) [4]. Thus, we could only search for appropriately simulated MUAPs that were similar to the detectable MUAPs when performing optimization.

It is a multi-objective optimization problem. Here, we will compare the difference between a real MUAP and a simulated MUAP one at a time in a detectable set. The loss function for each MUAP is similar to the example shown in (2.1); to use the square loss function, we will use the loss function shown in equation (2.7) is the loss function here we use:

$$L^{(n)} = \sum_{t=1}^{T_s} (\Phi_{real}^{(n)}(t) - \Phi_{sim}^{(n)}(t - d_f, r_r, r_m))^2 \tag{2.7}$$

where $L^{(n)}$ is the loss function for n-th MU in a detectable area, $\Phi_{real}^{(n)}$ and $\Phi_{sim}^{(n)}$ are the experimental and simulated MUAP of n-th MU, r_r and r_m are two parameters to generate RTs for all MUs in which $r_r = r_N r_1$ indicates the range between the maximum RT and the minimum RT and $r_m = r_N$ is the value of the maximum RT.

2.3.2 Methodology

To make the model more realistic, Dr Konstantin applies three stochastic processes to assign MFs to a MU, to assign MFs to special branches on a MN axon and the diameters of MFs [4],[52]. However, this makes the simulator complex and hard to optimize since the MUAP generates different parameters and different MF-to-MU assignments for each iteration. This will cause the MUAPs to be quite different in each iteration. Figure 2.4 shows this process.

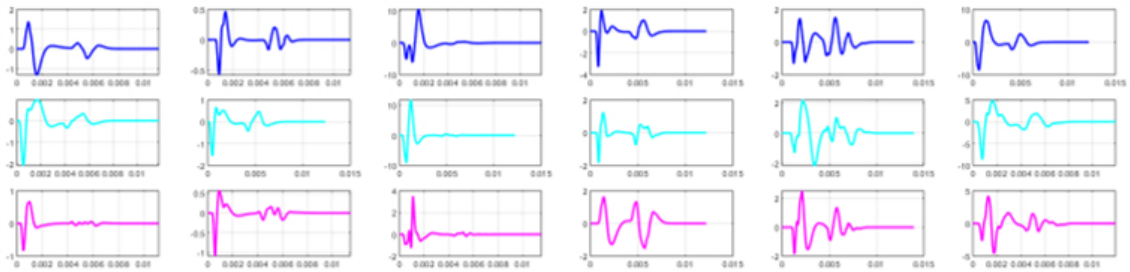


Figure 2.4: MUAPs at different times of simulation in a detectable area using this simulator Here, we chose a detectable area that included 6 MUs and simulated their MUAPs three times. The MUAPs generated were different for each MU each time, though some of them were very similar.

Traditional gradient or Hessian methods were not available here because of the random MU innervation assignment in each iteration. Two possible optimization types that may be used here are the traversal algorithm and the heuristic algorithm based on intuition and experiences to speculate the possible positions of parameters that the global optimal may exist and then applying some mechanism to narrow the search range until finding an appropriate value [75]. Since the traversal algorithm requires a large amount of computation to generate the physiological structure simulation for a large number of parameters, this algorithm is not appropriate here.

2.3.2.1 Choosing the Optimization Method

Many types of heuristic algorithms have been proposed, including the simulated annealing algorithm (SA), the genetic algorithm (GA), the Tabu search (TS) and the ant colony algorithm (ACA) [80],[81],[82]. The ACA is cleverer at dealing with route searching problems [82],[83]. Though the SA and TS algorithms are based on neighbourhood searching, which shares the same foundations with GA, these two methods are not good at dealing with stochastic optimization problems [80],[84],[85],[86]. The TS method forbids the optimiser searching approach to the local optimum under the distribution on current iteration though it may be the global optimum in following iterations for a stochastic optimization model [81]. The SA method requires a searching point moving randomly in each iteration [80]. This method can avoid the problems of the TS algorithm, but its movement is limited to a small region surrounding searching points rather than executing a random movement for searching points across a large range in a feasible region as a result, the searching performance of the SA method is lower than GA [80],[87]. Because of these limitations, we chose to use the GA method as the optimization algorithm.

2.3.2.2 Genetic Algorithm

This part is a multi-objective optimization problem. One possible method is to transform the multi-objective optimization problem into a single-objective optimization by summating the loss function of each objective considered a weight [75],[88]. This is the simplest method but requires extra hyper-parameter optimization to find the appropriate weight for each objective, making the optimiser more complex and increasing the time needed for hyper-parameter searching [89]. To avoid this, we applied a fast non-dominated sorting genetic algorithm (NSGA- II) mentioned in a study by Dr. Konak [90].

NSGA-II is a popular multi-objective GA proposed by Deb based on former studies on NSGA [90],[91]. For multi-objective optimizations, a solution that is optimal for all objectives compared to other solutions is called a non-dominated solution. Many non-dominated solutions exist for a multi-objective optimisation; the set of these non-dominated solutions are called the Pareto set [92]. NSGA and NSGA-II search the Pareto set by layering the sorted non-dominated solutions. Compared to NSGA, NSGA-II decreases the sorting complexity, making it faster and increasing the convergence for a solution set [91]. The following pseudo-code algorithm 1 illustrates how the NSAG-II algorithm produces the non-dominated solution sorting.

A Pareto optimal solution set on the first front was obtained after sorting and layering. NSGA and other methods require manual handling of these solutions to choose the best one according to the different requirements in the real scenario [90]. However, NSGA-II provides a general method to select an appropriate solution by calculating the crowding distance of each solution, sorting them in order and choose the solution with the largest crowding distance in the first layer of the non-dominated level [93]. The crowding distance is defined as the half perimeter (i.e., the sum of the length of the long side and the short side) of the cuboid that determines the nearest two points in the same front for the solution which is not the boundary point of this front [93]. Boundary points have a crowding distance equal to infinite [93]. This is shown in the following figure 2.5. Equation (2.8) shows the calculation of the crowding distance [93]:

$$\mathcal{I}_{(i)} = \begin{cases} \sum_{i=2}^{l-1} \frac{\left| f_m^{(i-1)} - f_m^{(i-1)} \right|}{f_m^{max} - f_m^{min}} & i \neq 1 \quad or \quad i \neq l \\ +\infty & i = 1 \quad or \quad i = l \end{cases} \tag{2.8}$$

where $f_m^{(k)}$ is the m-th objective function of k-th solution; f_m^{max} and f_m^{min} are the maximum and minimum objective values of m-th objective function in same Pareto set and l is the solution number in the front. Algorithm 2 presents the method used to generate the crowding distance .

The objective function used is the loss function for each objective. After the non-dominated sort and crowding distance assignment has been performed, this is optimized by applying a genetic algorithm. However, the assignment of MF to MU costs too much time. To improve the optimization speed and limit the memories it takes, we adjusted the original genetic algorithm by limiting the populations of each offspring and the crossover mate numbers for each individual. The critical procedure to execute the genetic algorithm is shown here:

Algorithm 1 The sorting algorithm for non-dominated solutions in a solution set [93] (non_dim_sort(P))

Input:
The solution set P
Output:
The layered non-dominated solution set $\mathcal{F}$

for each $p \in P$ **do**
  $S_p = \emptyset$;
  $n_p = 0$;
  for each $q \in P$ **do**
    if $(p \prec q)$ **then** ▷ If p dominate q
      $S_p = S_p \cup q$; ▷ Add q to the solution set dominated by p
    else if $(q \prec p)$ **then**
      $n_p = n_p + 1$; ▷ Increasing the dominate level counting number of p
    end if
  end for
  if $n_p = 0$ **then** ▷ If p belongs first front
    $p_{rank} = 1$;
    $\mathcal{F}_1 = \mathcal{F}_1 \cup p$;
  end if
end for
$i = 1$; ▷ Initialize the front counter
while $\mathcal{F}_i \neq \emptyset$ **do**
  $Q = \emptyset$; ▷ Temporary store the members of the next front
  for each $p \in \mathcal{F}_i$ **do**
    for each $q \in S_p$ **do**
      $n_q = n_q - 1$;
      if $n_q = 0$ **then**
        $q_{rank} = i + 1$; ▷ q belongs to the next front
        $Q = Q \cup q$;
      end if
    end for
  end for
  $i = i + 1$;
  $\mathcal{F}_i = Q$;
end while

Algorithm 2 Generation of the crowding distance [93] (crowd_dis_gen($\mathcal{P}$))

Input:
The solution set in a front $\mathcal{P}$
Output:
The crowding distance for each solution $\mathcal{I}_{(i)}$
Sorted solution set in a front $\mathcal{P}$

$l = |\mathcal{P}|$ ▷ number of solutions in P
for each i, **set** $\mathcal{I}_{(i)} = 0$ **do** ▷ initialize distance
  for each ojective m **do**
    $\mathcal{P}$ =sort$(\mathcal{P}, m)$; ▷ sort solutions in $\mathcal{P}$ according to m-th objective function
    $\mathcal{I}_{(1)} = \mathcal{I}_{(l)} = +\infty$; ▷ the crowding distance of boundary point are setting as infinity
    for $i = 2$ **to** $i = l - 1$ **do**
      $\mathcal{I}_{(i)} = \mathcal{I}_{(i)} + \frac{|f_m^{(i-1)} - f_m^{(i-1)}|}{f_m^{max} - f_m^{min}}$
    end for
  end for
end for

- **Genetic pool of initial parents** — The genetic pool of initial parents is generated randomly under the limitation $r_r \in [10, 100)$ and $r_m \in [0.1, 1)$ here.

- **The fitness function of each individual in initial parent** — The fitness functions are the linear summation of loss function in each objective here.

- **Crossover** — First, an individual will randomly choose a limited number of crossover mates according to a weight which is the probability that consist of the reciprocal of fitness function and multiplied with the reciprocal of non-dominated layer number. The following equation (2.9) shows the calculation of this weight. Here, $F_{fit}(n)$ is the fitness function for n-th individual in a parent's genetic pool and N is the total number of this genetic pool.

$$w_n = \frac{P_c^{(n)}}{\sum_{i=1}^{N} P_c^{(i)}} \tag{2.9a}$$

where P_n is the crossover probability and can be calculated:

$$P_c^{(n)} = \frac{1}{F_{fit}(n) \cdot n_p} \tag{2.9b}$$

where the fitness function is:

$$F_{fit}(n) = \sum_{i=1}^{O} L^{(i)}(n) \tag{2.9c}$$

In equation (2.9b), $F_{fit}(n)$ is the fitness function of the n-th individual in a genetic pool and n_p is the level of the non-dominated layer. In equation (2.9c), O is the total number of objectives and $L^{(k)}(n)$ is the loss function of the k-th objective for n-th individual in the genetic pool. Here, if the Pareto frontier level of this point is higher than a limited level or is the boundary point, its crossover probability will be set to a very small value.

- **Variation** — After crossover and generation of the full genetic pool of offspring, we executed the variation operation by randomly selecting individuals from the full genetic pool of offspring. The parameters designated the different parts of a chromosome. In this part, since there are only two parameters, we randomly appointed the selected offspring to vary by one parameter. The variation rate here is set at 0.1.

- **Natural Selection** — After variation, we combined the original parent's genetic pool, the fully generated genetic pool of the offspring and the extra genetic pool to maintain the genetic diversity. The surviving offspring were then selected from this new genetic pool by considering their non-dominated level (the highest priority), fitness function and crowding distance for non-boundary points lower than a select level. For the boundary points, we set the natural selection probability to 0.1; for points higher than the selected level, the probability was 0.01. This approach keeps the genetic pool dispersed as much as possible, and all objectives are expected to be highly matched.

- **Optimum Selection** — The initial optimum we selected is the parameter pair which has the highest crowding distance that excludes boundary points in the first Pareto frontier set if there were two more points in the set. Since we expect that all objectives are small enough instead of some of the objectives are too big though the others are small enough. If there are two points, the algorithm will choose from the one with the small fitness function. For each iteration, we will choose the parameters that have the smallest fitness function compared to the former best parameter pair. If all objective functions in the pair are smaller than the former pair, the pair will be accepted as the new best optimum.

Algorithm 3 shows the approach of genetic algorithm.

Algorithm 3 Genetic Algorithm for this Optimization

Input:
Randomly generated parent genetic pool P
Set for real MUAPs Y
Pareto frontier level limit n_l
Crossover mate limit n_c
Maximum generation limit g_{lim}

Output:
The best optimal value $L_(n)$ and its parameters (r_r, r_m)
The MUAPs of best optimal value $\Phi_{(np)}$

for each parameters-pair $p \in P$ **do**
  $F_{fit}(p) = 0$;
  Simulate MUAPs using parameters in P and generate $L^{(i)}(p)$;
    ▷ Simulate MUAPs and generate of fitness function for each parameters-pair
  for each objective O **do**
    $F_{fit}(p) = F_{fit}(p) + L^{(i)}(p)$;
  end for
end for
non_dim_sort(P);
for each parameters-pair $p \in P$ **do** ▷ Generate of cross probabilities
  if $n_p(p) \geq n_l$ **then**
    $P_c^{(p)} = 0.01$; ▷ Set a small value if the Pareto frontier level too high
  else

if p is the boundary point in this Pareto frontier level **then**
▷ Set a small value if the point is the boundary point in low Pareto frontier
$P_c^{(p)} = 0.1$;
else
$P_c^{(p)} = \frac{1}{F_{fit}(n) \cdot n_p}$;
end if
end if
end for
for each parameters-pair $p \in P$ **do**
for each parameters-pair $q \in P$ **do**
if $p = q$ **then**
$w_{p,q} = 0$;
else
$w_{p,q} = \frac{P_c^{(p)}}{\sum_{i=1}^{N} P_c^{(i)}}$
end if
end for
end for
for each parameters-pair $s_p \in S_{op}$ **do**
if each objective are smallest and crowding distance are largest **then**
▷ Choose initial best
Save this parameter-pair s_p as new best value and its related information $(P_{best} = [r_r^{(s_p)}, r_m^{(s_p)}]$, $\Phi_{(np)})$;
end if
Save S_{sp} as new parents;
end for
while generation number $g < g_{lim}$ **do**
Chose n_c crossover mate randomly with weight $w_{p,q}$, and crossover; ▷ Crossover
Generate full offspring genetic pool S_{op}
for $i = 1$ **to** 2 **do** ▷ Variation
Random choose variation individuals in genetic pool for i-th parameters;
Random variate, generate new genetic pool after variation S_{opv};
S_{op}=S_{opv};
end for
Randomly generated extra genetic pool E;
$S_{op} = S_{op} \cup P \cup E$;
for each parameters-pair $s_p \in S_{op}$ **do** ▷ Choose global best
$F_{fit}(s_p) = 0$;
Simulate MUAPs using parameters in P and generate $L^{(i)}(p)$;
for each objective O **do**
$F_{fit}(s_p) = F_{fit}(s_p) + L^{(i)}(s_p)$;
end for
end for
for each parameters-pair $s_p \in S_{op}$ **do**
if $n_p(s_p) \geq n_l$ **then**
$P_c^{(s_p)} = 0.01$;
else
if p is the boundary point in this Pareto frontier level **then**
$P_c^{(s_p)} = 0.1$

else
$P_c^{(s_p)} = \frac{1}{F_{fit}(s_p) \cdot n_p}$;
end if
end if
end for
for each parameters-pair $s_p \in S_{op}$ **do**
$w_{s_p} = \frac{P_c^{(s_p)}}{\sum_{i=1}^{N} P_c^{(i)}}$
end for
Selecting survival offspring S_{sp} from S_{op} randomly under considerate of weight w_{s_p};
▷ Natural Selection
non_dim_sort(S_sp);
crowd_dis_gen(S_sp);
for each parameters-pair $s_p \in S_{op}$ **do**
if $F_{fit}(s_p) = 0$ **then**
Save this parameter-pair s_p as new best value and its related information ($P_{best} = [r_r^{(s_p)}, r_m^{(s_p)}]$, $\Phi_{(np)}$);
else
if each objective are smallest and crowding distance are largest **then**
Save this parameter-pair s_p as new best value and its related information ($P_{best} = [r_r^{(s_p)}, r_m^{(s_p)}]$, $\Phi_{(np)}$);
end if
Save S_{sp} as new parents;
end if
end for
end while

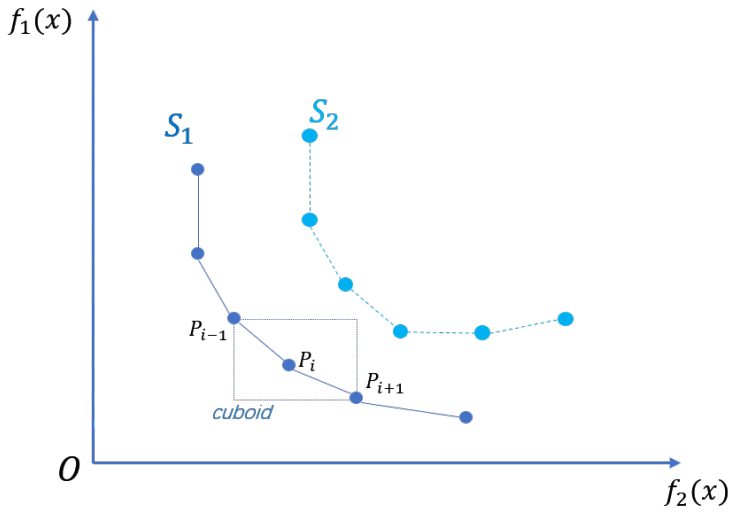


Figure 2.5: The method to confirm the crowding distance [93] P_i is the point needed to calculate the crowding distance. This is calculated by taking the difference of the objective value of the nearest two points P_{i-1} and P_{i+1} summing them and dividing the difference of the maximum value and the minimum value of the objective function in the same front separately.

2.3.3 Result

2.3.3.1 The Result of the Single Objective Genetic Algorithm

Figure 2.6 shows the result for a single MUAP. Due to the limitations of the computational equipment, 25 generations were used and the population for each parent was 50. The variation rate was 0.1, and each time we added 30 new parameter pairs to the full offspring genetic pool. As shown in figure 2.1, it is well simulated to the real MUAP.

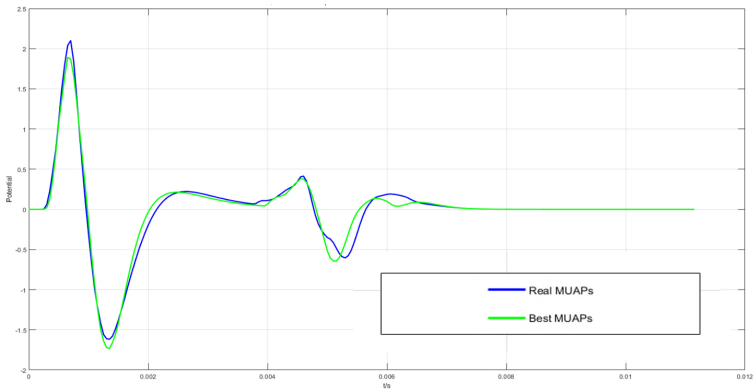


Figure 2.6: Result of Optimizing Motor Unit Territories and Muscle Fibre Innervation via Recruitment Threshold Parameters for a Single MUAP The blue line indicates the real MUAPs and the green line shows the simulated MUAP after optimization. The result was obtained at the range between the largest and the smallest recruitment threshold $r_r = 90$, and the maximum RT $r_m = 0.5$. The loss function we obtained was equal to 1.43

2.3.3.2 The Result of the Multi-objective Genetic Algorithm

The parameter settings used were similar to the settings indicated above. However, due to the limitations of the equipment, more than two objectives would have cost too much time to optimize to a satisfying value. We tested this optimization model using two MUAPs in a detectable area. Figure 2.7 shows the result.

2.4 Complete Optimization Model

2.4.1 Integrated Optimization Model

2.4.1.1 Methodology

In this part, we combined the above two parts since the optimization of motor unit territories and muscle fibre innervation via recruitment thresholds parameters including all procedures optimizing the electrode position (the generation of MUAPs from SFAPs). Thus, the genetic algorithm can be applied here continually.

The only differences between these two methods are the crossover and variation methods. Since the above GA only optimize 2 parameters, so we randomly choose individuals in the genetic pool to variate or exchange one parameter in the crossover. In this part, the variation is the same to use the single point variation method. However, it will randomly insert an exchange site, so that

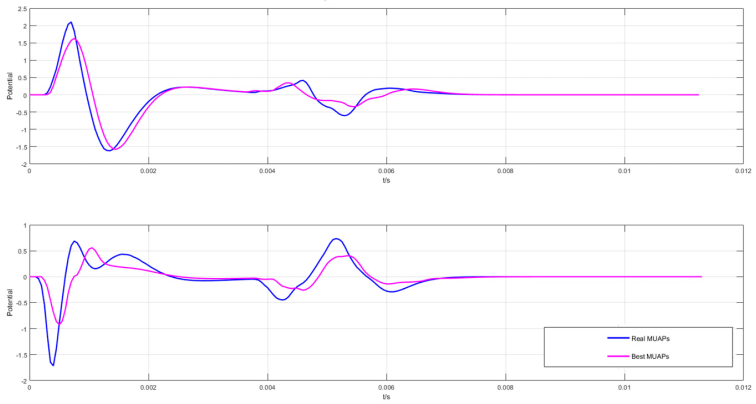


Figure 2.7: Result for Optimization of Motor Unit Territories and Muscle Fibre Innervation via Recruitment Threshold Parameters for double MUAPs The blue line shows the real MUAPs and the magenta line shows the simulated MUAP after optimization. The result was obtained at the range between the largest and smallest RT $r_r = 90.5$, and the maximum RT $r_m = 0.67$. The loss function we obtained was equal to 3.6 for the first MU and 8.4 for the second MU.

the parameters before that will keep, and the parameters after that will exchange with the same part of a crossover mate which has the same exchange site. The following figure 2.8 illustrates how this method operating.

Result

Figure 2.9 shows the result of the integrated optimization model, where the blue line is the real MUAP and the magenta line is the simulated MUAP after optimization. This optimization model is less accurate than the above optimization due to the larger number of parameters. Here, we set the simulation to 25 generations and the population for each parent was 50. The variation rate was 0.15 and two MUAPs were tested in a detectable area.

2.4.2 Simplification of the Optimization Model

Methodology

Although the genetic algorithm can partially improve the searching efficiency compared to the enumeration method, it still costs too much time to optimize especially when a MU controls a large number of MFs since the genetic algorithm only compares simulated MUAP for corresponding real MUAP which has the same MU position. However, since the generation of MF and MN centres use the FPS algorithm, which is quasi-randomly distributed, the MF and MN can occur at any point in the muscle section. This means that we can compare the real MUAP with all simulated MUAPs one by one to find an appropriate MUAP. Furthermore, to further simplify this model, the radius distance between the electrode and MFs will be applied directly instead of applying the position of the electrode and MFs. Theoretically, this algorithm can obtain infinite groups of solutions by applying several conditions to screen these solutions and selecting a solution that best enough. By repeating these procedures on each channel of the simulator, manually selecting the overlapping MUs in each channel and confirming their position, we can attain a solution set that contains all the detectable MUs. This procedure can be represented as the following algorithm 4. Note that since the conduction velocity was different in the radial and

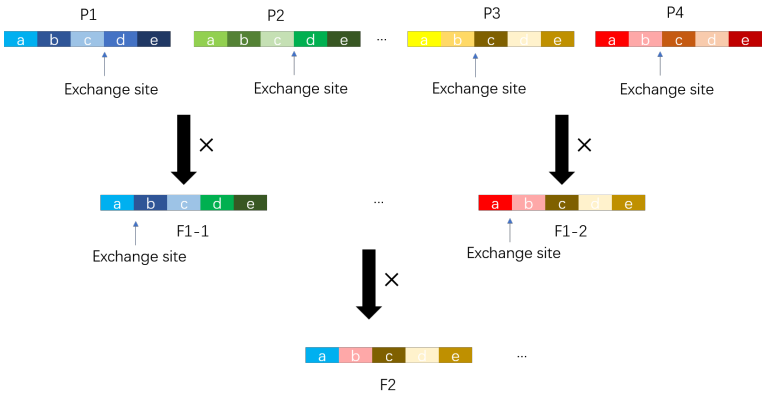


Figure 2.8: The Random Entirety Exchange Methods As shown here, $P1$ and $P2$ is exchanged after the third parameter, so their offspring $F1-1$ acquire a combined chromosome containing partial information from two parents, which is the parameters a,b and c from $P1$ and parameters d and e from $P2$. For $P3$ and $P4$, since the exchange site is after the second parameter, their offspring $F1-2$ acquire the parameters a and b from $P4$ and parameters c, d and e from $P3$. These two offspring then crossover with the exchange site after the first parameter, and the final offspring $F2$ ontains information from 3 individuals in parent (since $F1-1$ gives the parameter a which is obtained from $P1$ and $F1-2$ gives parameters b, c, d and e where b is obtained from $P4$ and the other parameters are from $P3$).

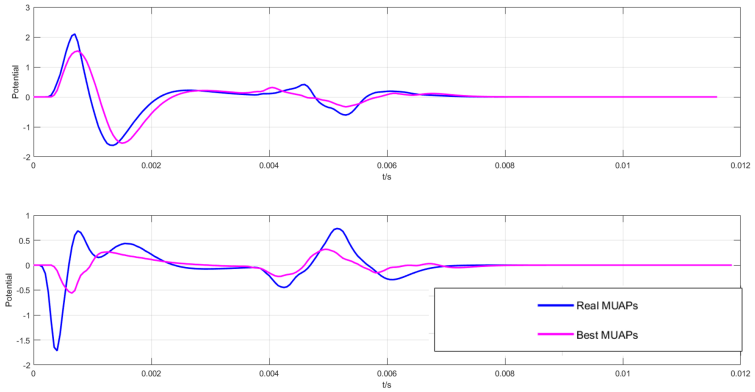


Figure 2.9: Result for Optimizing the Integrated Above Two Parts in 2.3 The blue line indicates the real MUAPs and the magenta line shows the simulated MUAPs after optimization. The result was obtained at the range between the largest and smallest RT $r_r = 79.2$, and the maximum RT $r_m = 0.7$. The loss function was equal to 3.54 for the first MU and 12.4 for the second MU.

axial, it was hard to optimize the z-coordinate in terms of the distance between the electrodes and MFs. Here, we fixed it at the half-length of the total muscle. Furthermore, we simplified the distances from an electrode to a MF to be the distances between the electrode and the MU for the MFs innervated by the MU.

Algorithm 4 Simplified Optimization Model

Input:

Randomly generated parent genetic pool P
The set for real MUAP Y
Limitation of numbers of solution satisfied s_{lim}
Limitation for each objective function f_{lim}
Limitation of assignments generate for each parameter vector a_{lim}

Output:

Parameter vector which require the searching conditions (r_r, r_m, x_p, y_p) r_r and r_t for RT parameters and x_p, y_p for electrode position parameters

```
Generate parameters P randomly
n_ps = 0
while all objectives' possible solution number n_ps ≤ s_lim do
    Generate some random radial distance R between electrodes and MU center
    n_ps = n_ps + 1
    for each p ∈ P do
        for each r ∈ R do
            while assgniment generated number n_as ≤ a_lim do
                Generate a MF-to-MN assignment
                for each y ∈ Y do
                    Use the MF-to-MN assignment generated to calculate Loss function L(y, p)
                  between y and p;
                    if L(y, p) ≤ f_lim then
                        save p-th parameters vector and the MUAP it generates;
                        n_as = n_as + 1;
                    else
                        continue;
                    end if
                end for
            end while
        end for
    end for
end while
```

Result

As you can see, the following figure 2.10 shows the results that the solution set obtained for two MUAPs in a detectable set. here we choose 20 parameter vectors in each group and repeating make MFs to MN assignment 300 times for a single parameter vector.

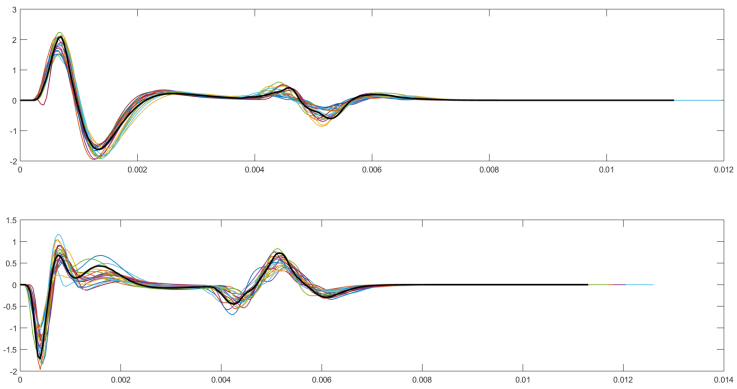


Figure 2.10: The Result of the Simplified Optimization Model The thinner, colourful lines are the simulated MUAPs we selected that are close to real MUAP, and the black line is the real MUAP.

Chapter 3

Discussion

3.1 Evaluation

The optimization model for optimizing electrodes matches the real experimental MUAPs since no stochastic processes were included. The other part of the optimization included a stochastic process; thus, we can generate a MUAP that is similar to real data even while using a genetic algorithm. A greater population of the parent genetic pools for each iteration produces better matching results.

The two complete models in 2.4, as it shows, the first model used genetic algorithm has the optimization result that it is less accurate than the optimization result in RT parameters optimization in 2.3 since it is included more parameters. Using the simplified method, we can find many MUAPs similar to the two MUAPs. This is because many MUs innervate similar numbers of MFs; as a result, there are more possible assignments that can approach the real data. However, since the simplified model used the enumeration method, the optimization procedure still costed too much time.

3.2 Improvement

The project still requires some improvement. In terms of optimizing the electrode position, while the ADAM method can converge at a very small loss function, there are still some shortcomings since it takes too much time to converge. One possible method to further improve this method is to use the Shampoo algorithm, which has been widely tested in machine learning and can greatly improve the optimization speed [94],[95]. The following equation (3.1) is the iteration equation of the Shampoo algorithm[94]. Here the $\boldsymbol{g}$ is the gradient of the loss function. As shown here, the inverse and root of $\boldsymbol{R}$ and $\boldsymbol{L}$ needs to be calculated, which can be achieved by singular value decomposition (SVD) or the Schur-Newton algorithms [96]. Since this requires large quantities of calculation, the differences between the other methods and the Shampoo algorithm will be more significant in the multi-GPU calculation [95],[97].

Update pre-conditioners:

$$\boldsymbol{L_{(n)}} = \boldsymbol{L_{(n-1)}} + \boldsymbol{g_n} \cdot \boldsymbol{g_n^T} \tag{3.1a}$$

$$\boldsymbol{R_{(n)}} = \boldsymbol{R_{(n-1)}} + \boldsymbol{g_n^T} \cdot \boldsymbol{g_n} \tag{3.1b}$$

Update parameter:

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \eta \boldsymbol{L}_{(n)}^{-\frac{1}{4}} \boldsymbol{g}_n^T \boldsymbol{R}_{(n)}^{-\frac{1}{4}} \tag{3.1c}$$

Another improvement is in the second part of 2.3, which is the optimization of motor unit territories and muscle fibre innervation via recruitment threshold parameters. Since this part and the total completed optimization model are all based on genetic algorithms as that in 2.4.1, these two parts are similarly improved. Since the procedure of assigning MF to MU costs too much time on a personal PC, we can only generate random parameters in a small batch. Thus, parallel computation is a way to significantly improve the computation speed, generating a large population for parameters. Figure 3.1 shows the design of parallel computing for these two parts . To avoid exponential explosion growth, the natural selection mechanism will still be applied but have larger population in each iteration here.

Similarly, the simplified model can also perform parallel computing. Figure 3.2 shows the design of parallel computing. The design for single-channel optimization involves assigning different parameter vectors to different GPUs and then assigning a MF to a MN many times by using the parameters assigned. For multi-channel optimization, a parallel computing structure same as the one we designed will be used.

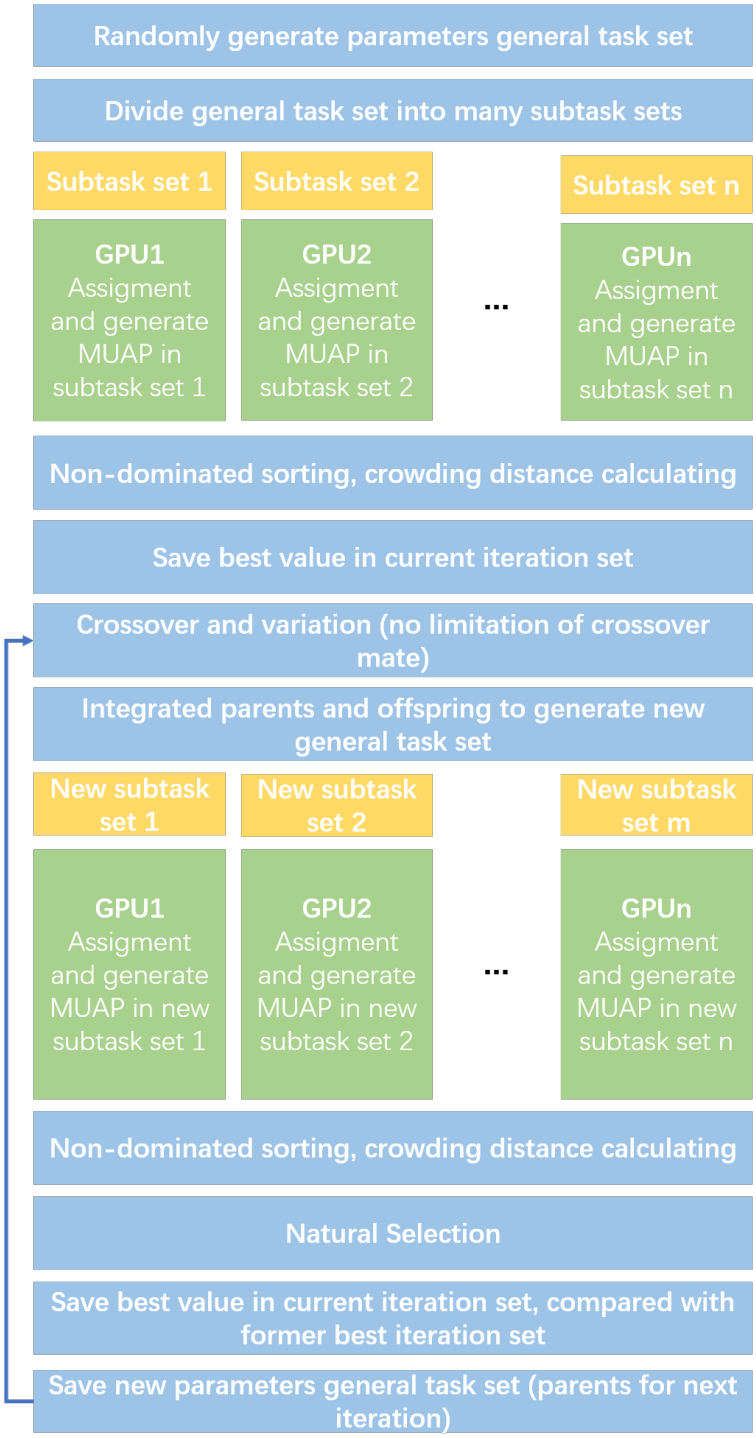


Figure 3.1: The Parallel Computation Design for the Genetic Algorithm The procedure in blue boxes will complete by the CPU, and the procedure in the green boxes will complete by the multi-GPU. It can no longer to limit the crossover mate for each individual. The procedure before crossover is to generate the initial parents, sorting according to non-dominated level and calculating the crowding distance.

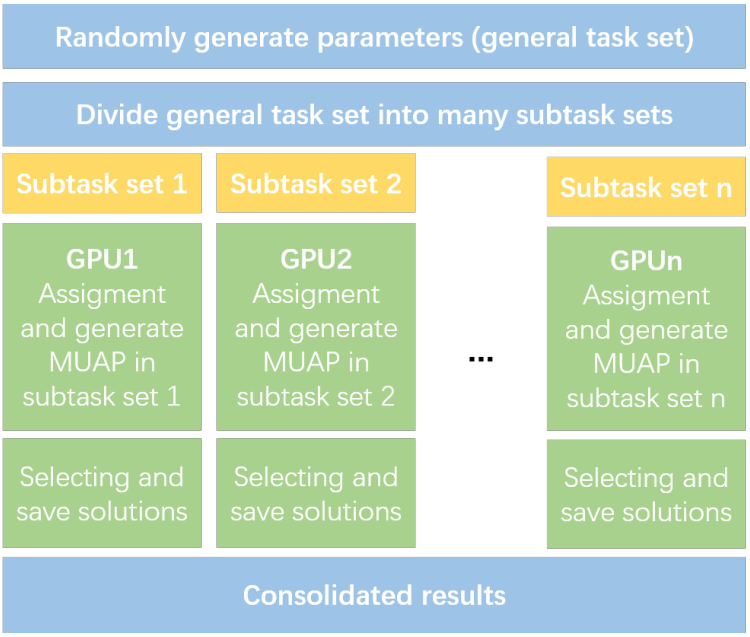


Figure 3.2: The Parallel Computation Design for a Simplified Optimization Model The procedure in the blue boxes will be finished by the CPU, and the procedure in the green boxes will be finished by the multi-GPU.

Chapter 4

Conclusion

The optimization result showed that the simulated MUAP best matches the real experimental data. The authenticity of the simulator can be demonstrated since some simulated MUAP can be found that are similar to real MUAPs (with a loss function smaller than 5). However, the second part in 2.3 and the total complete model uses genetic algorithms in 2.4.1 that are less accurate than using our simplified model in 2.4.2. This is because of the high computation volume and the limitations of the equipment we used.

To improve the accuracy, parallel computing is recommended since it can hugely improve the computation speed by assigning MF to MN for different parameters by applying multi general-purpose GPU (GPGPU) calculations or multi-CPU calculations simultaneously. A possible design for parallel computing is provided above.

Bibliography

[1] E. Kandel, S. Mack, T. Jessell, J. Schwartz, S. Siegelbaum, and A. Hudspeth, *Principles of Neural Science, Fifth Edition*, ser. McGraw-Hill's AccessMedicine. McGraw-Hill Education, 2013.

[2] R. E. Burke, *Motor Units: Anatomy, Physiology, and Functional Organization.* American Cancer Society, 2011, pp. 345–422. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cphy.cp010210

[3] C. Heckman and R. M. Enoka, "Motor unit," *Comprehensive physiology*, vol. 2, no. 4, pp. 2629–2682, 2012.

[4] K. Akhmadeev, "Probabilistic models based on emg decomposition for prosthetic control," Ph.D. dissertation, University of Nantes, 11 2019.

[5] M. Bear, B. Connors, and M. Paradiso, "Spinal control of movement," in *Neuroscience: Exploring the Brain, Enhanced Edition: Exploring the Brain, Enhanced Edition*, M. Bear, B. Connors, and M. Paradiso, Eds. London: Jones & Bartlett Learning, 2020, pp. 453–482.

[6] P. M. R. Cruz, J. Cossins, D. Beeson, and A. Vincent, "The neuromuscular junction in health and disease: molecular mechanisms governing synaptic formation and homeostasis," *Frontiers in Molecular Neuroscience*, vol. 13, 2020.

[7] F. Buchthal and H. Schmalbruch, "Motor unit of mammalian muscle." *Physiological reviews*, vol. 60, no. 1, pp. 90–142, 1980.

[8] I. B. Levitan, I. B. Levitan, L. K. Kaczmarek *et al.*, *The neuron: cell and molecular biology.* Oxford University Press, USA, 2002.

[9] I. Rodriguez-Carreno, L. Gila-Useros, and A. Malanda-Trigueros, "Motor unit action potential duration: measurement and significance," in *Advances in clinical neurophysiology.* IntechOpen, 2012.

[10] P. Dayan and L. Abbott, "Neural encoding i: firing rates and spike statistics," *Theoretical neuroscience: computational and mathematical modeling of neural systems. MIT Press, Cambridge, MA*, pp. 1–38, 2001.

[11] W. van Drongelen, "Spike tain analysis," in *Signal Processing for Neuroscientists*, W. van Drongelen, Ed. Burlington: Academic Press, 2007, pp. 219–243. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123708670500012

[12] J. P. T. Ward and R. W. A. Linden, "Motor unit: Recruiment and summation," in *Physiology at a Glance 4th ed.*, J. P. T. Ward and R. W. A. Linden, Eds. London: Wiley Blackwell, 2017, pp. 36–37.

[13] A. J. Fuglevand, D. A. Winter, and A. E. Patla, "Models of recruitment and rate coding organization in motor-unit pools," *Journal of Neurophysiology*, vol. 70, no. 6, pp. 2470–2488, 1993, pMID: 8120594. [Online]. Available: https://doi.org/10.1152/jn.1993.70.6.2470

[14] T. Buchanan, D. Lloyd, K. Manal, and T. Besier, "Neuromusculoskeletal modeling: Estimation of muscle forces and joint moments and movements from measurements from neural command," *Journal of Applied Biomechanics*, vol. 20, no. 4, pp. 367–395, 2004.

[15] "Velocity and Duration of Muscle Contraction," 8 2020, [Online; accessed 2021-08-21]. [Online]. Available: https://med.libretexts.org/@go/page/7534

[16] W. Gerstner, A. K. Kreiter, H. Markram, and A. V. M. Herz, "Neural codes: Firing rates and beyond," *Proceedings of the National Academy of Sciences*, vol. 94, no. 24, pp. 12 740–12 741, 1997. [Online]. Available: https://www.pnas.org/content/94/24/12740

[17] C. J. De Luca and P. Contessa, "Hierarchical control of motor units in voluntary contractions," *Journal of neurophysiology*, vol. 107, no. 1, pp. 178–195, 2012.

[18] "Control of Muscle Tension," 8 2020, [Online; accessed 2021-08-21]. [Online]. Available: https://bio.libretexts.org/@go/page/14011

[19] R. Person and L. Kudina, "Discharge frequency and discharge pattern of human motor units during voluntary contraction of muscle," *Electroencephalography and Clinical Neurophysiology*, vol. 32, no. 5, pp. 471–483, 1972. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0013469472900582

[20] E. Henneman, G. Somjen, and D. O. Carpenter, "Functional significance of cell size in spinal motoneurons," *Journal of Neurophysiology*, vol. 28, no. 3, pp. 560–580, 1965, pMID: 14328454. [Online]. Available: https://doi.org/10.1152/jn.1965.28.3.560

[21] G. Kamen, "Electromyographic kinesiology," in *Research Methods in Biomechanics 2rd ed.*, G. Kamen, Ed. Champaign, IL: Human Kinetics Publisher, 2014, pp. 179–202.

[22] D. Farina, D. F. Stegeman, and R. Merletti, *Biophysics of the Generation of EMG Signals*. John Wiley Sons, Ltd, 2016, ch. 2, pp. 1–24. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119082934.ch02

[23] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin, "Techniques of emg signal analysis: detection, processing, classification and applications," *Biological procedures online*, vol. 8, no. 1, pp. 11–35, 2006.

[24] D. F. Stegeman, J. H. Blok, H. J. Hermens, and K. Roeleveld, "Surface emg models: properties and applications," *Journal of Electromyography and Kinesiology*, vol. 10, no. 5, pp. 313–326, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1050641100000237

[25] L. Mesin, R. Merletti, and A. Rainoldi, "Surface emg: The issue of electrode location," *Journal of Electromyography and Kinesiology*, vol. 19, no. 5, pp. 719–726, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1050641108001181

[26] C. J. De Luca, A. Adam, R. Wotiz, L. D. Gilmore, and S. H. Nawab, "Decomposition of surface emg signals," *Journal of Neurophysiology*, vol. 96, no. 3, pp. 1646–1657, 2006, pMID: 16899649. [Online]. Available: https://doi.org/10.1152/jn.00009.2006

[27] A. Peter, E. Andersson, A. Hegyi, T. Finni, O. Tarassova, N. Cronin, H. Grundstrom, and A. Arndt, "Comparing surface and fine-wire electromyography activity of lower leg muscles at different walking speeds," *Frontiers in Physiology*, vol. 10, p. 1283, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphys.2019.01283

[28] E. Petersen and P. Rostalski, "A comprehensive mathematical model of motor unit pool organization, surface electromyography, and force generation," *Frontiers in Physiology*, vol. 10, p. 176, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphys.2019.00176

[29] R. Merletti and D. Farina, "Analysis of intramuscular electromyogram signals," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1887, pp. 357–368, 2009.

[30] R. F. Weir, P. R. Troyk, G. A. DeMichele, D. A. Kerns, J. F. Schorsch, and H. Maas, "Implantable myoelectric sensors (imess) for intramuscular electromyogram recording," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 1, pp. 159–171, 2009.

[31] S. Jose, S. T. George, M. S. P. Subathra, V. S. Handiru, P. K. Jeevanandam, U. Amato, and E. S. Suviseshamuthu, "Robust classification of intramuscular emg signals to aid the diagnosis of neuromuscular disorders," *IEEE Open Journal of Engineering in Medicine and Biology*, vol. 1, pp. 235–242, 2020.

[32] C. J. De Luca, "Reflections on emg signal decomposition," pp. 33–37, 1989.

[33] J. Fang, G. C. Agarwal, and B. T. Shahani, "Decomposition of multiunit electromyographic signals," *IEEE transactions on biomedical engineering*, vol. 46, no. 6, pp. 685–697, 1999.

[34] A. Gerber, R. M. Studer, R. J. De Figueiredo, and G. S. Moschytz, "A new framework and computer program for quantitative emg signal analysis," *IEEE transactions on biomedical engineering*, no. 12, pp. 857–863, 1984.

[35] P. Guiheneuc, C. Doncarli, and M. Le Bastard, "Concomitant multitrain automatic analysis of motor unit potentials and macroemg," *Computer-aided electromyography and expert systems*, pp. 83–90, 1989.

[36] R. Gut and G. S. Moschytz, "High-precision emg signal decomposition using communication techniques," *IEEE transactions on signal processing*, vol. 48, no. 9, pp. 2487–2494, 2000.

[37] W. Haas and M. Meyer, "An automatic emg decomposition system for routine clinical examination and clinical research—artmup," in *Computer-Aided Electromyography and Expert Systems*. Elsevier, 1989, pp. 67–81.

[38] R. Kadefors, "Recruitment of low threshold motor-units in the trapezius muscle in different static arm positions," *Ergonomics*, vol. 42, no. 2, pp. 359–375, 1999.

[39] R. S. LeFever and C. J. De Luca, "A procedure for decomposing the myoelectric signal into its constituent action potentials-part i: technique, theory, and implementation," *IEEE transactions on biomedical engineering*, no. 3, pp. 149–157, 1982.

[40] R. S. LeFever, A. P. Xenakis, and C. J. De Luca, "A procedure for decomposing the myoelectric signal into its constituent action potentials-part ii: execution and test for accuracy," *IEEE transactions on biomedical engineering*, no. 3, pp. 158–164, 1982.

[41] K. C. McGill, K. L. Cummins, and L. J. Dorfman, "Automatic decomposition of the clinical electromyogram," *IEEE Transactions on Biomedical Engineering*, no. 7, pp. 470–477, 1985.

[42] K. McGill and L. Dorfman, "Automatic decomposition electromyography (ademg): methodologic and technical considerations," pp. 91–101, 1989.

[43] D. Stashuk and C. De Luca, "Update on the decomposition and analysis of emg signals," *Computer-aided electromyography and expert systems*, pp. 39–53, 1989.

[44] P. Wellig and G. S. Moschytz, "Analysis of wavelet features for myoelectric signal classification," in *1998 IEEE International Conference on Electronics, Circuits and Systems. Surfing the Waves of Science and Technology (Cat. No. 98EX196)*, vol. 3. IEEE, 1998, pp. 109–112.

[45] Wellig, Peter and Moschytz, George S, "Electromyogram decomposition using the single-lineage clustering algorithm and wavelets," in *ICECS'99. Proceedings of ICECS'99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No. 99EX357)*, vol. 1. IEEE, 1999, pp. 537–540.

[46] C. De Luca, R. LeFever, M. McCue, and A. Xenakis, "Behaviour of human motor units in different muscles during linearly varying contractions," *The Journal of physiology*, vol. 329, no. 1, pp. 113–128, 1982.

[47] De Luca, CJ and LeFever, RS and McCue, MP and Xenakis, AP, "Control scheme governing concurrently active human motor units during voluntary contractions," *The Journal of physiology*, vol. 329, no. 1, pp. 129–142, 1982.

[48] D. Stashuk, "Emg signal decomposition: how can it be accomplished and used?" *Journal of Electromyography and Kinesiology*, vol. 11, no. 3, pp. 151–173, 2001.

[49] T. Yu, K. Akhmadeev, E. Le Carpentier, Y. Aoustin, and D. Farina, "On-line recursive decomposition of intramuscular emg signals using gpu-implemented bayesian filtering," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 6, pp. 1806–1818, 2019.

[50] T. Yu, K. Akhmadeev, É. Le Carpentier, Y. Aoustin, R. Gross, Y. Péréon, and D. Farina, "Recursive decomposition of electromyographic signals with a varying number of active sources: Bayesian modeling and filtering," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 2, pp. 428–440, 2019.

[51] D. Farina, A. Crosetti, and R. Merletti, "A model for the generation of synthetic intramuscular emg signals to test decomposition algorithms," *IEEE transactions on biomedical engineering*, vol. 48, no. 1, pp. 66–77, 2001.

[52] A. Konstantin, T. Yu, E. Le Carpentier, Y. Aoustin, and D. Farina, "Simulation of motor unit action potential recordings from intramuscular multichannel scanning electrodes," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 7, pp. 2005–2014, 2019.

[53] E. Henneman, G. Somjen, and D. O. Carpenter, "Functional significance of cell size in spinal motoneurons," *Journal of neurophysiology*, vol. 28, no. 3, pp. 560–580, 1965.

[54] S. Nandedkar, E. St *et al.*, "Simulation of single muscle fibre action potentials," *Medical and Biological Engineering and Computing*, vol. 21, no. 2, pp. 158–165, 1983.

[55] E. Petersen and P. Rostalski, "A comprehensive mathematical model of surface electromyography and force generation," *bioRxiv*, p. 273458, 2018.

[56] D. Farina and A. Rainoldi, "Compensation of the effect of sub-cutaneous tissue layers on surface emg: a simulation study," *Medical engineering & physics*, vol. 21, no. 6-7, pp. 487–497, 1999.

[57] A. Hamilton-Wright and D. W. Stashuk, "Physiologically based simulation of clinical emg signals," *IEEE Transactions on biomedical engineering*, vol. 52, no. 2, pp. 171–183, 2005.

[58] A. J. Fuglevand, D. A. Winter, and A. E. Patla, "Models of recruitment and rate coding organization in motor-unit pools," *Journal of neurophysiology*, vol. 70, no. 6, pp. 2470–2488, 1993.

[59] V. Carriou, J. Laforêt, S. Boudaoud, and M. Al Harrach, "Realistic motor unit placement in a cylindrical hd-semg generation model," in *2016 38th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE, 2016, pp. 1704–1707.

[60] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[61] D. P. Mitchell, "Generating antialiased images at low sampling densities," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 65–72.

[62] G. Peyre, "Surface sampling," in *Numerical Mech Processing*, 2008, pp. 36–42.

[63] P. Kamousi, S. Lazard, A. Maheshwari, and S. Wuhrer, "Analysis of farthest point sampling for approximating geodesics in a graph," *Computational Geometry*, vol. 57, pp. 1–7, 2016.

[64] J. Navallas, A. Malanda, L. Gila, J. Rodriguez, and I. Rodriguez, "Comparative evaluation of motor unit architecture models," *Medical & biological engineering & computing*, vol. 47, no. 11, pp. 1131–1142, 2009.

[65] J. Navallas, A. Malanda, L. Gila, J. Rodríguez, and I. Rodríguez, "A muscle architecture model offering control over motor unit fiber density distributions," *Medical & biological engineering & computing*, vol. 48, no. 9, pp. 875–886, 2010.

[66] P. Rosenfalck, "Intra-and extracellular potential fields of active nerve and muscle fibres," *Acta Phisiologica Scandinavica*, vol. 321, 1969.

[67] D. Farina and R. Merletti, "A novel approach for precise simulation of the emg signal detected by surface electrodes," *IEEE transactions on biomedical engineering*, vol. 48, no. 6, pp. 637–646, 2001.

[68] R. Plonsey, "The active fiber in a volume conductor," *IEEE transactions on biomedical engineering*, no. 5, pp. 371–381, 1974.

[69] E. Stalberg and J. V. Trontelj, "The study of normal and abnormal neuromuscular transmission with single fibre electromyography," *Journal of neuroscience methods*, vol. 74, no. 2, pp. 145–154, 1997.

[70] J. L. Dideriksen, D. Farina, and R. M. Enoka, "Influence of fatigue on the simulated relation between the amplitude of the surface electromyogram and muscle force," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1920, pp. 2765–2781, 2010.

[71] P. Contessa and C. J. D. Luca, "Neural control of muscle force: indications from a simulation model," *Journal of neurophysiology*, vol. 109, no. 6, pp. 1548–1570, 2013.

[72] E. Petersen and P. Rostalski, "A comprehensive mathematical model of motor unit pool organization, surface electromyography, and force generation," *Frontiers in physiology*, vol. 10, p. 176, 2019.

[73] B. Feinstein, B. Lindegård, E. Nyman, and G. Wohlfart, "Morphologic studies of motor units in normal human muscles," *Cells Tissues Organs*, vol. 23, no. 2, pp. 127–142, 1955.

[74] A. Wald, "Statistical decision functions." 1950.

[75] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for optimization*. Mit Press, 2019.

[76] J. C. Meza, "Steepest descent," *WIREs Computational Statistics*, vol. 2, no. 6, pp. 719–722, 2010. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.117

[77] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[78] L. W. Ehrlich, "A modified newton method for polynomials," *Communications of the ACM*, vol. 10, no. 2, pp. 107–108, 1967.

[79] B. Peacock, T. Westers, S. Walsh, and K. Nicholson, "Feedback and maximum voluntary contraction," *Ergonomics*, vol. 24, no. 3, pp. 223–228, 1981.

[80] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[81] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*. Springer, 1998, pp. 2093–2229.

[82] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[83] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[84] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical science*, vol. 8, no. 1, pp. 10–15, 1993.

[85] F. Glover, "Tabu search—part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[86] Glover, Fred, "Tabu search—part ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.

[87] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *IEEE Circuits and Devices magazine*, vol. 5, no. 1, pp. 19–26, 1989.

[88] K. Deb, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.

[89] M. Caramia and P. Dell'Olmo, "Multi-objective optimization," in *Multi-objective management in freight logistics*. Springer, 2020, pp. 21–51.

[90] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability engineering & system safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[91] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[92] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*. IEEE, 2005, pp. 84–91.

[93] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[94] V. Gupta, T. Koren, and Y. Singer, "Shampoo: Preconditioned stochastic tensor optimization," 2018.

[95] R. Anil, V. Gupta, T. Koren, K. Regan, and Y. Singer, "Scalable second order optimization for deep learning," 2021.

[96] C.-H. Guo and N. J. Higham, "A schur–newton method for the matrix\boldmath p th root and its inverse," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 788–804, 2006.

[97] R. Anil, V. Gupta, T. Koren, K. Regan, and Y. Singer, "Second order optimization made practical," *arXiv preprint arXiv:2002.09018*, 2020.