Particle Swarm Optimizer Variations for Multiple Sequence Alignment

by

Nicholas Aksamit

Department of Computer Science

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

To both my parents and Emily, in honour of their support.

Abstract

Four variants of the particle swarm optimizer (PSO), namely binary, angular modulated, and two multi-guided, are applied to the multiple sequence alignment (MSA) problem. The latter two are a combination of both angular modulated and binary with the multi-guided PSO. Two forms of the MSA problem are employed: single-objective, which uses a weight aggregated function, and multi-objective, which applies each individual objective function separately. Solution qualities are measured by the number of aligned characters, and number of trailing spaces. Introspective study of each PSO algorithm is explored to determine the affect of parameters on sequence alignment outcome, followed by inter-algorithm comparisons. Concluding are contrasts against a state-of-the-art algorithm named ProbCons, which is only done with the angular modulated variants due to infeasibility of the binary PSOs.

Acknowledgments

Gratitude is recognized for those who have assisted in the creation, and completion of this work. Without them the quality would have certainly perished.

- Professor Beatrice M. Ombuki-Berman, for supervision throughout the entire research process, and for consistently pushing me so as to never remain stagnant.
- Pawel Jocko, for solutions to queries about the multi-guided particle swarm optimizer.
- Tyler Crane, for constructive criticism on previous works of mine that included writing.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The desire to solve complex problems is a natural phenomenon, of which many hours could be spent in the process. Some of these problems also do not have a known way of being solved in a timely fashion, and where the number of inputs increase linearly, the amount of steps needed to solve it may rise exponentially. For such complications which must be solved in a timely fashion, or naturally have a large input, faster methods need be used. A certain class of problems which are not yet known to be solved in a polynomial amount of operations is denoted as NP-hard [1].

Thus, the way to produce results in a sufficient manner is seemingly to either prove that one of these NP-hard problems may be completed in a polynomial number of operations, or to use other techniques in an attempt of finding a solution. The Multiple Sequence Alignment (MSA) problem will be studied, which is known as being NP-complete [2]. Because of its NP-completeness, it is NP-hard, but can be verified with a suitable polynomial time algorithm. This alone helps reduce the complexity of algorithms used in this work.

Optimizing problems is a means of finding the best solution possible. This may be seen as the search for a global maximum, or global minimum of some function. The function is then employed as a measure of how 'good' its respective outcome is. This allows for comparisons between different solutions, to which a highest or lowest output can be comparatively received. Some of the challenges with solving optimization problems come from their class of computational complexity, where the aforementioned NP-hard category struggles. This, in turn, is one reasoning behind studies of MSA.

Metaheuristics have been applied to a variety of problems with much success, and PSO [3] is one such algorithm. Although solutions can potentially be found in a quicker amount of time, it comes with the sacrifice of a guaranteed optimum. In

other words, speed is gained at the cost of precision. As was stated previously, MSA is an NP-complete problem, and since this is a subset of NP-hard, polynomial time algorithms have not yet been found. Whether they exist or not, it is desired that PSO provides a good deal of precision for the time that it reduces in computing an outcome.

The remainder of this chapter is distributed as follows: Section 1.1 describes the motivation behind this study. Following this, Section 1.2 gives an overview of swarm intelligence. Later, Section 1.3 lists the various objectives this study has completed, accompanied by Section 1.4, which illustrates an organization of the remaining work.

1.1 Motivation

The MSA problem is not new, nor unpopular, as a publication describing a methodology named CLUSTALW has 64,000 citations as of this writing [4]. Other known programs in the area include DIALIGN-TX [5], Kalign [6], MAFFT [7], MUSCLE [8], T-Coffee [9], and ProbCons [10]. In [11], comparisons are drawn between each of these listed algorithms. Among them, it is MAFFT (linsi), T-Coffee, and ProbCons that achieve best overall alignment quality over 218 selected benchmark references. After reviewing publications on various comparisons among MSA techniques, it becomes clear the lack of PSO involvement [12, 11, 13]. This may be due to it being an iterative approach of solving MSA, which as the name suggests, refines solutions throughout many iterations [13]. It is stated in [14] that iterative methods "lack speed and reliability".

There are still, however, many variants of PSO that seemingly have not been attempted for MSA. Some of these include AMPSO [15] and MGPSO [16], whereas BPSO [17] has already been featured in [18]. Although the attractiveness of PSO for MSA has been sluggish when compared with others, its many variants may give results that are competitive to the state-of-the-art algorithms, such as the ones that are mentioned previously. However, it should be noted that this work only provides comparisons between PSO variants, and not between PSO and other methods of solving MSA.

Nonetheless, not only will a single-objective MSA problem be used, but also multi-objective one. It is the hope that aggressive results can be obtained, in an attempt to express the power of PSO and its many variants.

1.2 Overview

A branch of evolutionary computation is swarm intelligence, which is defined as a collection of individuals that interact with each other, and adapt to their environment. Their control remains decentralized, with the independents expected to eventually organize themselves. From this, the ideology of swarm intelligence corresponds with the inadequacy of few, but power of plenty. Many such algorithms exist, such as Ant Colony Optimization, Artificial Bee Colony, and PSO [19]. PSO in particular has been applied to many areas, such as but not including: antenna design, combinatorial optimization, design of electricity networks, and detection, as well as recovery from faults [20].

Since its creation, a large number of variants have been produced, along with theoretical analyses [21]. From this emanates several that support a discrete search space, in particular: Integer and Categorical PSO [22], Integer PSO (PSO values rounded), and the aforementioned BPSO and AMPSO. While BPSO has already been used for MSA [18], AMPSO and MGPSO seemingly have not. As such, comparisons will be made between all of these PSO variants, in order to determine which of them provide superior results, if any.

As it stands, MSA can be represented as a multi-objective optimization problem, where it has already achieved success [23, 24, 25]. However MGPSO, which is an algorithm that by default does not support binary spaces, will require alterations. Thus two versions are created, denoted as B-MGPSO and AM-MGPSO, respectively naming a combination of MGPSO with BPSO and AMPSO. Unlike some of the previously referenced works, two objective functions will be utilized for MGPSO. In the singular-objective formation used by BPSO and AMPSO, the exact two objective functions are used, only they are instead formatted as a weighted aggregation function.

1.3 Objectives

In general, the focus of this study is to perform a comparative study of four unique PSO variants when applied to the MSA problem. In the path of achieving this, there are many minor objectives that are realized, such as:

- presenting a general overview for already existing MSA solving methods.
- an introduction to the reasoning behind concentration on PSO.

- investigation into the affects parameters have on algorithmic output.
- interpreting the rationality of certain introspective, and inter-algorithm disparities.
- comparing superior PSO variants with the state-of-the-art algorithm ProbCons.

1.4 Organization

The remaining chapters of this thesis have the following structure:

- Chapter 2 gives background information on topics that will be used in further chapters. This comprises of multi-objective optimization, MSA, PSO, and MGPSO, which also contain sub portions that explain related matters.
- Chapter 3 provides details on the experimentation process, which includes the formation of the problem to be solved, and the parameters used throughout all of the utilized PSO variants.
- Chapter 4 discusses the results that have been gathered from experimentation. Introspective, along with inter-algorithmic comparisons are performed not only to determine a superior set of parameters, but also algorithms. Following these PSO variant comparisons are additional contrasts made with ProbCons.
- Chapter 5 summarizes the work completed in this thesis, with overall observations stated. Discussion is also raised as to potential studies in the future involving MSA and PSO.
- Appendix A charts all of the sequence sets utilized for experimentation, along with information of their names, each of the sequence lengths, and the number of sequences in the set.
- Appendix B illustrates all of the full-page figures that are referenced throughout the text. In addition, information is provided that assists with understanding the graphs displayed. All graphs are associated in such a way as to not distract the reader throughout Chapter 4.
- Appendix C shows all of the acronyms used in this work, along with the full length word each acronym represents.

Chapter 2

Background

In order to take advantage of the results found in this study, there must be an understanding of various subject matters. MSA can be expressed as a multi-objective problem, which requires knowledge about multi-objective optimization in general. Thus, multi-objective optimization is the topic of first discussion in Section 2.1. Next, Section 2.2 elucidates the subject of MSA. Following this is Section 2.3, where PSO, along with AMPSO and BPSO are synopsized. Lastly, in Section 2.4, MGPSO, in conjunction with the variants B-MGPSO and AM-MGPSO are explained.

2.1 Multi-objective Optimization

MSA has the ability to be interpreted as a problem with multiple objective functions, each of which being optimizable. However, it is not the only one of its kind [26]. At times, these functions can be in conflict with one another. That is to say, after some threshold, where one increases, the other may decrease, or vice versa. If the motivation is to maximize both values, this is not an adequate situation. A multi-objective optimization problem can then be loosely represented as a set of optimal trade-off values. The various definitions that are listed in the subsequent subsections come from a list of publications discussing multi-objective optimization [27].

2.1.1 Multi-objective Problem

A Multi-objective Optimization Problem (MOOP) covers the optimization of multiple objective functions $f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_{n_m}(\mathbf{x})$, each of which measure some quality of an obtained solution. Upon optimization of the objective functions, better solutions should be obtained for the respective Multi-objective Problem (MOP). That is to

say, an MOOP optimizes the objective functions, which in turn should result in an improved outcome for the MOP.

The search space of an MOOP is represented as an n_x-dimensional $\mathcal{S} \subseteq \mathbb{R}^{n_x}$. After constraints are applied, the feasible search space $\mathcal{F}$ shrinks to $\mathcal{F} \subseteq \mathcal{S}$, but if no constraints are employed then $\mathcal{F} = \mathcal{S}$. Let the objective space be signified as the n_m-dimensional $\mathcal{O}$, where an MOOP optimizes a mapping $x \in \mathcal{F} \rightarrow y \in \mathcal{O}$. The previously mentioned mapping then has a sizing $n_x \rightarrow n_m$.

Below are a set of definitions to assist in further understanding of the topic. Note that for Definition 2.4, the duality principle states that because $\min f(\mathbf{x}) = \max -f(\mathbf{x})$, an MOOP could be expressed with either as maximization or minimization of $f(\mathbf{x})$.

Definition 2.1 ***Decision vector*** *A decision vector consists of selected numerical values which constitute a feasible solution to an MOP. It takes the form* $\mathbf{x} = [x_1, x_2, ..., x_{n_x}] \in \mathcal{F}$*, where* $x_1, x_2, ..., x_{n_x}$ *are denoted as decision variables.*

Definition 2.2 ***Objective vector*** *An objective vector consists of numerical objective function values that shape a potential outcome to the MOOP. It takes the form* $\mathbf{y} = [y_1, y_2, ..., y_{n_m}] \in \mathcal{O}$*, where* $y_1, y_2, ..., y_{n_m}$ *are denoted as objective variables.*

Definition 2.3 ***Objective Function*** *The objective function takes the mapping* $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$*, and defines some mathematical criterion of an optimization problem. It is denoted as* $f(\mathbf{x})$*, where* $\mathbf{x}$ *is the above-defined decision vector.*

Definition 2.4 ***Multi-objective optimization problem*** *Without loss of generality, an MOOP takes the mathematical form of*

$$\min f(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_{n_m}(\mathbf{x})] \tag{2.1}$$

where $\mathbf{x} \in \mathcal{F}$*, and* $f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_{n_m}(\mathbf{x})$ *are the objective functions. This new minimization function has a mapping of* $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_m}$*, such that the input* $\mathbf{x}$ *is the decision vector, and the output becomes what is previously defined as the objective vector.*

2.1.2 Pareto Efficiency

As mentioned in Section 2.1, an MOOP can be loosely represented as a set of optimal trade-off values. This becomes clear when given the definitions from below:

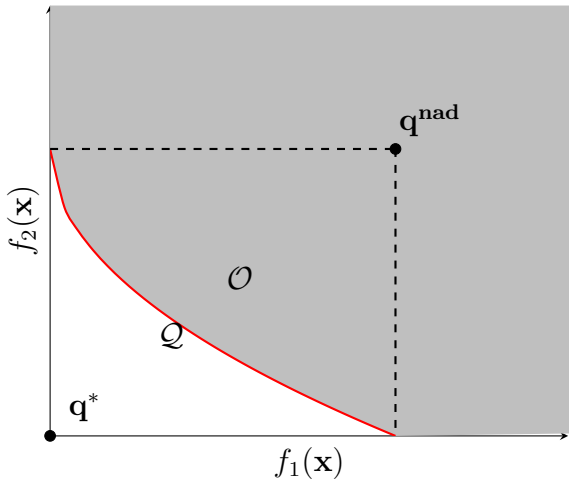

Figure 2.1: Pareto optimal front with nadir and ideal objective vectors

Definition 2.5 ***Pareto dominated*** *A decision vector $x_1 \in \mathcal{F}$ dominates another decision vector $x_2 \in \mathcal{F}$ (indicated by $x_1 \prec x_2$) if and only if $\forall i \in [1, n_m], f_i(x_1) \leq f_i(x_2)$, and $\exists i \in [1, n_m], f_i(x_1) < f_i(x_2)$.*

Definition 2.6 ***Pareto optimal*** *A decision vector $x_1 \in \mathcal{F}$ is pareto optimal if $\nexists x_2 \in \mathcal{S}$ such that $x_2 \prec x_1$.*

Definition 2.7 ***Pareto optimal set*** *A set $\mathcal{P}$ is pareto optimal if it contains only pareto optimal decision vectors.*

Definition 2.8 ***Pareto optimal front*** *A set $\mathcal{Q}$ is the pareto optimal front if for the pareto optimal set, it contains the corresponding objective vectors.*

Definition 2.9 ***Nadir objective vector*** *Denoted as $\mathbf{q}^{\mathbf{nad}}$, it is the objective vector that contains the worst objective values from the pareto optimal front.*

Definition 2.10 ***Ideal objective vector*** *Denoted as $\mathbf{q}^*$, it consists of the optimal objective values. For minimization this will be the lowest, while for maximization the highest.*

In Figure 2.1, it is implied that $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are to be minimized, with $(0, 0)$ as their lowest respective values. This is due to the ideal objective vector being positioned at that location. Likewise, minimization could also be spotted with the nadir point located away from $(0, 0)$ but with larger, positive values.

A	G	G	T	C		A	G	G	T	C	-
G	A	T	C	A	$\rightarrow$	-	G	A	T	C	A

Figure 2.2: Sample Alignment, with α = {A,G,C,T}

2.2 Multiple Sequence Alignment

MSA takes a significant role in the evolution of protein and DNA structures. Each of these sequences can be represented as a sequence of strings, with the individual characters in these strings coming from an alphabet of amino acids. When a mutation occurs, this can lead to three operations: insertion of amino acid, replacement, and deletion. Thus over time, sequences will appear increasingly dissimilar to their ancestors.

Below are Definitions 2.11 and 2.12, each of which remain crucial for understanding the fundamentals of MSA.

Definition 2.11 ***Alphabet*** *A set of characters, denoted as α. As an example, α = {a,b,c} allows for 'a', 'b', or 'c'.*

Definition 2.12 ***Sequence*** *Given an alphabet α, it takes the form of a list of its characters. For example, if α = {a,b,c}, then a possible sequence may be s = [a,b,c,b,a]. The length of s is denoted as $|s|$ and describes the total amount of characters in the sequence, which in this case is 5.*

As is derived from the name, MSA is the process of aligning three or more sequences which follow some alphabet α. The set of these multiple sequences is formally denoted as $S = \{s_1, s_2, ..., s_n\}$, where $s_1, s_2, ..., s_n$ are individual sequences, and $|S|$ is the number of sequences. Alignment is the process of matching characters between $s_1, s_2, ..., s_n$, and requires that $|S| > 1$. If viewed in a tabular form such as in Figure 2.2, it takes the form of homogeneous characters in a column. In order for alignment to proceed, spaces, or "-", must be inserted so as to match the characters between various sequences. These spaces will be referred to as indels (INsertion/DELetion).

2.3 Particle Swarm Optimization

PSO was first introduced as a stochastic process by James Kennedy and Russell Eberhart; inspired by swarm theory, bird flocking, and fish schooling [3]. A swarm of particles are initialized at random positions in feasible space, where for a set number

Figure 2.3: PSO Neighbourhood Topologies (Star; Left, Ring; Right)

of iterations, these particles are guided by and share information between themselves and other particles. The position of each particle is represented as a solution to the optimization problem that is being solved.

An objective function $f : \mathbb{R}^{n_x} \to \mathbb{R}$, where n_x is the dimension of the search space, measures the fitness value of each particle. Without loss of generality, the goal of PSO may be expressed either as the maximization or minimization of $f(\mathbf{x})$. For the purposes of this section, it will be considered as maximization. Let P denote the swarm of particles, with each particle $p \in P$ having a position $\{x_1, x_2, ..., x_{n_x}\}$ and velocity $\{v_1, v_2, ..., v_{n_x}\}$. As mentioned previously, the position is representative of a solution, meanwhile the velocity updates subsequent particle positions.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.2}$$

Equation 2.2 clarifies the position update function, where previous positions are added with the current velocity. t represents the iteration, and therefore $t+1$ corresponds with the value of the next iteration. Each particle p_i keeps track of its own best fitness position throughout all iterations, known as its personal best, and is denoted as y_i. Additionally, the swarm maintains a neighbourhood best, which can be formulated in many ways. Figure 2.3 illustrates two popular neighbourhood strategies, namely star and ring, from left to right respectively. The star structure is connected to all other points, meaning that the solution found is the best among all particles in the swarm. This is why alternatively, star can be referred to as the global best. Subsequently, ring consists of particles that are connected to only two others, and is also known as local best.

It has been found that selection of neighbourhood strategy has an affect on the performance of the PSO, such as star leading to quicker convergence than ring, and so it should be considered as another parameter for selection [28, 29]. That being

said, Algorithm 1 uses the global best method, where the position is denoted as $\hat{y}$.

$$v_i(t+1) = \omega \times v_i(t) + r_1 \times c_1 \times (y_i(t) - x_i(t)) + r_2 \times c_2 \times (\hat{y}(t) - x_i(t)) \tag{2.3}$$

Equation 2.3 expresses the particle velocity update function, which includes the sum of three terms. Each of these terms are named a component of the velocity update, and they go as follows:

- Inertia Component (ω): could also be known as momentum, as it contributes a portion of the previous velocity $v_i(t)$ into the current calculation. Increased ω values mean particles tend to have similar directions of motion in-between iterations.
- Cognitive Component (c_1): associated with the personal best, which is only known for that respective particle. It also includes r_1 which is a random variable generated between $(0, 1)$. As c_1 is raised, particles are more introspective and follow towards their previously discovered personal best.
- Social Component (c_2): connected to the neighbourhood strategy, which shares knowledge between at least two particles, such as the global best distributing information among all particles for the star topology. r_2 is affiliated with the social component, and is a random variable generated between $(0, 1)$. When c_2 is raised, the information shared between particles becomes more valuable.

As can be interpreted, the choice of parameters ω, c_1, and c_2 have great affects on the performance of PSO, and so great care should be put into their selection [30].

During the swarm initialization process, particle positions are randomly generated within some bound $[b_l, b_h]$, with velocities initiated at 0. Afterwards, the personal best is set to the initial position, with an update to the global best occurring after all particles are created. It should be noted that at initialization, $\hat{y}(0)$ could be set to any particle, as the first step during iteration is to update the $\hat{y}$ value correctly. Following the global best update, velocity and positions are updated using Equations 2.2 and 2.3, where the personal best is subsequently updated. This entire process is formally expressed as Algorithm 1.

Algorithm 1 Particle Swarm Optimizer

1: **for** $\forall p_i \in P$ **do** ▷ Initialize swarm
2: $x_i(0) \leftarrow [x_1, x_2, ..., x_{n_x}]$, where each $x_i = \text{rand}[b_l, b_h]$

```
    v_i(0) ← [0, 0, ..., 0]^{n_x}
    y_i(0) ← x_i(0)
    ŷ(0) is set to the best particle in P
t ← 0
while termination criteria not met do                    ▷ Iterations
    for ∀p_i ∈ P do                                      ▷ Update global best
        if f(y_i(t)) > f(ŷ(t)) then
            ŷ(t) ← y_i(t)
    for ∀p_i ∈ P do                                      ▷ Velocity and position update
        use Equation 2.3 for velocity update
        use Equation 2.2 for position update
        if f(x_i(t + 1)) > f(y_i(t)) then                ▷ Personal best update
            y_i(t) ← x_i(t + 1)
    t ← t + 1
```

2.3.1 Binary Particle Swarm Optimization

While the PSO algorithm works for real-valued spaces, alterations must be made to have it function correctly for discrete search spaces. James Kennedy and Russell Eberhart proposed a solution to this conflict in 1997, creating BPSO [17]. It behaves similarly to the PSO, but with a change to position initialization, the position update function, and interpretation of the velocity function. Naturally since in a binary space, the initialization of all particle positions must follow the form $\{x_1, x_2, ..., x_{n_x}\}$, where each $x_i = 0$ or 1.

The largest difference between PSO and BPSO is the position update function, which in case of the PSO, is defined as Equation 2.2. For the BPSO, it is defined as follows:

$$x_{ij}(t+1) = \begin{cases} 1, & \text{if rand}[0,1] < S(v_{ij}(t+1)) \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

where *rand* initializes a random value within the subsequently provided bounds, in this case $[0, 1]$, and S represents the sigmoid function. From first glance, it is noticeable that stochastic measures are involved in the process of updating a particle position. However, more indirectly, v_{ij} is acting as the probability that x_{ij} takes the value of 1. Algorithm 2 illustrates the pseudocode for BPSO.

Algorithm 2 Binary Particle Swarm Optimizer

for $\forall p_i \in P$ **do** ▷ Initialize swarm
    $x_i(0) \leftarrow [x_1, x_2, ..., x_{n_x}]$, where each $x_i = \text{rand}[0, 1]$
    $v_i(0) \leftarrow [0, 0, ..., 0]^{n_x}$
    $y_i(0) \leftarrow x_i(0)$
    $\hat{y}(0)$ is set to the best particle in P
$t \leftarrow 0$
while termination criteria not met **do** ▷ Iterations
    for $\forall p_i \in P$ **do** ▷ Update global best
        if $f(y_i(t)) > f(\hat{y}(t))$ **then**
            $\hat{y}(t) \leftarrow y_i(t)$
    for $\forall p_i \in P$ **do** ▷ Velocity and position update
        use Equation 2.3 for velocity update
        use Equation 2.4 for position update
        if $f(x_i(t+1)) > f(y_i(t))$ **then** ▷ Personal best update
            $y_i(t) \leftarrow x_i(t+1)$
    $t \leftarrow t + 1$

2.3.2 Angular Modulated Particle Swarm Optimization

The AMPSO algorithm was introduced by Pampera et. al. as a potential improvement to the BPSO [15]. Similarly to the latter, it performs on the binary space, however a main appeal is the low-dimensional position used, which attempts to achieve better quality solutions quickly. The particle positions for AMPSO are established as four real-valued dimensions, where each dimension represents the coefficients a, b, c, d found in the trigonometric generation function:

$$g(x) = sin(2\pi(x - a) * b * cos(A)) + d$$
$$\text{where } A = 2\pi * c(x - a) \tag{2.5}$$

and x is an evenly spaced value from an interval. As can be interpreted from the function, a represents the horizontal shift of the generation function, b is the maximum frequency of sin, c is the frequency of cos, and d is the vertical shift. For easier interpretation of what this generation function may create, Figure 2.4 displays the generation function with coefficients $(1, 2, 1, 2)$.

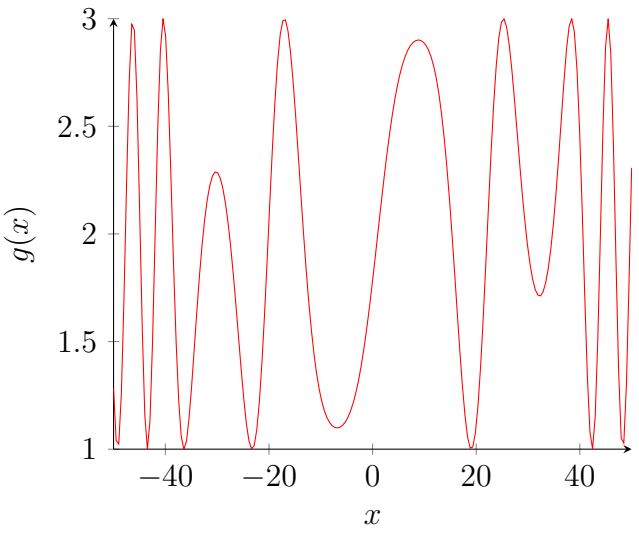

Figure 2.4: Generation Function using $(1, 2, 1, 2)$

In order to operate in the binary space, this generation function from Equation 2.5 can produce the amount of bits needed. This is done by submitting the evenly spaced x values within some interval, and if a positive $g(x)$ is returned, then the bit is 1, otherwise 0. The interval for x can be derived from the amount of bits needed. For example, if 5 bits are required, and each x value has a step of 0.1, then the interval would be $[0, 0.4]$. This method of bit generation will be used for all AMPSO experimentation. Mathematically the bit value is formulated as:

$$bit = \begin{cases} 1, & g(x) > 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

This allows us to retrieve an n_x-dimensional bit matrix, which is useful in formulation of problems such as MSA. A critical detail is the need to generate a bit matrix before fitness comparisons. For greater understanding of this algorithm, its respective pseudocode is found as Algorithm 3. One important, and unmentioned necessity is the designation of a particle position range, which is defined as $[b_l, b_h]$. To conclude on AMPSO, the position and velocity update rules follow that of the PSO. It is the utilization of a generation function and a fixed 4-dimensional position that are the principal differences. For situations where storage is not an issue, performance may be enhanced by storing y_i and $\hat{y}$ bit matrices, and updating them, instead of repetitive re-calculation.

Algorithm 3 Angular Modulated Particle Swarm Optimizer

1: **for** $\forall p_i \in P$ **do** ▷ Initialize swarm

```
    x_i(0) ← 4 random values between [b_l, b_h]
    v_i(0) ← [0, 0, 0, 0]
    y_i(0) ← x_i(0)
    ŷ(0) is set to the best particle in P
t ← 0
while termination criteria not met do                          ▷ Iterations
    gBestBitmatrix ← gen(ŷ(t))
    for ∀p_i ∈ P do                                            ▷ Update global best
        pBestBitmatrix ← gen(y_i(t))
        if f(pBestBitmatrix) < f(gBestBitmatrix) then
            ŷ(t) ← y_i(t)
    for ∀p_i ∈ P do                                            ▷ Velocity and position update
        use Eq. 1 for velocity update
        use Eq. 2 for position update
        bitmatrix ← gen(x_i(t + 1))
        pBestBitmatrix ← gen(y_i(t))
        if f(bitmatrix) < f(pBestBitmatrix) then               ▷ Personal best update
            y_i(t) ← x_i(t + 1)
    t ← t + 1
```

2.4 Multi-guided Particle Swarm Optimization

While techniques can be used to solve MOOPs with PSO, BPSO, and AMPSO, the MGPSO is specifically targeted towards solving such category of problems. Created by C. Scheepers in 2017, it contains one swarm per objective function, in addition to a new group of solutions named the archive [16]. Since the choice of neighbourhood topology affects each swarm individually, the archive is a source of inter-swarm knowledge transfer for the MGPSO. However, it only holds non-dominated solutions, of which domination is defined in Definition 2.5.

Introduction of a feasible solution into the archive requires it to not be dominated with those pre-existing. If a newly added solution dominates another in the archive, then the dominated is promptly removed. There is also a maximum capacity to the archive, where when breached, crowding distance calculations are made amongst the solutions present. Then, the entry that contains the lowest value is withdrawn from the archive. Effectively, this removes the most crowded particle and simultaneously

promotes diversity. The pseudocode for crowding distance is found as Algorithm 4.

Algorithm 4

```
1: procedure CROWDING DISTANCE(a)
2:     n ← length(a)
3:     for i=0 to n do
4:         a[i].distance ← 0                          ▷ Initializing distances
5:     for i=1 to n_m do                  ▷ Remember: n_m objective functions
6:         sort(a, f_i)                   ▷ Sort based on objective function f_i
7:         for j=1 to n-1 do
8:             a[i].distance ← a[i].distance + (f_i(a[i+1]) - f_i(a[i-1])
9:         a[0].distance ← ∞
10:        a[n].distance ← ∞
```

The insertion of a new archive component becomes clear when comparing Equation 2.7 with the PSO velocity function (Equation 2.3). This term comes paired with c_3, r_3, and $\hat{a}_i$, which respectively are the archive coefficient $[0, 1] \in \mathbb{R}$, stochastic vector $rand[0, 1]$, and archive guide. Additionally, a variable λ_i introduces itself within the social and archive components, and is used as an exploitation trade-off coefficient for particle i. Both the archive and neighbourhood guide act as social forces for each particle, and thus λ is provided to balance the social and cognitive influence.

$$v_i(t+1) = \omega * v_i(t) + r_1 * c_1 * (y_i(t) - x_i(t)) + \lambda_i * r_2 * c_2 * (\hat{y}(t) - x_i(t)) \\ +(1 - \lambda_i) * c_3 * r_3 * (\hat{a}_i(t) - x_i(t)) \tag{2.7}$$

When performing a velocity update, an archive guide $\hat{a}_i$ is a solution chosen from a competition pool of random archive solutions, usually 2 or 3, where the largest crowding distanced solution is chosen as the guide. Algorithm 5 illustrates the pseudocode for MGPSO. For easier interpretation, S_m is a prefix denoting the particular swarm of a following variable, with $|S_m|$ the quantity of particles in that swarm.

It is important to remember that each swarm has a duty to optimize only its respective objective function. Thus in Equation 2.7 the particle may move towards $\hat{a}$, but if the respective objective function is not improved, then the personal best and global best will both not be updated. The main benefit from this event may include a potential new addition to the archive, that is, if the solution is non-dominated to those already present. Reference to the pseudocode of MGPSO is displayed as Algorithm 5.

Algorithm 5 Multi-Guided Particle Swarm Optimizer

1: Initialize λ to 0
2: **for** each $f_m(x)$ where $m \in [1, n_m]$ **do**
3: **for** each $i \in [1, |S_m|]$ **do**
4: Create a swarm S_m of n_{s_m} particles uniformly within a defined feasible hypercube of dimension n_x
5: $S_m.y_i(t) \leftarrow S_m.x_i(t)$
6: $S_m.v_i(0) \leftarrow [0, 0, ..., 0]^{n_x}$
7: $S_m.\hat{y}_i(0)$ is set to the best particle in S_m
8: $S_m.\lambda_i \leftarrow \text{rand}[0, 1]$
9: $t \leftarrow 0$
10: **while** termination criteria not met **do**
11: **for** each f_m where $m \in [1, n_m]$ **do**
12: **for** each $i \in [1, |S_m|]$ **do**
13: **if** $f_m(S_m.x_i) < f_m(S_m.y_i)$ **then**
14: $S_m.y_i \leftarrow S_m.x_i$
15: **if** $f_m(S_m.y_i) < f_m(S_m.\hat{y}_i)$ **then**
16: $S_m.\hat{y}_i \leftarrow S_m.y_i$
17: Update archive with solution $S_m.x_i$ if feasible
18: **for** each f_m where $m \in [1, n_m]$ **do**
19: **for** each $i \in [1, n_{s_m}]$ **do**
20: Retrieve a solution $\hat{a}_i(t)$ from the archive using tournament selection
21: Use Equation 2.7 to update velocities
22: $S_m.x_i(t + 1) \leftarrow S_m.x_i(t) + S_m.v_i(t + 1)$
23: $t \leftarrow t + 1$

2.4.1 Binary Multi-guided Particle Swarm Optimizer

B-MGPSO is a combination of the aforementioned BPSO, and MGPSO. As such, it combines the quality of maintaining individual swarms, and an archive of non-dominated solutions. However, particle positions and position updates are altered to reflect what is completed in BPSO. One main difference in Algorithm 5 would come on line 22, where instead Equation 2.4 is used. Additionally, on line 4, each particle position would be initialized as $[x_1, x_2, ..., x_{n_x}]$, where each $x_i = \text{rand}[0,1]$. These modifications effectively make B-MGPSO its own variant of MGPSO, and

may produce different outcomes for MSA.

2.4.2 Angular Modulated Multi-guided Particle Swarm Optimizer

AM-MGPSO comes from the amalgamation of AMPSO and MGPSO. Similarly to B-MGPSO, main MGPSO aspects are retained such as the archive, and distinctive sub swarms. Despite that, amendments appear as a change in particle dimension, and usage of a generation function. Therefore on line 4, particle positions are initialized as 4 values in the range of $[b_l, b_h]$. Subsequently, before all fitness calculations and comparisons, such as on lines 13 and 15, the generation function (Equation 2.5) must be used to generate a bit matrix. This bit matrix is then used to obtain the fitness value that is compared. The changes presented within this section warrant AM-MGPSO being presented as its own version of the MGPSO.

2.5 ProbCons

ProbCons was created by Do et al. in 2005, and can also formally be referred to as probabilistic consistency-based multiple sequence alignment [10]. In their work, a novel scoring technique is constructed, named probabilistic consistency. ProbCons follows the progressive alignment strategy. Using this method, an MSA problem executes iterative pairwise alignments, and uses a tree based on similarity measures to determine a way of combining these pairwise conclusions [31]. In other words, numerous adjustments are performed between pairs of sequences, which are then progressively combined together.

Given a set of m sequences $S = \{s_1, s_2, ..., s_m\}$, ProbCons follows the following steps:

1. **Computation of posterior-probability matrices:** Let a^* be the alignment received from the HMM, in addition to x_i and y_i being characters from sequences x and y. Compute a matrix P where $P_{xy}(i, j) = \mathbf{P}(x_i \sim y_i \in a^* | x, y)$. This matrix exhibits the probability $\mathbf{P}$ that x_i and y_i are aligned from the HMM.

2. **Computation of expected accuracies:** Let a be the alignment between x and y. The expected accuracy is the expected number of matching character pairs, divided by the shortest sequence length, all with respect to the alignment

a^*. It takes the mathematical form:

$$\mathbf{E}_{\mathbf{a}^*}(\text{accuracy}(a, a^*)|x, y) = \frac{1}{\min(|x|, |y|)} \sum_{x_i \sim y_j \in a} \mathbf{P}(x_i \sim y_j \in a^*|x, y) \qquad (2.8)$$

Equation 2.8 is calculated on each pair of sequences for the alignment a that maximizes its value, by dynamic programming.

3. **Probabilistic consistency transformation:** Scores are re-estimated to integrate the similarity of sequences x and y to other sequences in S, using consistency alignments. A newly devised probability equation is provided for the matrix P that was calculated in step 1:

$$P'_{xy} \leftarrow \frac{1}{|S|} \sum_{z \in S} P_{xz} P_{zy}$$

This may be applied multiple times as needed. ProbCons also ignores multiplication values which are below a set threshold.

4. **Computation of guide tree:** Let $E(x, y)$ equal the expected accuracy in Equation 2.8. Initially, each sequence is placed in its own respective cluster. Then, two sequences x and y with the highest $E(x, y)$ value are combined. Afterwards, the combination of cluster xy with another sequence z should maximize the formula $E(x, y)(E(x, z) + E(y, z))/2$. This process is repeated until there is only one cluster remaining.

5. **Progressive alignment:** Sequences are aligned using the profile–profile Needleman–Wunsch procedure, as is determined by the guide tree calculated in the previous step. All alignments are scored using the sum-of-pairs scoring function, while also ignoring any gap penalties.

- **Iterative refinement (post-processing):** The alignment is separated into two groups, with the profile–profile Needleman–Wunsch procedure used on the partitions. This same process is repeated for a set amount of iterations, whereby the sum-of-pairs increases monotonically.

Chapter 3

Experimentation

All experimentation is completed using empirical analysis, with the computational intelligence methods mentioned in Sections 2.3.1, 2.3.2, 2.4. A parameter configuration can be defined as the set of usable parameters for an algorithm. They are provided in the following subsections, where 30 independent runs are completed for each algorithm, on each combination of listed configuration. In addition, each process is run for 2500 iterations.

First, the problem structure is formulated in Section 3.1. Then, the BPSO and AMPSO, as well as their MGPSO versions are discussed in Section 3.1.1 and Section 3.1.2, respectively. Concluding is Section 3.2, where it is outlined how comparative evaluations will take place between between algorithms of the same problem formation, and those that differ. This section is progressed by Section 4, where results of the experimentation are reviewed.

3.1 Problem Formation

The algorithms used for this study all need a defined, usable formation of the MSA problem. To begin, the longest sequence is denoted as s_l, and its respective length $|s_l|$. Then, let $col = \lceil 1.2 * |s_l| \rceil$. A bit matrix is subsequently defined as a matrix of size $|S|$ x col, where each element is only 0 or 1, representing a character of the sequence or an indel, respectively. Note that $|S|$ is the number of sequences as mentioned in

s_1	0	1	1	0	0	0	$\rightarrow$	A	-	-	G	A	T
s_2	0	0	1	0	0	1		G	T	-	A	T	-

Figure 3.1: Example Formation of Sequences {A,G,A,T} and {G,T,A,T}

Section 2.2, and as such, row i will reference sequence $s_i \in S$.

It has been found that many MSA problem sets do not require more than 20% indel insertions [32], and so this is the reasoning behind the calculation of *col*. For visual clarity, Figure 3.1 illustrates a bit matrix, with its corresponding sequence strings. Note how in s_2, there is one trailing indel which is found at the end, and one inserted, as it is in-between characters. As the number of trailing indels increases, the numbers of inserted indels decrease, and vice versa.

The formation of Figure 3.1 will be used for the position vector of particles in the BPSO and B-MGPSO algorithms. For AMPSO and AM-MGPSO, the bit matrix must be generated using Equation 2.5. Afterwards, it can be evaluated using the techniques mentioned in the succeeding Sections 3.1.1 and 3.1.2.

A feasible bit matrix is one that can accurately represent a sequence with indels. For this to be true, there must be $|s_i|$ characters for each $s_i \in S$, with the rest consisting of indels. Thus, at the very least, there must be as many 0 values for each row i as there are characters for its corresponding sequence s_i. If there are more 0 values than s_i, then simply convert ones trailing into indels so that the amount of indels matches $col - |s_l|$. This sort of feasibility check will be listed as *lt*, for it only checks if there are less than the required amount of 0 values.

Additionally, another feasibility requirement can be added in the form of *lt-gt*, which combines the notion of *lt* mentioned previously, and a new condition *gt*. This *gt* checks if there are more than the required amount of 0 values in row i, which makes the combination more restrictive than only checking for *lt*. However, as will be mentioned in Section 3.1.1, and later in 3.1.2, both AMPSO and AM-MGPSO respectively have struggles initializing using *lt-gt* as it is more restrictive, thus making the differentiation between *lt-gt* and *lt* necessary.

3.1.1 Single-objective

MSA can be formulated as a multi-objective problem, where multiple objective functions are used to evaluate the quality of a decision vector. However, the AMPSO and BPSO used in this work employ a single-objective approach, which disallow the use of more than one objective function. There are, however, techniques that allow for multiple objectives to be employed as singular, one of these being a weighted aggregation approach.

A weight aggregated objective function consists of the weighted sum of various

individual objective functions. Formally, this is defined as:

$$f(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + ... + w_n f_n(\mathbf{x}) \tag{3.1}$$

where w_i are the weights, $f_i(\mathbf{x})$ the individual objective functions, and $f(\mathbf{x})$ the weight aggregated objective function. For the purposes of this work, BPSO and AMPSO will use two individual objective functions: the number of aligned characters ($f_1(\mathbf{x})$), and number of trailing indels ($f_2(\mathbf{x})$). Note that as trailing indels increase, the number of indels inserted decreases. Given Equation 3.1, the optimization problem to be solved is then illustrated as:

$$\max f(\mathbf{x}) = \max \left[w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x})\right] \tag{3.2}$$

All particles are initialized as feasible. For this to be successful with BPSO, a bit matrix B is primarily created for each particle, only containing the value 0. Then, for each row $i \in B$, values of 1 are inserted in random locations, such that the number of 1 values in the row is equal to $\lceil 0.2 * |s_l| \rceil$. This takes the shape of randomly inserting indels into a sequence, but in such a way that it still remains feasible.

The process of feasible initialization for AMPSO is much simpler, although has seen issues in preliminary testing. Random values are initialized for the position, within the range of [-1, 1]. Then, a bit matrix is generated and validated. If feasible, then the particle is created successfully, but if not, then this process repeats until position values are generated such that the bit matrix is feasible. A subject of conflict is a potential infinite loop, where the algorithm will never begin as no feasible initial state can be found. As was discussed in Section 3.1, a restrictive constraint named *lt-gt* may create an infinite loop that causes AMPSO not to begin, and for this reason the *lt* constraint is used for all AMPSO feasibility assessments. Due to the potential for its feasible initialization, BPSO maintains the *lt-gt* constraint throughout its execution.

$$\omega \in (-1, 1), \;\; c_1 + c_2 < \frac{24(1 - \omega^2)}{7 - 5\omega} \tag{3.3}$$

Comparisons between AMPSO and BPSO are made first from the weight configuration, then size, and lastly coefficient. Table 3.1 provides coefficient values that follow the PSO stability condition outlined in Equation 3.3 [33]. Values were selected that allow for comparisons between the effectiveness of specific coefficients, as some appear less than others within the same row. Following is Table 3.2, in which weight, as well as swarm size configurations are seen. As a note, in Table 3.2 the selected weight values and swarm size are separate. Both AMPSO and BPSO are executed

#	ω	c_1	c_2
1	0.7098150314023034	1.6788775458244407	0.899446463381824
2	0.7861878289341226	1.2489605895810598	1.211314077689869
3	0.7226988763584911	0.7877525185622731	1.3569405150479763
4	0.7566563010222623	0.9360167168210383	0.8238319031198684
5	0.3260770959583924	1.397180934100446	1.3464223101885027

Table 3.1: AMPSO and BPSO Coefficient Configuration

w_1	w_2	n
1	0	30
0	1	40
0.5	0.5	50

Table 3.2: AMPSO and BPSO Weight (Left) and Size (Right) Configuration

for every combination of selected coefficient, weight, and size composition.

3.1.2 Multi-objective

In Section 3.1.1 the MSA problem is aggregated of multiple objective functions with weights. However, for an algorithm such as MGPSO, this conversion is not necessary, as the raw objective functions themselves can be utilized. These two functions will be denoted as f_1 and f_2, and will follow the optimization problem:

$$\min\ [-f_1(\mathbf{x}), f_2(\mathbf{x})] \tag{3.4}$$

where $f_1(\mathbf{x})$ is the number of aligned characters, and $f_2(\mathbf{x})$ is the amount of inserted indels. f_1 appears negative because the true objective is to maximize the number of

#	ω	c_1	c_2	c_3
1	0.7548684662307973	1.6403922907668573	1.1689889680218306	1.0925215379609754
2	0.664693311042931	0.12797292579587216	1.141633525977906	1.894374658387147
3	0.6283276486501926	0.462377069982959	0.3400439146349077	1.2075173599176188
4	0.7519652725549882	1.659900433872437	0.17954734318382237	0.4740247791781249
5	0.7499142729146882	0.6484731483137582	1.8292436916244121	0.47188949238877376
6	0.7444666103843537	1.382159367589628	0.35852915141793007	1.204326868805682
7	0.6002649267508061	1.39119402217411	1.7645977300331588	0.3903042201164242
8	0.268990284735547	1.5805667363521902	1.2012673626684758	1.1353123776616685
9	0.121495528917823	0.49423588598014656	0.32362374583425435	0.5277558146365762

Table 3.3: AM-MGPSO and B-MGPSO Coefficient Configuration

n_1	n_2
30	30
15	15
20	40
40	20

Table 3.4: AM-MGPSO and B-MGPSO Size Configuration

aligned characters, and from the duality principle, max $f_1(\mathbf{x}) = \min -f_1(\mathbf{x})$. Note should be taken that for both AM-MGPSO and B-MGPSO, $f_2(\mathbf{x})$ does not represent the amount of trailing indels as it did in Section 3.1.1. Instead, $f_2(\mathbf{x})$ represents the number of inserted indels as was previously indicated. This is an objective to be minimized.

$$0 < c_1 + \lambda c_2 + (1-\lambda)c_3 < \frac{4(1-\omega^2)}{1-\omega+\frac{(c_1^2+\lambda^2c_2^2+(1-\lambda)^2c_3^2)(1+\omega)}{3(c_1+\lambda c_2+(1-\lambda)c_3)^2}}, |\omega| < 1 \tag{3.5}$$

Tables 3.3 and 3.4 respectively illustrate the coefficient configuration, and swarm sizes that are utilized for the MGPSO variants during experimentation. For the coefficient values, some values are less than others, but all are within the MGPSO stability criteria shown in Equation 3.5 [34]. This will give insight into potential effects that arise from the combination of coefficients. Likewise, in Table 3.4 separate swarm sizes are employed for the two objective functions.

For both AM-MGPSO and B-MGPSO, a linearly increasing lambda value is utilized as it has been found to have good performance on two-objective problems [35]. λ will begin at a value of 0 and increase in a fashion of $\lambda = \lambda + \frac{1}{maxIter}$ at the end of every iteration, where $maxIter$ is the maximum amount of iterations. This ensures that throughout the entire algorithm execution, $\lambda \in [0, maxIter]$. In order to ensure such a λ value is within the stability region, Cleghorn et al. have stated that testing Equation 3.5 on the values $\lambda = 0$ and $\lambda = 1$ is sufficient [34].

As was discussed in Section 3.1, there are two feasibility requirements denoted as *lt* and *lt-gt*, where one inspects for less than the needed amount of 0s in a bit matrix row, while the other demands an exact amount. In support of its ability to generate a feasible swarm, the B-MGPSO makes use of the *lt-gt* requirement. However, AM-MGPSO has been found to never begin with such a strict feasibility constraint, and so the less rigorous *lt* is utilized.

3.2 Comparative Evaluation

As output from the BPSO and AMPSO, all results received are from the global best value, except for the amount of infeasible particles. This global best is the best value that was found throughout the entire set of iterations, among all particles. Thus, it remains a valid method to contrast the quality of results between the these two PSO variants. Figures are displayed in Appendix B which illustrate the average number of aligned characters, average number of inserted characters, and the average fitness among the global best outcomes. The first two criterions, or aligned characters and inserted indels, are the main aspects of quality measurement, since they are the individual objective functions which are aggregated to make up fitness calculations.

When mentioning MGPSO, the method of evaluation must differ from the single-objective approach as the global best is not characteristic of what this algorithm attempts to achieve. Instead, it is closeness to the actual Pareto front that is the goal of the MGPSO. In their work, Riquelme et al. list 54 various performance metrics that can be used for comparing multi-objective optimization problems [36]. Out of these, the hypervolume indicator will be used. When comparing results, the same reference point must be applied. However, this point will have coordinates that match that of the most extreme points among those outcomes that are being compared. That is to say, the x-coordinate will be the lowest number of aligned characters among all compared archives, and the y-coordinate the highest amount of inserted indels.

In later comparisons between MGPSO variants and AMPSO and BPSO, only global best values are used. The hypervolume indicator will not be applied as the latter two methods do not carry an archive, and so might be at a disadvantage in this measurement. Moreover, only results from the superior parameter configurations are used as the global best.

Further supplementing PSO comparability, the superior PSO variants are contrasted with a state-of-the-art algorithm named ProbCons [10]. Superior parameters found in the respective introspective study of each PSO algorithm will be used. A mean for inserted indels, and alignments will be calculated, along with their standard deviations. PSO is compared against ProbCons to determine the competitiveness against a popular MSA method.

Chapter 4

Results

Results are gathered from experimentation procedures addressed in the previous Section 3. Comparisons are made and deliberated between the algorithm variants. They are organized as follows: first BPSO and AMPSO will be discussed in Section 4.1 and 4.2 respectively, which is followed by a comparison between these algorithms in Section 4.3. Subsequently, introspective studies for B-MGPSO and AM-MGPSO are illustrated in their respective Sections 4.4 and 4.5, which is then accompanied by contrasting both of their performances in Section 4.6. Differentiation between all the PSO variants used is discussed in Section 4.7, and is succeeded by the concluding Section 4.8. Within the entire section, all comparisons are completed using the Mann-Whitney U statistical test. Each of these tests were executed with a significance level (α) of 0.05.

4.1 BPSO

BPSO was applied for the MSA problem, and had the ability to generate an initial feasible swarm. This could be done while still using the *lt-gt* feasibility constraint, as was mentioned in Section 3.1. However, among all combinations of BPSO configurations, this algorithm was never feasible after the initial generation. Therefore as a result, the global best particle is no better than a random search, since to start indels are inserted at random positions in a bit matrix.

In this random search, the only factors which may produce a slight improvement in outcome are swarm size and weight configuration. When the former is increased, there are more opportunities for a better solution to be randomly generated. Likewise, different weights give heterogeneous fitness calculations, and so the global best is chosen to either maximize the number of aligned characters, the amount of trailing

Configuration	$\{w_1 : 1, w_2 : 0\}$	$\{w_1 : 0.5, w_2 : 0.5\}$	$\{w_1 : 0, w_2 : 1\}$
$\{w_1 : 1, w_2 : 0\}$	-	36.1%	100%
$\{w_1 : 0.5, w_2 : 0.5\}$	0.6%	-	99.4%
$\{w_1 : 0, w_2 : 1\}$	0%	0%	-

Table 4.1: BPSO Weight Configuration Comparison by Alignment (% greater)

Configuration	$\{w_1 : 1, w_2 : 0\}$	$\{w_1 : 0.5, w_2 : 0.5\}$	$\{w_1 : 0, w_2 : 1\}$
$\{w_1 : 1, w_2 : 0\}$	-	1.1%	0%
$\{w_1 : 0.5, w_2 : 0.5\}$	59.4%	-	0%
$\{w_1 : 0, w_2 : 1\}$	100%	92.2%	-

Table 4.2: BPSO Weight Configuration Comparison by Inserted Indels (% less)

indels, or some combination of the two. For this reason, only swarm size and weight compositions will be compared. It should be noted that these results are produced using the *lt-gt* feasibility constraint.

The rest of this following section is organized into two subsections. Section 4.1.1 examines the affects weight configuration have on BPSO, while Section 4.1.2 follows similar procedures, but with various experimental swarm sizes. Analysis is concluded in Section 4.1.3, where notable verdicts are stated.

4.1.1 Weight Configuration

Figure B.1 displays various BPSO results on each reference set, with colours representing a different composition of weights. Each point of similar colour is seen clustered together, while also apart from those that are not of the same pigment. However in the case of reference set 6, results are slightly dispersed in the cases where $\{w_1 : 0, w_2 : 1\}$ (blue) and $\{w_1 : 0.5, w_2 : 0.5\}$ (yellow). These visual results are to be expected, as although BPSO acted as a random search, the global best will be chosen with respect to the fitness calculations. Since an alteration in weight values changes Equation 3.2, this will result in a different algorithmic outcome.

Table 4.1 displays the percentage of experiments where a weight configuration had statistically more aligned characters than another set of weights for the same remaining parameter values. It can be seen that when $\{w_1 : 1, w_2 : 0\}$, there are always more aligned characters than $\{w_1 : 0, w_2 : 1\}$, and only sometimes considerably more than $\{w_1 : 0.5, w_2 : 0.5\}$. On comparisons of using $\{w_1 : 0.5, w_2 : 0.5\}$, there were very few cases where the number of alignments was significantly more than

n	30	40	50
30	-	1.7%	1.7%
40	13.3%	-	0.6%
50	30.6%	14.4%	-

n	30	40	50
30	-	2.8%	2.2%
40	17.2%	-	3.9%
50	27.2%	7.8%	-

Table 4.3: BPSO Swarm Size Comparisons by Alignment (left; % greater) and Inserted Indels (right; % less)

$\{w_1 : 1, w_2 : 0\}$, but almost all were deemed superior than $\{w_1 : 0, w_2 : 1\}$. These results are not unreasonable, since when $\{w_1 : 1, w_2 : 0\}$, full attention is on making alignments, rather than a divided interest between this criterium and maximizing trailing indels as in $\{w_1 : 0.5, w_2 : 0.5\}$. Concluding with the number of alignments is $\{w_1 : 0, w_2 : 1\}$, which never gave any outcomes that were significantly better than the other two weight compositions.

Table 4.2 expresses the proportion of considerably less inserted indels between weight configurations with equal outstanding parameters. In the same manner as the results in Table 4.1, all the values in Table 4.2 are intuitive. Where w_2 values increase, the number of inserted indels are commonly less than their lower counterparts. This occurs from w_2 affecting the result of f_2 on fitness, which is to maximize the number of trailing indels. As such, $\{w_1 : 1, w_2 : 0\}$ is infrequently better than $\{w_1 : 0.5, w_2 : 0.5\}$, and never superior than $\{w_1 : 0, w_2 : 1\}$. Likewise, $\{w_1 : 0.5, w_2 : 0.5\}$ commonly surpasses $\{w_1 : 1, w_2 : 0\}$, while also never improving over $\{w_1 : 0, w_2 : 1\}$. To conclude, $\{w_1 : 0, w_2 : 1\}$ is always better than $\{w_1 : 1, w_2 : 0\}$ and predominantly outperforms $\{w_1 : 0.5, w_2 : 0.5\}$.

4.1.2 Size Configuration

Figure B.2 illustrates BPSO results, where each swarm size is indicated as the colourings blue, green, and red for sizes 30, 40, and 50, respectively. When $\{w_1 : 0, w_2 : 1\}$, it appears that the colour red is found closest to the y-axis most times. For the other configurations, such as when $\{w_1 : 1, w_2 : 0\}$ and $\{w_1 : 0.5, w_2 : 0.5\}$, red can be seen further along the x-axis than the other colours. This visual representation may suggest that having a larger swarm size provides a positive contribution to maximizing the number of aligned characters and trailing indels.

Table 4.3 includes two individual tables, each of which portray the percent of significantly greater alignments, and considerably less inserted indels, respectively from left and right. These are done with comparisons between swarm sizes in order

to deduce algorithm performance. From the values presented on the left of Table 4.3, it is clear that using a higher number of particles increased the chance of determining more aligned characters.

When $n = 30$, there are very low percentage rates against $n = 40$ and $n = 50$, both equalling 1.7%. Further, when $n = 40$, the proportion already increases to 13.3% against $n = 30$, however, as $n = 40$ is less than $n = 50$, the comparative value drops to 0.6%. $n = 50$, which is the highest swarm size value taken for experimentation, received both 30.6% and 14.4% significantly higher alignments made against $n = 30$ and $n = 40$ respectively. The higher swarm size has provided better contributions to results, most likely because since BPSO acted as a random search, a higher number of initialized particles means more probabilities for finding a solution with better alignments.

On the right hand side of Table 4.3, the percent of substantially less inserted indels is shown and contrasted among various swarm sizes. Similarly to before, scanning through the values would supply the notion that an increased number of particles reduces the number of spaces found in-between characters. With $n = 50$, 27.2% and 7.8% of experiments were significantly better than $n = 30$ and $n = 40$, respectively. Considering the values when $n = 40$, a large 17.2% of trials remain considerably improved over $n = 30$, but only 3.9% over $n = 50$. Lastly, $n = 30$ did not have superior results, as only 2.8% and 2.2% of algorithmic outcomes were substantial over $n = 40$ and $n = 50$, respectively, on the number of inserted indels.

It can be assumed that for both fitness objective functions, the swarm size makes a difference in outcome for BPSO. Where the number of particles increase, so do the proportion of good quality solutions. The most reasonable explanation for this is due to the algorithm having generated more potential solutions, which gives rise to an increased opportunity of finding one that is substantial.

4.1.3 Conclusion

It was introduced that due to its complete infeasibility throughout all iterations of execution, BPSO acts no better than a random search. Indels were inserted in random locations during the generation process, and since the global best is thereafter never updated, only the weight configuration and swarm size have an affect on results. For the former, it changes which particle is selected as the global best, as Equation 3.2 is modified depending on the weight values. On the latter, more particles corresponds with more generated solutions. Intuitively, more should be superior.

When comparing the results as discussed in Section 4.1.1 and 4.1.2, it becomes clear that they support the aforementioned sentiment. Employing various weight compositions did affect the final outcome, as the global best selection process was modified, and better suited for several different objective purposes. Furthermore, using a larger amount of particles in the swarm did affect the quantity of significantly improved alignments, with the same among significantly less inserted indels.

Since the BPSO acts like a random search, there was no need for using 2500 iterations, and so algorithm time could have been much quicker. It is likely that the curse of dimensionality had an affect on the infeasibility rates. For sequence reference set 6, the maximum length sequence is 906, which means the corresponding bit matrix had $\lceil 906 * 1.20 \rceil = 1087$ columns, and a total of 7 rows for each of its sequences. In total this would give the position, and velocity vectors 7609 entries for modification, every iteration.

During preliminary testing of the first parameter composition, on the first sequence set it was seen that initial velocity values were not large. As such, applying an initial velocity limit would seemingly not assist BPSO on the sequences used for this study. Although there may be faults in the selected parameters, this algorithm may simply not be meant to handle the manipulation of such large vectors.

4.2 AMPSO

In this section AMPSO is executed on the listed sequences in Appendix A, and results are compared between various parameters as expressed in Section 3.1.1. The following section is organized as follows: Section 4.2.1 begins with displaying results from different weight compositions, which is then followed by Section 4.2.2 where different swarm sizes are compared. The last portion of comparisons for AMPSO is found in Section 4.2.3, where coefficient configurations are differentiated. To conclude is Section 4.2.4, where general discussions of interest relating to AMPSO and MSA are discussed.

4.2.1 Weight Configuration

From Figure B.3, it becomes apparent that a change of weight values affects the outcome of AMPSO. Each colour differentiates between the weights, as is discussed in Appendix B, and from what can be seen, all colours are spaced out from each other, with points located relatively close to those that are of the same pigment.

Configuration	$\{w_1 : 1, w_2 : 0\}$	$\{w_1 : 0.5, w_2 : 0.5\}$	$\{w_1 : 0, w_2 : 1\}$
$\{w_1 : 1, w_2 : 0\}$	-	62.8%	100%
$\{w_1 : 0.5, w_2 : 0.5\}$	2.2%	-	100%
$\{w_1 : 0, w_2 : 1\}$	0%	0%	-

Table 4.4: AMPSO Weight Configuration Comparison by Alignment (% greater)

Configuration	$\{w_1 : 1, w_2 : 0\}$	$\{w_1 : 0.5, w_2 : 0.5\}$	$\{w_1 : 0, w_2 : 1\}$
$\{w_1 : 1, w_2 : 0\}$	-	0%	0%
$\{w_1 : 0.5, w_2 : 0.5\}$	99.4%	-	0%
$\{w_1 : 0, w_2 : 1\}$	100%	100%	-

Table 4.5: AMPSO Weight Configuration Comparison by Inserted Indels (% less)

All plots on Figure B.3 where $\{w_1 : 0, w_2 : 1\}$ (blue) express a maximum number of trailing indels, since where y represents the number of inserted indels, the value is always 0. Thus, the results received give perfect score on their respective weighted fitness calculations. That being said, the number of aligned characters for $\{w_1 : 0, w_2 : 1\}$ is always less than both $\{w_1 : 1, w_2 : 0\}$ and $\{w_1 : 0.5, w_2 : 0.5\}$. Likewise, the same occurs on $\{w_1 : 0, w_2 : 1\}$ against $\{w_1 : 1, w_2 : 0\}$ and $\{w_1 : 0.5, w_2 : 0.5\}$ for amount of inserted indels. These results are to be expected, since $w_1 = 0$ represents no effort is given on alignment, and $w_2 = 1$ full effort on maximizing trailing spaces. As a result of increasing the number of trailing spaces, inserted indels are minimized. The results for alignments regularly seem to be in favour of $\{w_1 : 1, w_2 : 0\}$ than $\{w_1 : 0.5, w_2 : 0.5\}$. This is sensible because there is an equal, but divided amount of attention between aligning characters and ensuring indels are trailing.

In Table 4.4, the percentage of experiments with significantly greater aligned characters between each weight configuration is illustrated. When $\{w_1 : 1, w_2 : 0\}$, AMPSO inserted indels that attempt to maximize the number of aligned characters, and it produced substantially high percentages against both of the other compositions, especially $\{w_1 : 0, w_2 : 1\}$. For $\{w_1 : 0.5, w_2 : 0.5\}$, a similar 100% is spotted against $\{w_1 : 0, w_2 : 1\}$, but a very low 2.2% against $\{w_1 : 1, w_2 : 0\}$. These conceptions follow what is seen in Figure B.3, as x values of $\{w_1 : 1, w_2 : 0\}$ were regularly superior, followed by $\{w_1 : 0.5, w_2 : 0.5\}$ nearby, and then $\{w_1 : 0, w_2 : 1\}$ with the least.

The subsequent Table 4.5 expresses the proportion of significantly less inserted indels amount among trial executions, and against weight compositions. Very clearly, a higher w_2 value leads to higher percentages against those that include a lower w_2.

Configuration	$\{w_1 : 1, w_2 : 0\}$	$\{w_1 : 0.5, w_2 : 0.5\}$	$\{w_1 : 0, w_2 : 1\}$
$\{w_1 : 1, w_2 : 0\}$	-	21.1%	26.1%
$\{w_1 : 0.5, w_2 : 0.5\}$	51.1%	-	40%
$\{w_1 : 0, w_2 : 1\}$	24.4%	11.1%	-

Table 4.6: AMPSO Weight Configuration Comparison by Infeasibility (% less)

As such, $\{w_1 : 1, w_2 : 0\}$ has no experiments with considerably less inserted indels, while both $\{w_1 : 0.5, w_2 : 0.5\}$ and $\{w_1 : 0, w_2 : 1\}$ greatly surpass it. Likewise, $\{w_1 : 0, w_2 : 1\}$ always exceeds the performance of both $\{w_1 : 0.5, w_2 : 0.5\}$ and $\{w_1 : 0, w_2 : 1\}$.

On Figure B.3, it was seen that plotted values for $\{w_1 : 0.5, w_2 : 0.5\}$ (yellow) were not far off from those of $\{w_1 : 1, w_2 : 0\}$ (red) on the x-axis. However, although $\{w_1 : 1, w_2 : 0\}$ has statistically more aligned characters than $\{w_1 : 0.5, w_2 : 0.5\}$, the latter also has substantially less inserted indels. This may perhaps be due to some indecent insertion of spaces, or from $\{w_1 : 0.5, w_2 : 0.5\}$ fearing the addition of too many. However, when $w_2 = 0$ there is no focus on the quality of an alignment, since there is no punishment for inserting indels that do not bring supplemental aligned characters. Thus, if quality of alignment is of concern, then verification of results when $w_2 = 0$ may be necessary.

Table 4.6 exhibits the amount of significantly less infeasible experiments on the same parameters, between weight compositions. Ideally, higher values are better as this would mean less infeasible solutions. Caution should be taken, however, as choice of weights has been shown to produce varying outcomes, but infeasibility should not be the primary choice when selecting these parameter values.

Nonetheless, the values on column $\{w_1 : 1, w_2 : 0\}$ of Table 4.6 show the highest percentages of significantly less infeasibility, meaning that this configuration performed the worst. Likewise, when $\{w_1 : 0, w_2 : 1\}$ is compared against other weight configurations, as is seen in its respective row, percentage values are lowest. This means that $\{w_1 : 0, w_2 : 1\}$ has a harder time maintaining feasible particles than all other selected weight values. It should also be noted that $\{w_1 : 0.5, w_2 : 0.5\}$ has the best percentages in its row, and the lowest overall in its respective column. Therefore when inspecting from an infeasible perspective, a balanced set of weight values may perform better than those that are overtly extreme, such as only $w_2 = 1$ or $w_1 = 0$.

Overall, choice of AMPSO weight composition has a drastic affect on the output that may be received. For higher w_1 values, more alignments are likely to be made, along with potentially more infeasible particles. Moreover, when selecting higher w_2

n	30	40	50
30	-	0.6%	0%
40	10.6%	-	1.1%
50	21.7%	9.4%	-

n	30	40	50
30	-	2.8%	2.8%
40	3.3%	-	3.3%
50	3.9%	5%	-

Table 4.7: AMPSO Swarm Size Comparisons by Alignment (left; % greater) and Inserted Indels (right; % less)

values, concentration is on maximizing trailing indels, but also infeasibility may be increased, similar to higher w_1 values. Appointing balanced weights, such as the equal $\{w_1 : 0.5, w_2 : 0.5\}$ does not favour in alignments, or trailing indels, but it does have better infeasibility. Although not expressed in this work, it is likely also that the quality of alignments for a balanced set of weights is greater than when $w_1 = 1$.

4.2.2 Swarm Size

Before discussion is raised on the choice of swarm sizes, it should be indicated that Figure B.4 does not include points where $\{w_1 : 0, w_2 : 1\}$. This is due to a severe lack of diversity as even among different swarm sizes, all solutions are the same: trailing indels are maximized, and therefore alignments are equivalent. Upon first glance of Figure B.4, the colours blue and green, which correspond with sizes of 50 and 40, are typically seen with greater x values than the remaining colour for size 30 (red). This gives the perception that utilizing additional particles can give a further benefit in the number of alignments.

The left side of Table 4.7 illustrates the percent total of significantly greater alignment scores on equivalent parameter values. It is important to note that for percentage calculations, cases where alignment scores are the same (mostly from $\{w_1 : 0, w_2 : 1\}$) are included, which leads to a decreased value. From what can be spotted, 30 particles were too few for many problems, as a better solution could have been found had this parameter been increased. This is supported by rows $n = 40$ and $n = 50$, which each have the highest percentages, with $n = 50$ in the lead. However, in more general terms, this conclusion informs that choice of swarm size has an affect on the number of alignments produced, even if slightly, and only some of the time.

Subsequent tests with respect to inserted indels can be viewed in Table 4.7, on the right side. These percentages represent the amount of trials among equal parameter values where the swarm size had significantly less inserted indels. This can be interpreted also as having more trailing spaces. Cases where inserted indel out-

n	30	40	50
30	-	35%	66.7%
40	0.6%	-	20.6%
50	0%	0.6%	-

Table 4.8: AMPSO Swarm Size Comparisons by Infeasibility (% less)

comes are identical were included in these calculations, such as the aforementioned $\{w_1 : 0, w_2 : 1\}$ parameter values.

There are still some cases where in trailing indel calculations, using a lower swarm size outperformed a higher amount of particles. This makes it difficult to determine whether the swarm size makes enough of a difference. From results expressed in Figure B.3, it is clear that to ensure a significant affect on maximizing trailing indels, or aligned characters, it is the weight values that should be altered. Then afterwards, a change in swarm size can refine already existing good solutions.

Feasibility between different swarm sizes was compared, and results complement what would be expected. Table 4.8 exhibits the percentages of significantly less infeasible particles amongst the swarm sizes. A lower swarm size of 30 had significantly less infeasible particles than 40 (35%) and 50 (66.7%), while at 40 this was found to occur in 0.6% and 20.6% of trials against 30 and 50, respectively. Following the trend of decreased proportions, 50-particle swarms never had significantly less infeasible solutions than 30, and only 0.6% opposing 40. These results are reasonable, since a swarm with more particles will have more potential opportunities to be included in the infeasible counter.

In general, it was found that choice of swarm size makes a large difference in the alignments that are made, with more particles being better. Take note that this may converge to a point where an increase in size does not make an improvement on the outcome. However, when it comes to the amount of inserted indels, having more particles may not affect the outcome all that much. Although $n = 50$ had higher percentages in its row than $n = 40 and n = 30$, $n = 30$ still had more than half of the largest overall percentage (5%) against both remaining configurations. Therefore it's possible that $n = 50$ performed best, although not by a lot. To conclude, infeasibility proportions were intuitive, since less particles means less opportunity to be infeasible, and so $n = 30$ performed the best in this regard.

	1	2	3	4	5
1	-	4.6%	22.2%	6.5%	21.3%
2	5.6%	-	15.7%	6.5%	22.2%
3	0.9%	0.9%	-	2.8%	7.4%
4	0.9%	1.9%	12%	-	12%
5	0%	0.9%	4.6%	0.9%	-

	1	2	3	4	5
1	-	3.7%	6.5%	10.2%	3.7%
2	5.6%	-	6.5%	11.1%	4.6%
3	2.8%	3.7%	-	5.6%	1.9%
4	0.9%	1.9%	2.8%	-	0%
5	4.6%	5.6%	7.4%	8.3%	-

Table 4.9: AMPSO Coefficient Configuration by Alignment (left; % greater) and Inserted Indels (right; % less)

	1	2	3	4	5
1	-	67.6%	0%	0%	0%
2	1.9%	-	0%	0%	0%
3	45.4%	86.1%	-	4.6%	0%
4	66.7%	81.5%	43.5%	-	0%
5	86.1%	88.9%	60.2%	42.6%	-

Table 4.10: AMPSO Coefficient Configuration Comparison by Infeasibility (% less)

4.2.3 Coefficient Configuration

Figure B.5 displays AMPSO experimentation with colourations divided between selected coefficient configurations. Similarly to Figure B.4, trials where $\{w_1 : 0, w_2 : 1\}$ were not included to illustrate diversity among the other points. The colours blue (set 1) and green (set 2) are noticeably greater in the x-axis for many sequences, although not all. Likewise, both red (3) and dark magenta (5) are seen having the lowest x values by comparison. The remaining colour yellow (4) is seen as the middle ground.

The left of Table 4.9 illustrates the percentage of significantly greater alignments when comparing against various coefficient parameters. Thus, when viewing column values lower is better, while higher is best for rows. This table seems to harmonize the previous figure interpretations, since columns 1 and 2 have a largely decreased percentage rate than 3 and 5, with only a slight reduction from 4. Column 5 illustrates the highest percentages, meaning many other coefficient configurations have more significantly greater alignment scores. Similarly, on row 5 all percentages are low except for against configuration 3. Configuration 4 is in-between the higher and lower percentages from other coefficient sets. Most of the higher values in its respective column come from that of the superior alignment compositions 1 and 2.

On the right side of Table 4.9, the percent of significantly less inserted indels on equivalent parameters other than coefficients are expressed. Configurations 1 and 2

still have superior inserted indel percentages, similar to their alignment of characters. Their respective rows feature the highest proportion, while their columns are among the lowest, being beaten by set 5. It should be noted that where 4 was considered a midpoint between the remaining coefficients on alignments, this does not hold for inserted indels. Percentage values for row 4 are much less than all other configurations, meaning worse performance against the other coefficient sets. Additionally, column 4 has the highest percentages. Perhaps this reveals set 4 is sloppy in its alignments, and inserts unnecessary indels. The coefficients for set 3 and 5 are not the worst, meaning satisfactory measures are kept to ensure redundant indels are trailing. However, with their insufficient character alignment results, they cannot be recommended.

When discussing coefficient parameters and infeasibility, it becomes clear that where 1 and 2 were superior, they are not on this criterium. Table 4.10 manifests these results, with greater percentages meaning more trials with statistically less infeasible solutions. Interestingly, although the first and second configurations are undoubtedly inferior, it appears that on infeasibility, the latter is much worse than the former. Nearly all percentages indicate 5 to be the greatest for having higher percentages in its row, and only 0% values in its column. This may be attributed to the low inertia component (ω) in set 5, which perhaps otherwise would have unrestrained the particles moving in their current velocity and caused them to become infeasible.

4.2.4 Conclusion

When selecting AMPSO for usage on MSA, there are three primary sets of parameters that need be fine-tuned to receive the best outcome. However, these parameters are not equal in algorithm performance. It can be seen from Figure B.3 that much divergence comes from the choice of weight values. This is reasonable as from Equation 3.2, weights determine what is of importance between f_1 and f_2. Thus, these should be selected first as the entire outcome is based on what objective function is deemed a priority, from either maximizing the number of aligned characters or trailing indels.

The subsequent parameter for selection should be the AMPSO coefficients. Results discovered that picking good values could improve both alignment scores and trailing indels, but at a lesser amount than the weight configuration. Coefficient sets 1 and 2 were found to provide the best outcome out of all selected compositions, but at the cost of their high infeasibility. However, the main focus should not be brought solely on number of infeasible solutions, as is expressed by high percentages against the superior results.

	BPSO	AMPSO
BPSO	-	3%
AMPSO	95.6%	-

	BPSO	AMPSO
BPSO	-	0%
AMPSO	100%	-

Table 4.11: BPSO and AMPSO Comparison by Alignment (left; % greater) and Inserted Indels (right; % less)

Lastly, although swarm size has an affect among the objective functions f_1 and f_2, it should be used to refine already 'good' answers found by proper weights and coefficients. In the experiments run, it was beneficial to use more particles, particularly for locating more alignments. However, only a slight effect could be spotted between swarm size and having less inserted indels. Thus, the optimal amount of particles may widely vary depending on the complexity of the reference set.

4.3 AMPSO and BPSO

Both AMPSO and BPSO are used to solve problems in a discrete space, but do so in different ways. For the purposes of this study, a commonality in their employment is the usage of a single-objective MSA formation. Their results on the sequence sets displayed in Appendix A will be discussed. As was mentioned in Section 4.1, BPSO was never found to be feasible throughout its iterations, and so acted primarily as a random search algorithm. This, in turn, did not occur with AMPSO. The results of BPSO were not affected by coefficient parameters because the global best is only updated for feasible solutions, of which there were none after initial generation.

Figure B.6 displays the global best result averages with respect to the number of aligned characters, amount of inserted indels, and the overall fitness. Each of these are represented as their corresponding axes x, y, and z, respectively. The colour black denotes BPSO, while blue is for AMPSO. On almost all of the reference sets, it appears that the colour blue has larger x-values than black, meaning more alignments. Additionally, it can be seen that there are lower y-values for blue as well, primarily for the case where AMPSO has the weights $\{w_1 : 0, w_2 : 1\}$. For some of the reference sets, plot dispersion is similar between the colours. However, this does not hold among all. For example, in sets 2, 4, 5, 6, 10, and 11, blue appears concave down, while black presents itself as concave up.

Table 4.11 exhibits the percentage of significantly greater alignments and considerably less inserted indels. These were constructed with respect to each identical

parameter composition between the two algorithms BPSO and AMPSO. It becomes quickly clear that BPSO does not uphold many respectable results, with the only notable performance coming from 3% of significantly better trials than the alternative. Otherwise, AMPSO dominates, with 100% considerably less inserted indels, and a large 95.6% of trails with comparably substantial amounts of aligned characters.

Overall, it is found that BPSO has not endorsed much strength. Its weak performance may be due to the large dimensionality of both its particle positions and velocity. A quality feature of AMPSO is its employment of only 4 dimensions. This restrains the curse of dimensionality, which may be plaguing BPSO for large MSA sequences. Perhaps for lower-dimensional problems, BPSO gives competitive results. This, however, is not the case for the experimental sequences used.

4.4 B-MGPSO

The B-MGPSO algorithm was employed for optimizing the MSA problem. All experimental cases applied every combination of parameters from Tables 3.3 and 3.4.

Even with all the coefficient values taking place within the stability region of Equation 3.5, all results were found to have complete infeasibility right after the initial generation. This means that effectively, the choice of coefficient configuration would have no affect on the outcome of this algorithm, as particles are not attempted for addition into the archive if infeasible. As such, only the swarm size composition will be compared as it can make a valid amendment in output results.

Since all particles are initially made feasible for the *lt-gt* constraint, an increased number of particles in both sub swarms gives higher potential of adding non-dominated solutions into the archive. Thus, it is intuitively expected that those compositions where there are more overall particles will bestow higher significantly improved hypervolume indicator values than their less populated counterparts.

This section will be divided into two of its own respective subsections. In Section 4.4.1, comparisons are drawn between B-MGPSO swarm sizes. Then in the following Section 4.4.2, a conclusion is made among this algorithm and the results received.

4.4.1 Swarm Size

Figure B.7 displays the best archives on each sequence by hypervolume indicator value, with individual colours each representing a swarm size. Colour explanations can be found in Table B.4. As is to be expected, the experimentation with the least

	$\{n_1 : 30, n_2 : 30\}$	$\{n_1 : 15, n_2 : 15\}$	$\{n_1 : 20, n_2 : 40\}$	$\{n_1 : 40, n_2 : 20\}$
$\{n_1 : 30, n_2 : 30\}$	-	53.7%	3.7%	3.7%
$\{n_1 : 15, n_2 : 15\}$	0.9%	-	0%	1.9%
$\{n_1 : 20, n_2 : 40\}$	6.5%	53.7%	-	1.9%
$\{n_1 : 40, n_2 : 20\}$	6.5%	58.3%	6.5%	-

Table 4.12: B-MGPSO Swarm Size Comparison by Hypervolume Indicator (% greater)

	$\{n_1 : 30, n_2 : 30\}$	$\{n_1 : 15, n_2 : 15\}$	$\{n_1 : 20, n_2 : 40\}$	$\{n_1 : 40, n_2 : 20\}$
$\{n_1 : 30, n_2 : 30\}$	-	74.1%	35.2%	0%
$\{n_1 : 15, n_2 : 15\}$	0%	-	0%	0%
$\{n_1 : 20, n_2 : 40\}$	0%	26.9%	-	0%
$\{n_1 : 40, n_2 : 20\}$	30.6%	98.1%	77.8%	-

Table 4.13: B-MGPSO Swarm Size Comparison by Aligned Characters (% greater; S_{n_1})

particles, namely $\{n_1 : 15, n_2 : 15\}$ (red), is found having the lowest amount of non-dominated archive solutions a majority of the time. By generating more bit matrices than 30, it is reasonable that the outcome should be more likely to have increased solutions in the archive, as there are more attempts for new additions. Also, red can be seen in some sequences featuring archival solutions closer to the nadir point than the others. This comes with the anomaly on sequence 8 where blue is consistently much closer to the nadir than the remaining pigments.

Intuitively, in Figure B.7, it could be considered that having more particles in swarm 2 (S_{n_2}) would show slightly less inserted indels, and those with more in swarm 1 (S_{n_1}) increased alignments. However, since B-MGPSO failed to render feasible particles after swarm initialization, what appears in the best archive figure is simply the act of random generation. In this case, the number of particles in total, or $n_1 + n_2$ is the factor that determines what will be the result, with growing values likely providing better archive quality.

This is greatly expressed by Table 4.12, which illustrates the percentage of trials with significantly greater hypervolume indicator values compared amongst the various swarm size configurations. Each of these had equal remaining parameter values, and so the reference point was calculated using each coefficient composition, on every separate reference set. Then, hypervolume measures could be retrieved from the four respectively selected swarm sizes.

It is clearly visible that the larger swarm sizes had more experimentations that

	$\{n_1 : 30, n_2 : 30\}$	$\{n_1 : 15, n_2 : 15\}$	$\{n_1 : 20, n_2 : 40\}$	$\{n_1 : 40, n_2 : 20\}$
$\{n_1 : 30, n_2 : 30\}$	-	62%	0.9%	32.4%
$\{n_1 : 15, n_2 : 15\}$	0%	-	0%	0.9%
$\{n_1 : 20, n_2 : 40\}$	25%	87%	-	64.8%
$\{n_1 : 40, n_2 : 20\}$	0.9%	20.4%	0%	-

Table 4.14: B-MGPSO Swarm Size Comparison by Inserted Indels (% less; S_{n_2})

were significantly greater than $\{n_1 : 15, n_2 : 15\}$, as these three percentages (53.7%, 53.7%, 58.7%) are all much more than those leftover. As was said previously, this must be due to simply producing more particles. This gives rise to more opportunities for a better solution to be found. However, it remains surprising that during the random generation, still 6.5% of some trials were significantly greater in favour of the increased individual swarm sizes, rather than the balanced $\{n_1 : 30, n_2 : 30\}$, and for one case even $\{n_1 : 20, n_2 : 40\}$. There is no genuine interpretation for this, since both include 60 particles total, and in which swarm they are placed has no affect on the archive.

The subsequent Table 4.13 expresses the proportion of trials where B-MGPSO swarm sizes had significantly greater global best alignments strictly using S_{n_1}. Percentages are very representative of what should be expected. When utilizing more particles in S_{n_1}, there is more focus on f_1. This is reflected in the table since there are many significantly greater trials against all other configurations when $\{n_1 : 40, n_2 : 20\}$, with a lowest of 30.6% against the swarm size that has only 10 less particles in S_{n_1}. Likewise, $\{n_1 : 15, n_2 : 15\}$ fared the worst against all other swarm sizes. Further supporting this notion, even $\{n_1 : 30, n_2 : 30\}$ received 35.2% trials that had significantly greater aligned characters than $\{n_1 : 20, n_2 : 40\}$ in S_{n_1}, again with a difference of only 10 particles.

Established afterwards is Table 4.14, where the percentage of experiments that have significantly less global best inserted indels are shown, but only for S_{n_2}, and by swarm size. Findings were similar to Table 4.13, since more particles in S_{n_2} provides more potential for a better solution to f_2. All swarm sizes where $n_2 > 15$ had trials that performed much better than $\{n_1 : 15, n_2 : 15\}$, as is to be expected since the lowest selected n_2 value is 15. Moreover, $\{n_1 : 15, n_2 : 15\}$ had only 0.9% of trials that resulted in significantly less inserted indels than $\{n_1 : 40, n_2 : 20\}$, but none for the remaining configurations. A trend is evident, whereby having increased n_2 particles led to more trials with significantly less inserted indels. This is expected, since more n_2 particles correlates with additional f_2 calculations. The highest percent can be seen as 87% for $\{n_1 : 20, n_2 : 40\}$ against $\{n_1 : 15, n_2 : 15\}$.

It should be noted that for both Tables 4.13 and 4.14, only the global best results of the respective objective function sub swarms are used. For example, alignment calculations are made using f_1, which is corresponded with the sub swarm S_{n_1}. Thus, Table 4.13 only uses the sub swarm results of S_{n_1}, and Table 4.14 from S_{n_2}. This is due to each sub swarm only emphasizing its corresponding objective function. So while inspecting a sub swarm by another objective function may bring additional insight into performance, there may not be a direct correlation because that sub swarm is not directly calculated in fitness by the other objective functions.

4.4.2 Conclusion

Overall, the performance of B-MGPSO is unideal. It had a completely infeasible set of particles in both swarms right after initialization. Thus, the outcomes that were presented in this section more so resemble the performance of randomly producing MSA solutions. Only swarm sizes were compared as this is the only configuration change that would allow for a difference in output. Previously, it was stated that including a higher swarm size would allow for more potential non-dominated solutions in the archive, and an improved global best for the sub swarm. This has been shown correct by Table 4.12, where all total swarm sizes of $n_1 + n_2 = 60$ performed better than 30.

In addition, results confirmed that an increase in specific sub swarm size varies performance. An increased n_1 led to more significantly aligned trials, and higher amounts of particles within S_{n_2} had more considerably less inserted indel trials. As such, on the experimentation parameters employed, $\{n_1 : 40, n_2 : 20\}$ was best for the former, and $\{n_1 : 20, n_2 : 40\}$ the latter. However, the infeasibility of this algorithm simply cannot be overlooked. Perhaps B-MGPSO may have better applications when the sequences used are smaller, but with no opportunity to refine any solutions generated, this algorithm cannot be recommended for use with large sequence size reference sets.

4.5 AM-MGPSO

AM-MGPSO is formed from the combination of AMPSO and MGPSO. Some of the benefits from AMPSO are expected to arise from this combination as well, namely the reduced number of dimensions to four. In turn this should minimize the curse of dimensionality that may emerge in bit matrices that are large, such as the size of 14

	$\{n_1 : 30, n_2 : 30\}$	$\{n_1 : 15, n_2 : 15\}$	$\{n_1 : 20, n_2 : 40\}$	$\{n_1 : 40, n_2 : 20\}$
$\{n_1 : 30, n_2 : 30\}$	-	56.5%	20.4%	0%
$\{n_1 : 15, n_2 : 15\}$	0%	-	1.9%	0%
$\{n_1 : 20, n_2 : 40\}$	0.9%	23.1%	-	0.9%
$\{n_1 : 40, n_2 : 20\}$	14.8%	68.5%	44.4%	-

Table 4.15: AM-MGPSO Swarm Size Comparison by Hypervolume Indicator (% greater)

x 558 for reference set 3.

This section comprises of AM-MGPSO execution on the reference sets enumerated in Appendix A, using the parameter configurations found in Section 3.1.2. In Section 4.5.1, comparisons are drawn between particle swarm sizes, and is followed by exact-styled distinction by coefficients in Section 4.5.2. Trailing is Section 4.5.3, where conclusive AM-MGPSO findings are expressed.

4.5.1 Swarm Size

A difference in total swarm size should have the contribution of either more focus on the objective functions, or less. Additionally, it can be used to specify supplementary particles for concentration on a particular subset of functions. Figure B.8 displays the best archives found for all configuration of swarm sizes, on each reference set, and using the hypervolume indicator metric. Interestingly, there is not a large amount of dispersion between configurations. At times brown ($\{n_1 : 20, n_2 : 40\}$) is found closer to the nadir point, while in some cases red ($\{n_1 : 15, n_2 : 15\}$) takes this position, and blue ($\{n_1 : 30, n_2 : 30\}$) as well. Alternatively, black ($\{n_1 : 40, n_2 : 20\}$) has seemingly taken a routine for having the most competitive best archive. Note that although all were expressed previously, colourings for Figure B.8 can be seen in Table B.4.

From the results in Table 4.15, $\{n_1 : 40, n_2 : 20\}$ has the most significantly greater hypervolume indicator trials than all other swarm size compositions. Interesting to note is that this configuration is pigmented as black in the previously mentioned Figure B.8. $\{n_1 : 15, n_2 : 15\}$ is seen as the worst swarm size for this archive criterium, since it consists of only 1.9% significantly greater results than $\{n_1 : 20, n_2 : 40\}$, and none for the other two remaining compositions. Likewise, all other compositions have high percentages against $\{n_1 : 15, n_2 : 15\}$, such as 56.5%, 23.1%, and 68.5%. It becomes important to note that the balanced $\{n_1 : 30, n_2 : 30\}$ is slightly worse than $\{n_1 : 40, n_2 : 20\}$ on this measure. Although it is directly unknown why, this suggests that extra particles searching for alignments is beneficial. This sort of notion

	$\{n_1 : 30, n_2 : 30\}$	$\{n_1 : 15, n_2 : 15\}$	$\{n_1 : 20, n_2 : 40\}$	$\{n_1 : 40, n_2 : 20\}$
$\{n_1 : 30, n_2 : 30\}$	-	56.5%	23.1%	0%
$\{n_1 : 15, n_2 : 15\}$	0%	-	0%	0%
$\{n_1 : 20, n_2 : 40\}$	0%	16.7%	-	0%
$\{n_1 : 40, n_2 : 20\}$	15.7%	87.0%	57.4%	-

Table 4.16: AM-MGPSO Swarm Size Comparison by Aligned Characters (% greater; S_{n_1})

is complemented by the poor performance of $\{n_1 : 20, n_2 : 40\}$ with comparison to $\{n_1 : 30, n_2 : 30\}$ and $\{n_1 : 40, n_2 : 20\}$, and even $\{n_1 : 15, n_2 : 15\}$ against $\{n_1 : 20, n_2 : 40\}$.

The subsequent Table 4.16 gives the percentage of trials with significantly greater aligned characters, divided between all swarm sizes. Only the global best results from swarm S_{n_1} were considered, as it employs f_1, which calculates the number of aligned characters. From what could be expected, the columns follow a decreasing fashion from $n_1 = 15$ to $n_1 = 40$. That is to say, when comparing against $\{n_1 : 30, n_2 : 30\}$, only $\{n_1 : 40, n_2 : 20\}$ manages to obtain some significant outcomes. This occurs for all competitors where the n_1 value is higher than the one being compared against. It is demonstrated that $\{n_1 : 40, n_2 : 20\}$ features the best results, with $\{n_1 : 15, n_2 : 15\}$ in last. This is reasonable, since the increase in n_1 should give rise to more opportunities for an improved global best output in S_{n_1}.

Similarly to Table 4.16, Table 4.17 expresses outcomes from only the global best of S_{n_2}, with percentages of trials with significantly less inserted indels by various swarm size experiments. The greatest percentage spotted is 8.3% for $\{n_1 : 30, n_2 : 30\}$ against $\{n_1 : 15, n_2 : 15\}$. This is interesting, given that $\{n_1 : 20, n_2 : 40\}$ has more particles applied to minimizing the number of inserted indels, yet only achieves 7.4% of trials with significantly less inserted indels against $\{n_1 : 15, n_2 : 15\}$. However, $\{n_1 : 20, n_2 : 40\}$ is also seen with 1.9% against $\{n_1 : 40, n_2 : 20\}$, unlike $\{n_1 : 30, n_2 : 30\}$ with the lesser 0.9% opposing the same swarm size. Although it is unclear why $\{n_1 : 30, n_2 : 30\}$ has a higher percentage, it is clearly shown that $\{n_1 : 15, n_2 : 15\}$ performs the worst among the applied swarm sizes for trailing indels, given its high columns values. Likewise, $\{n_1 : 30, n_2 : 30\}$ and $\{n_1 : 20, n_2 : 40\}$ perform considerably better than $\{n_1 : 40, n_2 : 20\}$ on only a very slight amount of trials.

It can be seen that from the columns $n_2 = 15$ to $n_2 = 20$ the amount of significantly different trials decrease, and from $n_2 = 30$ to $n_2 = 40$ the comparisons are at 0%.

	$\{n_1 : 30, n_2 : 30\}$	$\{n_1 : 15, n_2 : 15\}$	$\{n_1 : 20, n_2 : 40\}$	$\{n_1 : 40, n_2 : 20\}$
$\{n_1 : 30, n_2 : 30\}$	-	8.3%	0%	0.9%
$\{n_1 : 15, n_2 : 15\}$	0%	-	0%	0%
$\{n_1 : 20, n_2 : 40\}$	0%	7.4%	-	1.9%
$\{n_1 : 40, n_2 : 20\}$	0%	5.6%	0%	-

Table 4.17: AM-MGPSO Swarm Size Comparison by Inserted Indels (% less; S_{n_2})

	$\{n_1 : 30, n_2 : 30\}$	$\{n_1 : 15, n_2 : 15\}$	$\{n_1 : 20, n_2 : 40\}$	$\{n_1 : 40, n_2 : 20\}$
$\{n_1 : 30, n_2 : 30\}$	-	0%	5.6%	4.6%
$\{n_1 : 15, n_2 : 15\}$	75.9%	-	67.6%	80.6%
$\{n_1 : 20, n_2 : 40\}$	9.3%	0%	-	14.8%
$\{n_1 : 40, n_2 : 20\}$	4.6%	0%	9.3%	-

Table 4.18: AM-MGPSO Swarm Size Comparison by Infeasibility (% less)

This suggests that although increasing n_2 may lessen the amount of inserted indels, there is a point where more particles will not provide additional benefit.

Lastly, Table 4.18 describes the percentage of significantly less infeasible trials among the swarm sizes. Row $\{n_1 : 15, n_2 : 15\}$ has the highest percentages because the particle total is comparatively the lowest to begin with. However, what remains interesting is that some of the other swarm size configurations of the same total have non-zero percentages among one another. Row $\{n_1 : 20, n_2 : 40\}$ is seen with the second-highest values, with $\{n_1 : 40, n_2 : 20\}$ following, and $\{n_1 : 30, n_2 : 30\}$ the lowest. All of these mentioned compositions have the same total particles, yet $\{n_1 : 30, n_2 : 30\}$ is found with the least considerably less infeasible as compared with $\{n_1 : 40, n_2 : 20\}$ and $\{n_1 : 20, n_2 : 40\}$, each of which specialize more so on a particular objective. With row $\{n_1 : 20, n_2 : 40\}$ having the second-highest percentages, this could be drawn from it being an easier task to not insert indels, rather than to find alignments. However, under that notion, $\{n_1 : 30, n_2 : 30\}$ should not be in last place as it has more particles in n_2, and less in n_1 than $\{n_1 : 40, n_2 : 20\}$. Also, $\{n_1 : 40, n_2 : 20\}$ should then not have 9.3% of trials with less inserted indels than $\{n_1 : 20, n_2 : 40\}$, compared with only 5.6% from $\{n_1 : 30, n_2 : 30\}$. Therefore a strict conclusion cannot be drawn from these results, other than the highest percentages for $\{n_1 : 15, n_2 : 15\}$ likely being due to having the lowest total amount of particles.

4.5.2 Coefficient Configuration

Varying coefficient parameters allow for changes in the prominence of a particular component within the velocity update equation. As such, particles could be moved in a different location, and by a different quantity. This gives rise to a potential of various outcomes, depending on the configuration of coefficients.

Figure B.9 presents the best archives for AM-MGPSO on each reference set, with colours and shapes for differing coefficients. Note should be taken that the archives are different than Figure B.8, because the reference point for hypervolume indicator values was taken among different coefficient executions, not swarm size. Some of the individual graphs have black triangles that are found nearing the top-left corner, or closer to the nadir point. This belongs to the coefficient set 9 from Table 3.3. Since the figure contains the best archives, set 9 somewhat attaches itself to a non-superior status. The other archive points are seemingly close to one another a majority of the time, with some infrequent anomalies occurring from red stars (set 2), and red triangles (set 7).

Table 4.19 seems to agree with the poor results of coefficient set 9 and 2, which host some of the highest percentages in their respective columns. Set 7 however may have been a rare anomaly for the graph, as its column values are the second-lowest out of the rest, and its row the highest. Without a doubt, it is noticed that 9 has the worst column values, meaning many other configurations were much better. However, when glancing at its respective row values, although they are low, they are not as inferior as composition 2 and 3, which host some zero entries. Thus, although all of the remaining coefficients seemingly performed better than 9, it did not express absolute lackluster performance. Overall, coefficients 2, 3, and 9 performed poorly, with good measures arising from 5, 7, and 8. Comparing with Table 3.3, it appears that perhaps the change of c_1 value should be focused on, as set 2 and 3 have this parameter decreased, but by much more than 5. Also, it is possible that a lower c_3 or ω with respect to remaining parameters is beneficial, as is the case with compositions 5, 7 and 8. Likewise, the worst set 9 has all parameters decreased, which is presumably the reason behind its poor performance.

When pursuing global best alignments in swarm S_{n_1}, it is row 8 of Table 4.20 that is superior to all remaining arrangements. However, column 5 quantifies much lower values than column 8. The greatest proportions are received again in column 9, although much worse than in the previous Table 4.19. Sets 1, 2, 3, 4 and 9 all had inadequate executions, while the similar 5, 7, and 8 achieved the most significantly

	1	2	3	4	5	6	7	8	9
1	-	25%	16.7%	4.2%	2.1%	6.2%	4.2%	4.2%	50%
2	6.2%	-	2.1%	0%	0%	0%	0%	0%	35.4%
3	6.2%	10.4%	-	0%	0%	2.1%	2.1%	0%	45.8%
4	8.3%	27.1%	20.8%	-	2.1%	8.3%	2.1%	6.2%	52.1%
5	25%	31.2%	29.2%	14.6%	-	14.6%	4.2%	8.3%	66.7%
6	2.1%	18.8%	12.5%	4.2%	0%	-	6.2%	2.1%	56.2%
7	22.9%	47.9%	31.2%	22.9%	6.2%	27.1%	-	10.4%	62.5%
8	10.4%	33.3%	22.9%	12.5%	10.4%	10.4%	2.1%	-	58.3%
9	10.4%	12.5%	16.7%	6.2%	2.1%	8.3%	2.1%	8.3%	-

Table 4.19: AMPSO Coefficient Configuration Comparison by Hypervolume Indicator (% greater)

	1	2	3	4	5	6	7	8	9
1	-	20.8%	12.5%	2.1%	0%	2.1%	0%	2.1%	72.9%
2	6.2%	-	0%	2.1%	0%	6.2%	6.2%	0%	56.2%
3	12.5%	10.4%	-	0%	0%	4.2%	4.2%	0%	68.8%
4	8.3%	31.2%	18.8%	-	0%	8.3%	2.1%	2.1%	70.8%
5	29.2%	41.7%	29.2%	14.6%	-	16.7%	12.5%	8.3%	77.1%
6	4.2%	25%	16.7%	2.1%	0%	-	2.1%	2.1%	77.1%
7	22.9%	41.7%	27.1%	10.4%	4.2%	12.5%	-	8.3%	72.9%
8	35.4%	45.8%	33.3%	22.9%	10.4%	31.2%	18.8%	-	77.1%
9	2.1%	6.2%	4.2%	2.1%	2.1%	2.1%	0%	0%	-

Table 4.20: AM-MGPSO Coefficient Comparison by Aligned Characters (% greater; S_{n_1})

greater aligned characters in their global bests of sub swarm S_{n_1}. The results found in Table 4.20 are comparable to the previously discussed table. With identical coefficient compositions possessing superior outcomes, it exemplifies their ability to obtain good quality solutions, both throughout the archive and global best alignments.

Table 4.21 expresses the ratio of significantly less inserted indels within the sub swarm S_{n_2}. Immediately, many 0% values are visible in all columns except 9, however this is due to the ease in ability for AM-MGPSO to obtain perfect scores on this criteria. Thus, it is impossible to obtain significantly less inserted indels between coefficient values if they are the same throughout experimentation. In column 9, multiple exact values of 72.9% are visible. Thus it is likely that in at least 72.9% of executions, coefficient composition 9 did not obtain a perfect trailing indel score. Its parameter values are all decreased, meaning that particles had little intention on

	1	2	3	4	5	6	7	8	9
1	-	0%	0%	0%	0%	0%	0%	0%	72.9%
2	0%	-	0%	0%	0%	0%	0%	0%	72.9%
3	0%	0%	-	0%	0%	0%	0%	0%	72.9%
4	0%	0%	0%	-	0%	0%	0%	0%	72.9%
5	0%	0%	0%	0%	-	0%	0%	0%	72.9%
6	0%	0%	0%	0%	0%	-	0%	0%	72.9%
7	0%	0%	0%	0%	0%	0%	-	0%	72.9%
8	0%	0%	0%	0%	0%	0%	0%	-	72.9%
9	0%	0%	0%	0%	0%	0%	0%	0%	-

Table 4.21: AM-MGPSO Coefficient Comparison by Inserted Indels (% less; S_{n_2})

	1	2	3	4	5	6	7	8	9
1	-	0%	2.1%	41.7%	33.3%	0%	45.8%	4.2%	43.8%
2	52.1%	-	12.5%	62.5%	43.8%	20.8%	72.9%	12.5%	62.5%
3	58.3%	39.6%	-	85.4%	72.9%	45.8%	93.8%	25%	91.7%
4	29.2%	4.2%	0%	-	2.1%	8.3%	16.7%	0%	14.6%
5	41.7%	14.6%	0%	41.7%	-	29.2%	60.4%	0%	37.5%
6	50%	0%	4.2%	58.3%	41.7%	-	68.8%	10.4%	56.2%
7	20.8%	2.1%	0%	0%	0%	6.2%	-	0%	4.2%
8	56.2%	31.2%	0%	87.5%	54.2%	43.8%	95.8%	-	81.2%
9	25%	6.2%	0%	12.5%	0%	12.5%	25%	0%	-

Table 4.22: AM-MGPSO Coefficient Comparison by Infeasibility (% less)

repositioning towards their personal best, global best, or the archive guide.

To conclude this section on the differences between coefficient combinations, infeasibilities must be discussed. Table 4.22 shows the proportion of significantly less infeasibility throughout experimentation between selected coefficient values. Parameter sets 9, 7, 5, 4, and 1 all received poor results on this criterium, as is seen by the large values in their respective columns. This is interesting, given that 5 and 7 received sufficient findings when compared on alignment and hypervolume indicator measurements. Among the bad outcomes, all but coefficient set 1 have reduced c_3 values. Additionally, set 9 again contains the largest measures in its column. Its small coefficients may have pushed particles into infeasible territory, and also may have made it more difficult for a recovery to take place. Although other parameters have seen significantly less infeasible particles than sets 5 and 7, it cannot be denied that their other results are enticing, making them, along with set 8, among the most superior coefficient configurations utilized on the reference sets.

4.5.3 Conclusion

AM-MGPSO was successful in finding solutions to the MSA problem, and did not encounter major problems with feasibility. Comparisons were made between individual swarm sizes, and coefficient configurations using the measurements of hypervolume indicator for archive quality, global best alignments and inserted indels on their respective sub swarms, and infeasibility. In the matter of swarm size, $\{n_1 : 40, n_2 : 20\}$ and $\{n_1 : 30, n_2 : 30\}$ performed the best for quality of solutions, with the former being ultimately superior. Although infeasibility was higher for $\{n_1 : 40, n_2 : 20\}$ when compared with $\{n_1 : 30, n_2 : 30\}$, this did not provide any consequence in outcome.

When speaking in terms of coefficients, set 5, 7, and 8 of Table 3.3 assembled excellent findings. On the measurements for hypervolume indicator and global best alignments, these sets of values have obtained the best outcome out of the remaining 6 configurations. Additionally, they had no troubles finding the optimal value for global best inserted indels. However, a topic worth noting is the large infeasibilitity percentages against both columns 5 and 7 on Table 4.22. 8 does not feature these high percentages, and performs relatively similar in terms of outcome quality.

4.6 AM-MGPSO and B-MGPSO

B-MGPSO and AM-MGPSO have the ability to operate with multiple, individual objective functions at the same time. However, as the naming would suggest, these algorithms involve the combination of different algorithms with MGPSO. In turn this leads to unique methods of processing the swarms. As was explained in Section 4.4, B-MGPSO is never feasible after initial generation. It therefore has the effects of a random search, with a very limited number of searches since the maximum number of particles chosen in total never surpassed 60.

Alternatively, AM-MGPSO had an overall average amount of infeasible particles of 3133, which is low even for 2500 iterations on the lowest amount of particles used, being 30 (4.2%). This allows for an increased potential of additions to the archive, along with updates to the global bests of both swarms. Overall this means increased potential for much better results. In general, comparisons can be summarized in a simple manner for concision: for all of the criteria measures used, whether it be by hypervolume indicator, alignments on swarm S_{n_1}, inserted indels on swarm S_{n_2}, or infeasibility, AM-MGPSO was always significantly greater than B-MGPSO for the first two criterions, while consistently following this pattern for considerably less inserted indels, and infeasibility.

Thus, when deciding between employment of AM-MGPSO and B-MGPSO on large reference sets, the outcome is simple. Likely due to the curse of dimensionality, the B-MGPSO does not feature good performance perhaps from its BPSO heritage. For reference set 3, there are 14 individual sequences, with the highest length being 465. This means the position and velocity vectors are each matrices of size 14 x 558, which is reasonably large. The advantage of AM-MGPSO comes from its AMPSO origin, whereby only 4 dimensions are used for position and velocity, which in turn generate a bit matrix using the generation function.

4.7 Inter-algorithm Comparison

For BPSO and AMPSO, weights are used to alter the overall focus on outcome. For $w_1 > w_2$, the main direction is on maximizing the number of alignments. Alternatively when $w_2 > w_1$, particles are concentrated towards optimizing the amount of trailing indels. Thus, when $w_1 = 1$ $w_2 = 0$, there is undivided attention for the objective function f_1, and vice versa for f_2.

MGPSO is comprised of two separate sub swarms that solely focus on an individual

objective function, similarly to what occurs if only one weight value is zero in AMPSO and BPSO. Thus, the single-objective and multi-objective problem formations can be compared by concentration on each individual objective functions. Comparisons between AMPSO and BPSO have already been made in Section 4.3, and among AM-MGPSO and B-MGPSO in Section 4.6. Thus, these semblances will not be made in this section.

This section is divided into four distinct areas. In Section 4.7.1, BPSO and B-MGPSO are compared, while the same occurs for BPSO and AM-MGPSO in Section 4.7.2. Afterwards, Section 4.7.3 features AMPSO against B-MGPSO, while Section 4.7.4 highlights the difference between algorithms AMPSO and AM-MGPSO.

4.7.1 BPSO and B-MGPSO

To begin, BPSO and B-MGPSO will be compared. Both algorithms performed equally as poor for infeasibility, as it was always 100% after initial generation. Thus, only weight configuration (strictly for BPSO) and swarm size were contrasted, as these are the parameters that would be of importance in affecting the outcome. In terms of BPSO and weight configuration, $\{w_1 : 1, w_2 : 0\}$ was found to have best results for alignment, and $\{w_1 : 0, w_2 : 1\}$ for inserted indels. Likewise, with swarm sizes it is $n = 50$ that produced best results. Based on B-MGPSO and swarm size, $\{n_1 : 40, n_2 : 20\}$ was superior for aligned characters, and $\{n_1 : 20, n_2 : 40\}$ for inserted indels. Thus, statistical testing among global bests is performed between the set of parameters that bring the best results.

Out of the twelve sequence sets which were utilized for experimentation, BPSO was statistically better than B-MGPSO for 7 with respect to alignment scores, and 8 for trailing indels. Likewise, B-MGPSO was never significantly better for either of the criterions. These results are reasonable, since for an algorithm that acted like a random search, a higher n leads to more possibilities of finding solutions. In the experimentations used for this study, the highest n of 50 was used for AMPSO and BPSO. For AM-MGPSO and B-MGPSO, it was only 40 for each respective sub swarm. Similarly to how $n = 50$ outclassed $n = 40$ in Section 4.1.2, BPSO excelled against B-MGPSO.

4.7.2 BPSO and AM-MGPSO

Unlike BPSO, AM-MGPSO did not execute with nearly the same amount of infeasible particles. Therefore throughout its iterations, AM-MGPSO had more opportunities

to arrive at a new global best solution. In Section 4.5.1, it was shown that $\{n_1 : 40, n_2 : 20\}$, along with coefficient set 8 was the best for finding aligned characters on S_{n_1}. By the same token, $\{n_1 : 20, n_2 : 40\}$ performed greatest for maximizing trailing indels on S_{n_2}, and so will be used along with coefficient set 8 for inserted indel comparisons. Exact comparative parameters from Section 4.7.1 will be used here for BPSO.

For every sequence reference set and on both criterions, AM-MGPSO performs significantly better than BPSO. Given the low infeasibility of the former, this result comes as no surprise. In both of the angular modulated algorithms, BPSO does not feature respectable outcomes against its low-dimensional PSO counterpart. Having no feasible solutions after initial generation hampers its ability to find good objective measurements, and as stated previously, produces what could be considered as a random search.

4.7.3 AMPSO and B-MGPSO

Similarly to the BPSO, which AMPSO was compared with in Section 4.3, B-MGPSO also executes with a total capacity of infeasible particles, for all experimental compositions. The main difference between B-MGPSO and BPSO however is the swarm sizes, where the maximum used in the latter algorithm was 50, and in the former only a peak of 40 for individual sub swarms. Since these two acted primarily as a random search, it is expected that AMPSO will perform even better than B-MGPSO then as with BPSO. Exact parameter configurations will be used for B-MGPSO as was utilized in Section 4.7.1.

As far as AMPSO, $\{w_1 : 1, w_2 : 0\}$, a swarm size of 50, and the coefficient set 1 was found to have the best results for constructing alignments. In the same way for inserted indels, $\{w_1 : 0, w_2 : 1\}$, $n = 50$, and coefficient composition 2 are used. The results returned are just as appealing as when AM-MGPSO and B-MGPSO were compared in Section 4.6. For all of the reference sets, AMPSO manages to be substantially better than B-MGPSO on both the global bests of alignments and inserted indels. Thus, AMPSO is an overall algorithmic improvement over B-MGPSO for the chosen reference sets, and the selected experimental parameters.

4.7.4 AMPSO and AM-MGPSO

Both of these angular modulated algorithms did not feature an infeasibility rate of 100%, making them a highly competitive pair with regards to the algorithms chosen

in this work. Best overall configurations for the two methods have already been selected, and can be found for AMPSO in Section 4.7.3, and AM-MGPSO in Section 4.7.2. Interestingly, AMPSO considerably surpasses its multi-guided counterpart on 5 reference sequence sets on number of alignments, while AM-MGPSO does the same for only 1 conversely. However in connection with inserted indels, neither algorithm is significantly better than the other, due to their perfect performance in ensuring indels were trailing.

Although initially unexpected, AM-MGPSO does not focus on the global best of its sub swarms throughout the entire iteration process. This is because the lambda value is linearly increasing from 0 to 1, and so from Equation 2.7, the global best is concentrated more in later iterations than in previous. This contrasts AMPSO, which has full attention towards making alignments throughout the entire procedure.

This outcome also shines light on the purpose of the MGPSO in general, which is to find the pareto-optimal front. Although having a better global best may improve the respective extremal archive values, too much focus on finding the universally best objective output may distract from other, non-extremal additions to the archive. As such, a balance between the search for a global best and fresh archive additions must be maintained. This is not the worry of AMPSO, since the only objective is to obtain a overall global best, which it did well for alignments in comparison to AM-MGPSO, and perfectly for trailing indels.

4.8 Comparison with ProbCons

In this section, both AMPSO and AM-MGPSO are compared with ProbCons. Results will express whether these variants have the ability to compete alongside a state-of-the-art algorithm. As was done in Section 4.7, results from superior alignment parameters will be used. ProbCons concludes with one solution, and so comparisons are made only with global best outcomes. BPSO and B-MGPSO have not been selected for comparison because of their 100% infeasibility rate.

4.8.1 AM-MGPSO

AM-MGPSO performed second best among all the employed PSO variants for the MSA problem. In this section, AM-MGPSO is compared with ProbCons, one of the state-of-the-art algorithms for MSA. MGPSO parameters which provide the best alignment scores will be used. Thus, results are utilized where swarm size is

	Inserted Indels			Alignments		
Sequence Set	Mean	Std. Dev.	ProbCons	Mean	Std. Dev.	ProbCons
1	**12.57**	3.91	39	112.2	4.48	**146**
2	**23.17**	7.14	1608	243.07	3.88	**300**
3	**36.97**	11.28	6346	3002.63	11.03	**3399**
4	**19.03**	11.22	1460	**164.53**	3.93	162
5	**8.3**	2.94	277	**115.53**	2.33	112
6	**12.8**	4.1	5446	301.63	4.16	**392**
7	**5.37**	2.13	174	301.8	5.57	**362**
8	**12.43**	9.87	531	911.4	12.2	**1308**
9	**5.83**	2.12	52	355.17	4.68	**492**
10	**11.73**	9.06	727	147.0	2.91	**202**
11	**14.03**	5.98	1717	304.27	3.75	**414**
12	**23.3**	10.37	62	148.7	3.93	**216**

Table 4.23: AM-MGPSO and ProbCons Comparison

$\{n_1 : 40, n_2 : 20\}$ and with coefficient set 8 (Table 3.3). Global best results illustrated in Table 4.23 come from swarm S_{n_1}.

Table 4.23 displays the mean and std. dev. for AM-MGPSO, along with the ProbCons value for both inserted indels and alignments. AM-MGPSO clearly has each mean far lower than the ProbCons result for inserted indels. No average went above 36.97, even when the highest for ProbCons reaches 6346. This suggests that ProbCons may have a tendency to insert too many indels. However, there is a clear lack of substantial results on aligned characters. Only two values of the mean are in favour of AM-MGPSO, and even then, are only above ProbCons by a maximum of 3.53 alignments. At worst, AM-MGPSO underperformed by 396.6 aligned characters.

The results from Table 4.23 express that although AM-MGPSO was competitive in its lack of want to insert indels, it did not achieve that many alignments. Following the lead of ProbCons, inserting more indels in strategic positions likely would have led to more characters aligned in the outcome. This would have proven beneficial for AM-MGPSO, which follows a pattern of seemingly too little inserted indels. That being said, for sequence sets 4 and 2, where ProbCons had many more inserted indels, AM-MGPSO still obtained more alignments with much less insertions. Notice that specifically on sequence set 4, AM-MGPSO on average inserted 29.33 indels against 1460 from ProbCons.

Overall, when focusing solely on alignments, ProbCons is superior to AM-MGPSO on 10 of the 12 selected sequence sets. The latter was found to be more efficient with its decreased insertion of indels, but possibly at the expense of further aligned

	Inserted Indels			Alignments		
Sequence Set	Mean	Std. Dev.	ProbCons	Mean	Std. Dev.	ProbCons
1	**33.17**	12.9	39	113.8	5.09	**146**
2	**60.9**	24.26	1608	245.0	4.98	**300**
3	**67.3**	29.11	6346	3003.7	14.64	**3399**
4	**29.33**	11.25	1460	**169.37**	5.57	162
5	**16.5**	6.76	277	**114.27**	2.32	112
6	**29.17**	16.06	5446	301.87	3.68	**392**
7	**6.23**	2.4	174	303.13	4.42	**362**
8	**21.13**	14.26	531	916.33	8.60	**1308**
9	**9.53**	4.58	52	357.83	8.79	**492**
10	**20.33**	11.71	727	148.47	2.76	**202**
11	**26.3**	12.16	1717	306.6	4.26	**414**
12	**40.93**	16.78	62	152.87	6	**216**

Table 4.24: AMPSO and ProbCons Comparison

characters.

4.8.2 AMPSO

Among the four PSO versions used in this study, AMPSO achieved the best results with its global best. Throughout execution, it featured both a low infeasibility rate and low-dimension position vectors. These contributed to its comparably superior performance. In this section, AMPSO is equated against a state-of-the-art algorithm named ProbCons. Previously found superior alignment parameters will be used, or more specifically, weights $\{w_1 : 1, w_2 : 0\}$, a swarm size of 50, and coefficient set 1 (Table 3.1).

In the inserted indels section of Table 4.24, it is evident that AMPSO on average performs much better at ensuring indels are trailing. However, on sequence set one, standard deviation values suggest that at times, inserted indels may rise above that of ProbCons. While AMPSO achieves excellent results in the form of low inserted indels, the same cannot be said for alignments. Among the 12 sequence sets, only 2 have an average where AMPSO is better than ProbCons. Using the same measurement, the worst difference in AMPSO mean and ProbCons value is 395.3 on sequence set 4. Notice that the difference between the two MSA methods on inserted indels is 1430.67.

Given that ProbCons has more inserted indels and alignments than the average AMPSO execution, it insinuates that perhaps more indels need be inserted for better

alignment results. This is strange, given that in the weight values $\{w_1 : 1, w_2 : 0\}$ used, w_2 is zero. Note that w_2 correlates with fitness calculations concerning the amount of trailing indels. Being zero, there is no interest into how many indels are inserted, only if the number of aligned characters continue to rise. A further consideration may be to have the particle increasingly explore the search space.

When debating on the criterium of alignments, it is clear that ProbCons is superior than AMPSO. Even with its weight w_2 being zero, AMPSO on average did not attempt to insert nearly as many indels as ProbCons. This may be a large factor contributing to the lower amount of alignments produced. If more indels are to be inserted, then there are more opportunities to find a greater pairing of characters in each sequence. This also comes with the trade-off of including too many indels for the same quantity of alignments. In the case of AMPSO, it seemingly is not inserting enough indels. This could be limiting its ability to find more aligned characters, and is why the performance is lackluster when compared with ProbCons. As suggested previously, it may be of use to have AMPSO explore the vast search space before exploiting the knowledge it already has.

4.8.3 Conclusion

ProbCons was compared against AMPSO and AM-MGPSO, both of which provided superior solutions than their binary counterparts. The metrics used consisted of the average number of aligned characters, and inserted indels. In general, the PSO variants have on average, many more trailing indels than ProbCons. For example, in Table 4.24 on sequence set 6, AMPSO had an average of 29.17 inserted indels, while ProbCons had 5446. However, where both AMPSO and AM-MGPSO lack are in the alignment measurement. On the same sequence set 6, AM-MGPSO achieved an average of 301.63 alignments, and ProbCons reached 392. Although excess indels are not being inserted, it was suggested that not enough are inserted either. If more are to be inserted, there are more opportunities for characters to be aligned further into the sequences. Inserting indels is a trade-off that both PSO versions have seemingly to improve upon.

Chapter 5

Conclusion

This work consists of applying four variations of the PSO, namely BPSO, AMPSO, B-MGPSO, and AM-MGPSO, towards the MSA problem. MSA can be configured both as a single-objective and multi-objective problem. For the latter, two objective functions, specifically the number of aligned characters, and number of spaces are utilized. In addition, criterions used for multi-objective performance evaluation were hypervolume indicator calculations, global best alignments, global best inserted spaces, and infeasibility. For the single-objective approach, two objective functions are still applied, but instead in the form of a weight aggregated function. Moreover, compared measures for the single-objective problem include global best alignments, global best trailing spaces, and infeasibility. AMPSO and BPSO both use the single-objective form of MSA, where introspective comparisons are drawn between combinations of 3 weights, 3 swarm sizes, and 5 coefficients. Likewise, B-MGPSO and AM-MGPSO use the multi-objective format, and contrast between assortments of 4 swarm sizes, and 9 coefficient values. A total of 12 sequence sets are used from BaliBase, a benchmark for MSA. Out of those selected, half are from RV11 with the remaining half from the RV12 reference set. These sets consist of very dissimilar (<20%) and medium to very dissimilar (20%-40%) sequences.

It was found that both BPSO and AMPSO outcomes differ drastically when their weight values w_1 and w_2 are altered. With an increase of w_1, more alignments are found. Likewise, when w_2 is increased, the number of trailing spaces grow. This is reasonable, since in fitness evaluation, w_1 is attached to f_1, which measures the number of aligned characters. In addition, w_2 is joined with f_2, calculating the quantity of trailing spaces. On discussion of swarm size, a larger amount of particles provide better results. This suggests that a refinement of solutions is possible with more particles. It should be noted that while both these determinations are true

for BPSO and AMPSO, the former also features an infeasibility rate of 100% after initial generation. Thus, since the global best is not updated when a particle is infeasible, a change in coefficients is not important for BPSO. However, for AMPSO, feasibility rates are high, and so this is not the case. On comparing different coefficient configurations, it was found that having an $\omega > 0.7$ is beneficial for alignments. Moreover, the compositions that scored the best experiments had $c_1 > 1$, and $c_1 > c_2$, with $c_2 > 0$. This suggests that particles obtain better solutions when concentrating on their personal best, but with some focus still on the global best.

Similarly to BPSO, B-MGPSO had an infeasibility rate of 100%. As stated previously, this means coefficients have no role in any of the measured criteria. When B-MGPSO has more particles, it performs better than the contrary. This is the same with AM-MGPSO, however when it comes to archive quality, it was preferable to have more particles in swarm S_{n_1} rather than S_{n_2}. This gives a concentration on making alignments, in preference over ensuring spaces are not inserted. It should be noted that AM-MGPSO was regularly able to achieve a perfect global best inserted space score, which is comparable to findings of AMPSO. On the topic of selecting coefficients, for AM-MGPSO it is by far the poorest to use values which are all below 0.53. At the same time, best performance is spotted from coefficients where c_2 is > 1.7, along with a low c_3 (< 0.5). However, an arrangement that had only a low inertia (ω; < 0.15) with high other parameters (> 1) performed well, and also did not execute with many infeasible particles.

Comparative analysis between AMPSO and BPSO was performed, where it was found that the former is significantly better in almost all trials for making alignments, and always a substantial improvement for trailing spaces. Likewise, similar comparisons are drawn between AM-MGPSO and B-MGPSO. It became evident that AM-MGPSO is always better than B-MGPSO, among all the criteriums used. Further investigations were constructed into the results of all four algorithms. Since AMPSO and BPSO do not hold an archive, unlike the MGPSO variants, only global bests are compared among previously found superior parameters. The order of ranking from worst to best go as follows: B-MGPSO, BPSO, AM-MGPSO, and AMPSO. BPSO was not last because its highest swarm size reached 50, whereas B-MGPSO finished at 40. They both acted as a random search due to their complete infeasibility after generation, and so a higher swarm size was the determining factor between these two algorithms. Among AM-MGPSO and AMPSO, it is the latter that provided improved global best conclusions. This is very likely due to the MGPSO variant not completely focusing on finding a global best. A linearly increasing lambda parameter

was used throughout iterations, which from Equation 2.7, means only later iterations concentrate on traversing towards the global best position. Discordantly, AMPSO has undivided concentration on searching for a global best.

When searching for solutions to the MSA problem, a decision must first be made about the problem approach. If it is optimal to find one best solution, then the single-objective formation should be chosen. If a series of potential answers are required, then the multi-objective MSA problem should be utilized. However, caution must be taken for both BPSO and B-MGPSO, as their constant infeasibility inhibits favourable outcomes. It is likely that this occurs due to the curse of dimensionality, as particle position and velocities are both high dimension matrices. From comparisons drawn in this study, the superior PSO competitors are AMPSO and AM-MGPSO.

Further observations are made among a state-of-the-art algorithm named ProbCons and the superior PSO variants. BPSO and B-MGPSO were not included in these contrasts due to their infeasibility. On discussion of inserted spaces, both AMPSO and AM-MGPSO are vastly superior than ProbCons. However, this is not the case for alignments. On the selected sequence sets, ProbCons found many more alignments than both AMPSO and AM-MGPSO on 10 of the 12 sets. This may have occurred due to a lack of inserting indels in the PSOs. As more spaces are inserted within the sequence, more characters may be aligned further along. A balance must be struck between the insertion of spaces and alignment of characters, of which the PSOs may need improvement.

This work concludes the introspective study and inter-algorithm comparison of four PSO variants applied for the MSA problem. Overall, AMPSO and AM-MGPSO performed the best among the four, with BPSO and B-MGPSO having a large infeasibility rate. However, upon comparisons with the state-of-the-art method ProbCons, both the superior AMPSO and AM-MGPSO fall short of delivering substantial alignments.

5.1 Future Studies

This study has concluded that both BPSO and B-MGPSO were not adequate on the selected sequences and parameters. Further investigation should inspect possible improvements for the aforementioned methods, such as a less strict constraint, or performance on smaller sequence-length sets. Likewise, it may be of benefit to use different λ values than one that is linearly increasing. This applies specifically for AM-MGPSO, which was not entirely infeasible throughout iterations and thus would

make use of various λ methods.

In addition, it may be of use to determine a method of ensuring better balance between inserting indels and finding alignments. Even when the weights are configured to give no penalty for spaces, not many are inserted. This may be the reasoning behind lackluster performance against ProbCons.

Appendix A

Sequences

In Table A.1, all of the reference sets used are listed. These sets were gathered from BaliBase [37], which is a benchmark used for the MSA problem. There are 12 in total, with 6 hand picked from the reference set RV11 (very dissimilar <20%) and the remaining from RV12 (medium to very dissimilar 20%-40%). Each of these sets have their quantity and sequence lengths listed, where the bold numbers denote highest length sequence in that set.

Number	Reference Set	Name	Quantity	Length
1	RV11	BB11001	4	83, 85, **91**, 86
2	RV11	BB11002	8	58, **193**, 83, 52, 83, 101, 134, 80
3	RV11	BB11005	14	421, 355, 412, 361, 429, 431, 427, 461, 382, 354, 329, 343, **465**, 412
4	RV11	BB11009	4	97, 109, **337**, 321
5	RV11	BB11013	5	54, 74, 51, 78, **101**
6	RV11	BB11026	7	76, 76, **906**, 94, 109, 98, 321
7	RV12	BB12003	8	64, 67, 60, **85**, **85**, 66, **85**, 58
8	RV12	BB12005	9	201, 211, 213, 202, 197, **234**, 199, 224, 200
9	RV12	BB12006	4	235, 220, **242**, 231
10	RV12	BB12009	5	67, 72, **201**, 70, 153
11	RV12	BB12014	9	65, 69, 68, 77, 49, 66, 79, 148, **254**
12	RV12	BB12020	4	118, **129**, 126, **129**

Table A.1: Experimental Sequences

Appendix B

Full Page Graphs

This chapter means to display full page graph illustrations used for comparisons. Both BPSO and AMPSO consist of graphs that are three-dimensional. Their x, y, and z axes respectively express the average alignment, average indel insertion, and average fitness value. The alternative charts from B-MGPSO and AM-MGPSO are two-dimensional, and illustrate the best archives.

Figures B.1 and B.3 show result averages between three unique weight values for AMPSO and BPSO, respectively. Each colour represents a separate weight configuration, which is listed for reference in Table B.1.

Colour	w_1	w_2
Red	1	0
Yellow	0.5	0.5
Blue	0	1

Table B.1: BPSO and AMPSO Weight Configuration Colours

Figures B.2 and B.4 illustrate differences between the swarm sizes of BPSO and AMPSO. It has a similar shape to their respective plots that compare different weight values (Figure B.1 and Figure B.3), but enforces a distinct colour mapping. Also, it should be noted that in Figure B.4, all entries where $\{w1 : 0, w2 : 1\}$ are omitted due to lack of variation in their location, and for greater insight into the other points. All colour references for Figures B.2 and B.4 are listed in Table B.2.

Similarly to Figures B.2 and B.4, Figure B.5 holds the same plot structure, but uses various colours for the representation of different coefficient configuration. Similarly to the aforementioned Figure B.4, entries with the weight values $\{w1 : 0, w2 : 1\}$ are absent. Table B.3 explains the colour variations of Figure B.5.

Colour	n
Blue	30
Green	40
Red	50

Table B.2: BPSO and AMPSO Swarm Size Colours

Colour	ω	c_1	c_2
Blue	0.7098150314023034	1.6788775458244407	0.899446463381824
Green	0.7861878289341226	1.2489605895810598	1.211314077689869
Red	0.7226988763584911	0.7877525185622731	1.3569405150479763
Yellow	0.7566563010222623	0.9360167168210383	0.8238319031198684
Magenta	0.3260770959583924	1.397180934100446	1.3464223101885027

Table B.3: BPSO and AMPSO Coefficient Configuration Colours

Colour	n_1	n_2
Blue	30	30
Red	15	15
Brown	20	40
Black	40	20

Table B.4: B-MGPSO and AM-MGPSO Swarm Size Colours

Figure B.6 illustrates distinctions between both AMPSO and BPSO global bests. Only two colours are used: blue and black, each of which correspond to AMPSO and BPSO respectively. Following is Figure B.7 and B.8, which displays the overall best archive by hypervolume calculation for B-MGPSO and AM-MGPSO. The colour variations each correspond to a specific set of swarm sizes, which is illustrated in Table B.4. To conclude, Figure B.9 also displays best archives by hypervolume indicator value, but compares among coefficient configurations. Varying colours and shapes are used, as can be seen in Table B.5.

#	Colour	Shape
1	Blue	Star
2	Red	Star
3	Brown	Star
4	Black	Star
5	Green	Star
6	Blue	Triangle
7	Red	Triangle
8	Brown	Triangle
9	Black	Triangle

Table B.5: B-MGPSO and AM-MGPSO Coefficient Configuration Colours

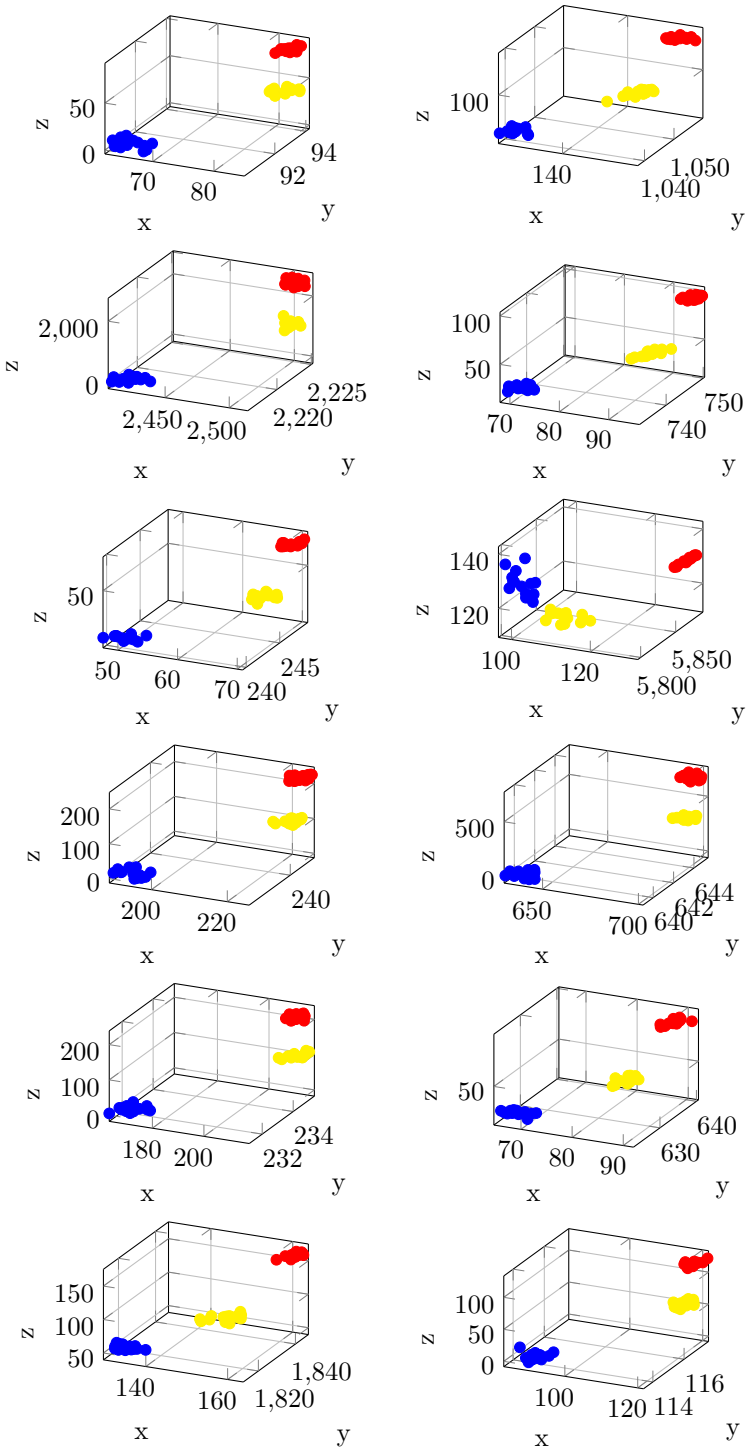

Figure B.1: BPSO Sequence Comparison by Weight

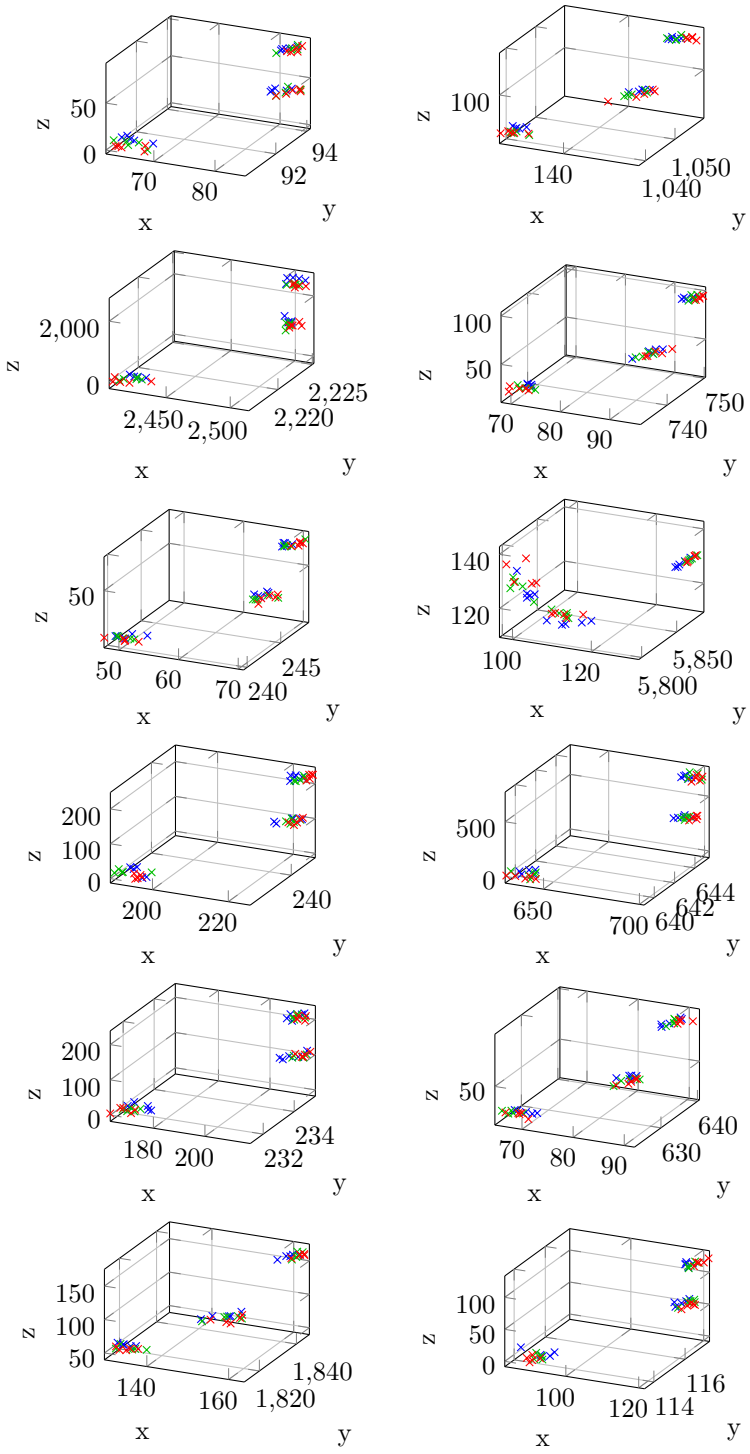

Figure B.2: BPSO Sequence Comparison by Swarm Size

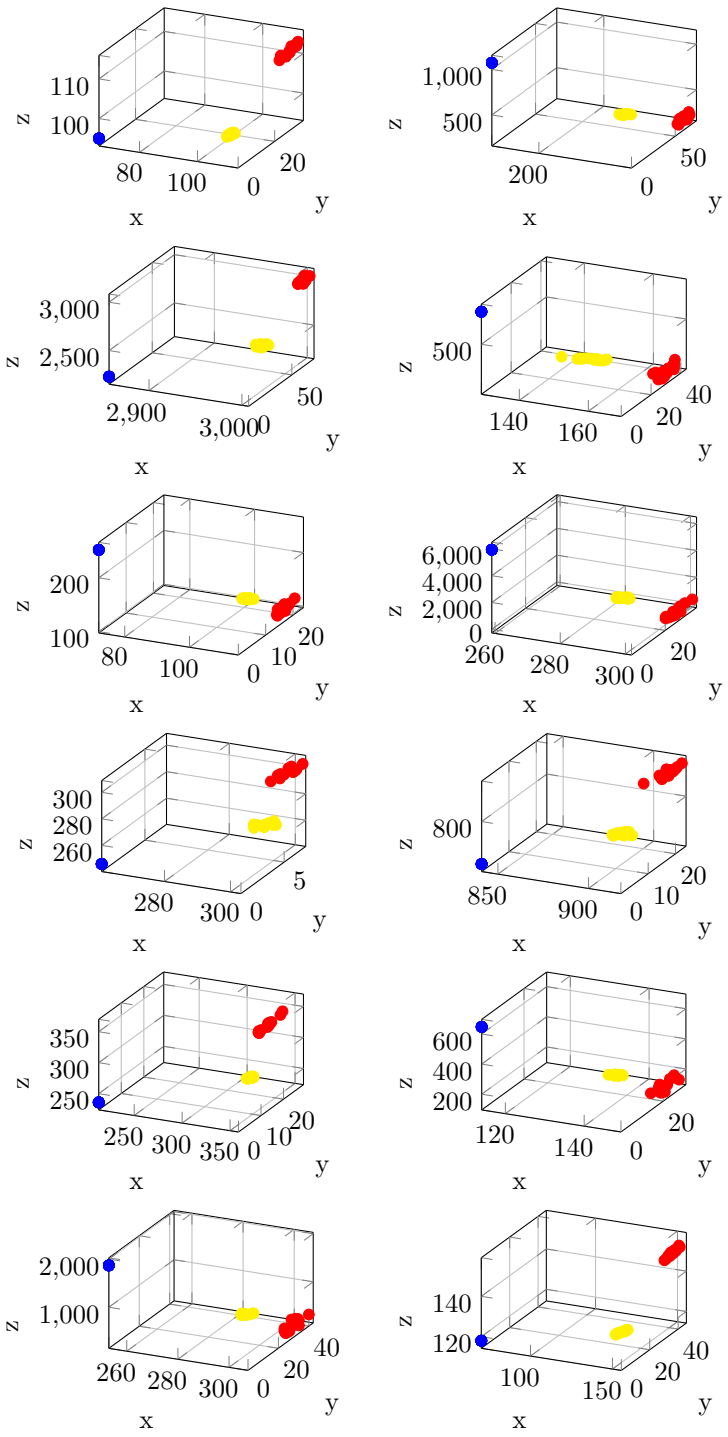

Figure B.3: AMPSO Sequence Comparison by Weight

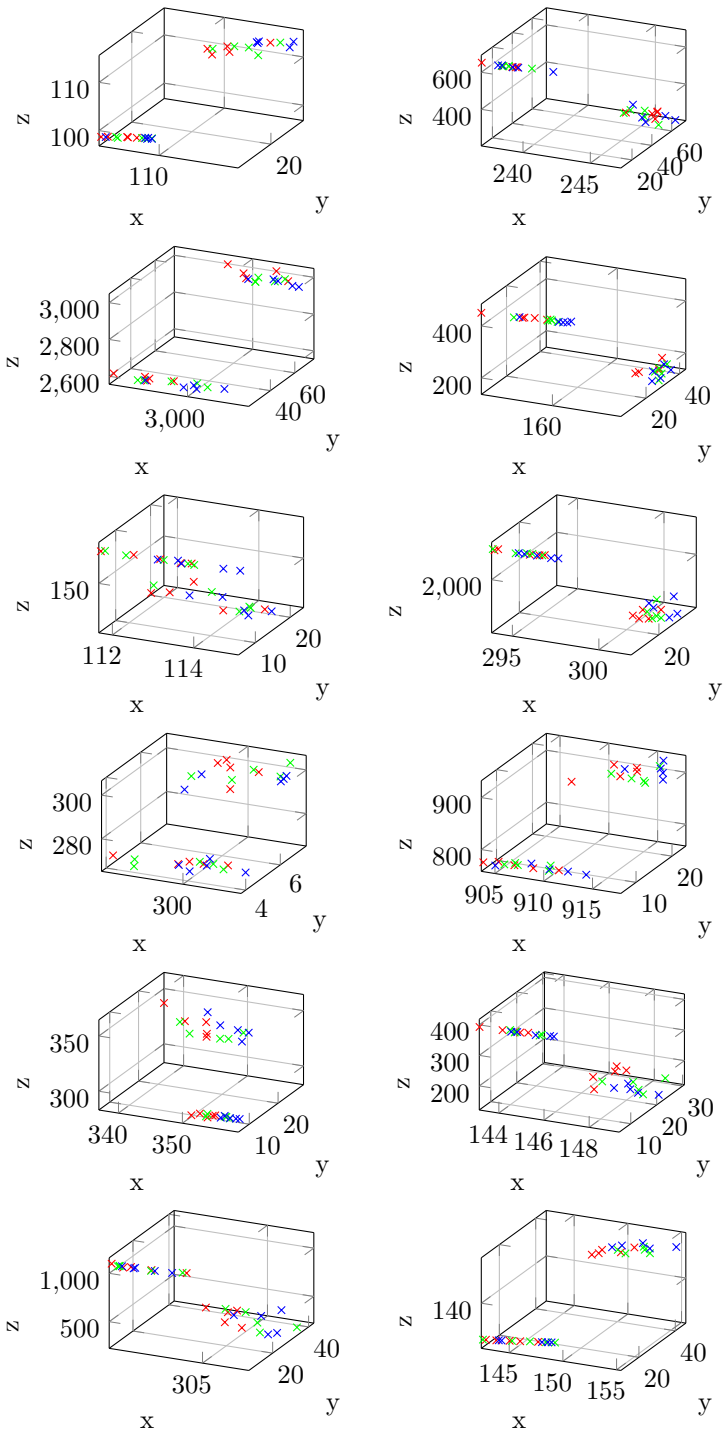

Figure B.4: AMPSO Sequence Comparison by Swarm Size

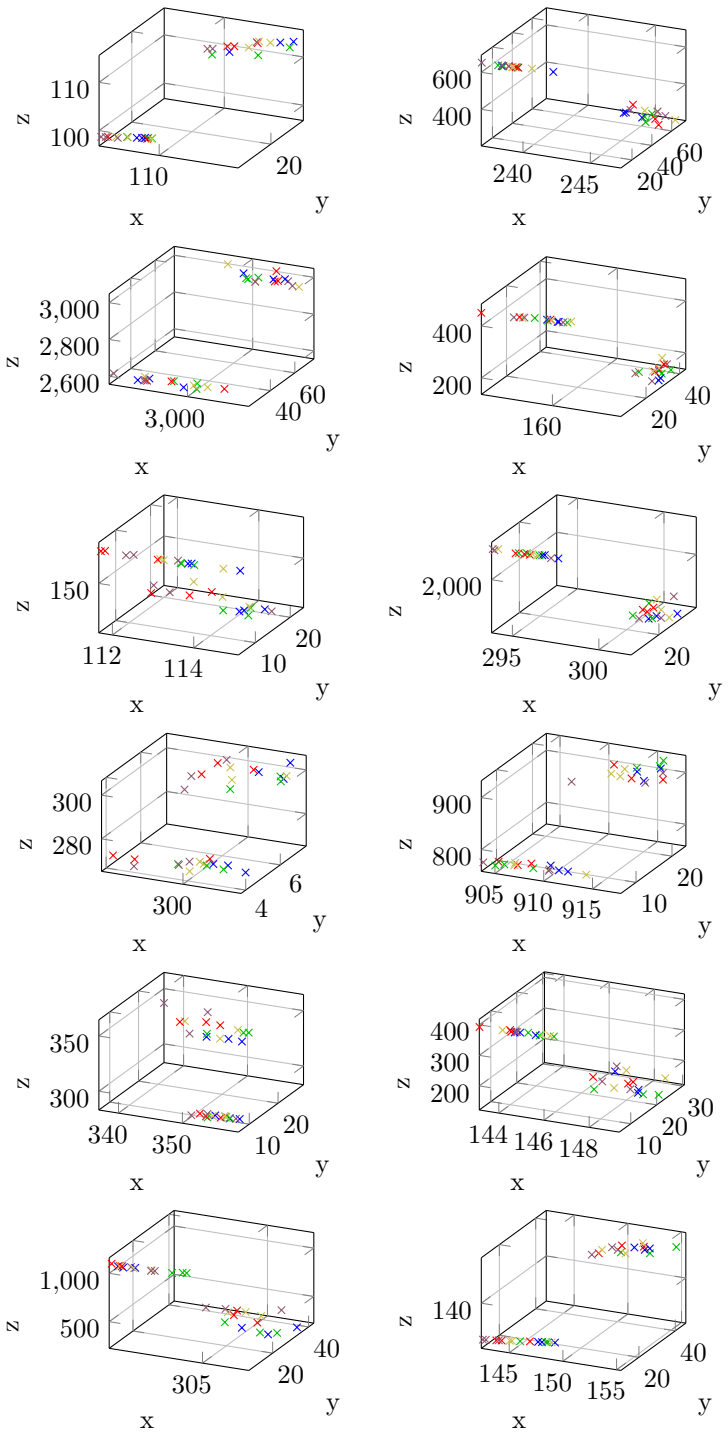

Figure B.5: AMPSO Sequence Comparison by Coefficient Configuration

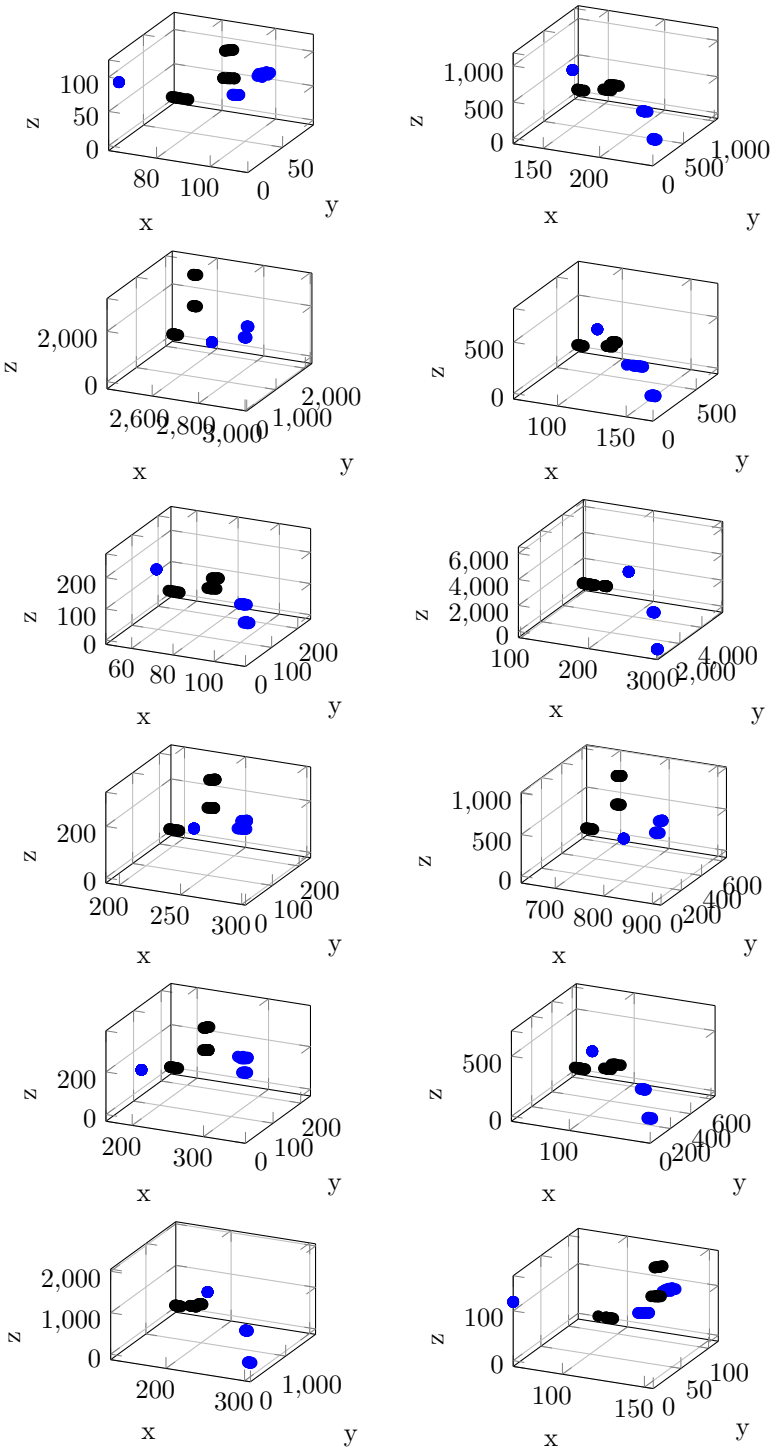

Figure B.6: BPSO and AMPSO Sequence Comparison

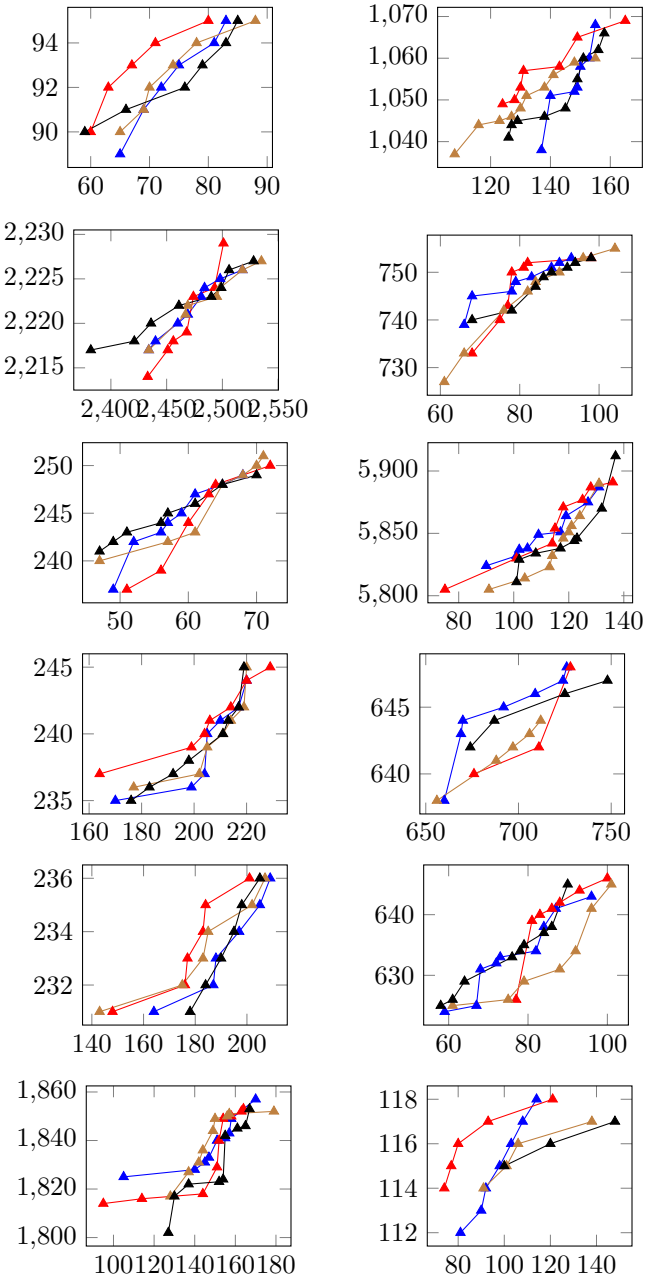

Figure B.7: B-MGPSO Best Archives by Swarm Size (x: Aligned Characters; y: Inserted Indels)

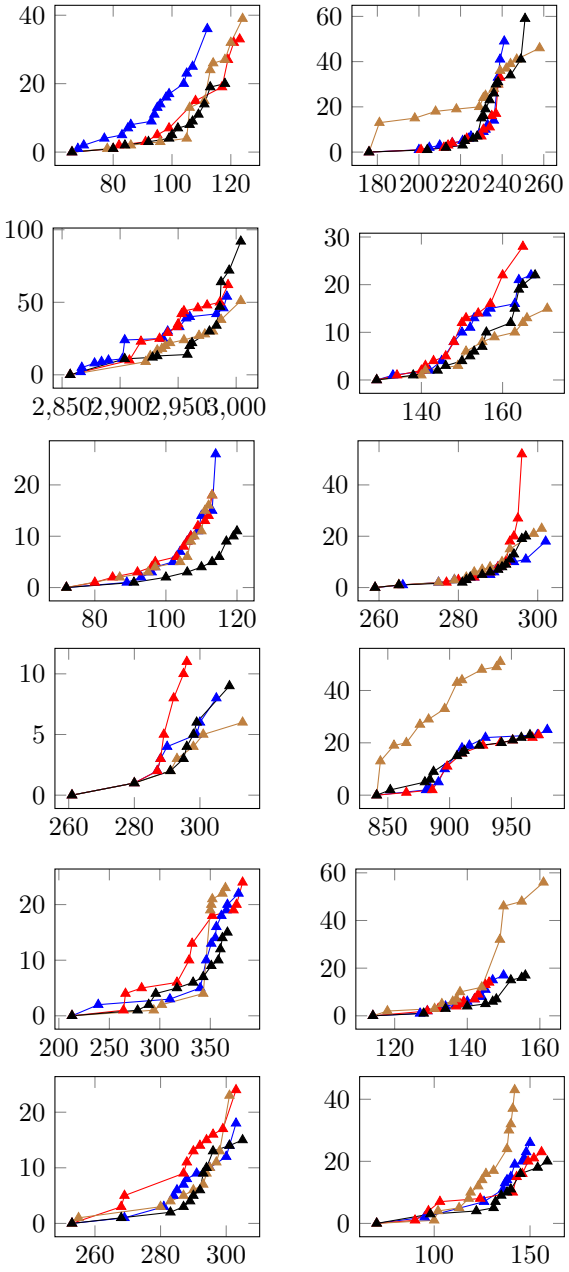

Figure B.8: AM-MGPSO Best Archives by Swarm Size (x: Aligned Characters; y: Inserted Indels)

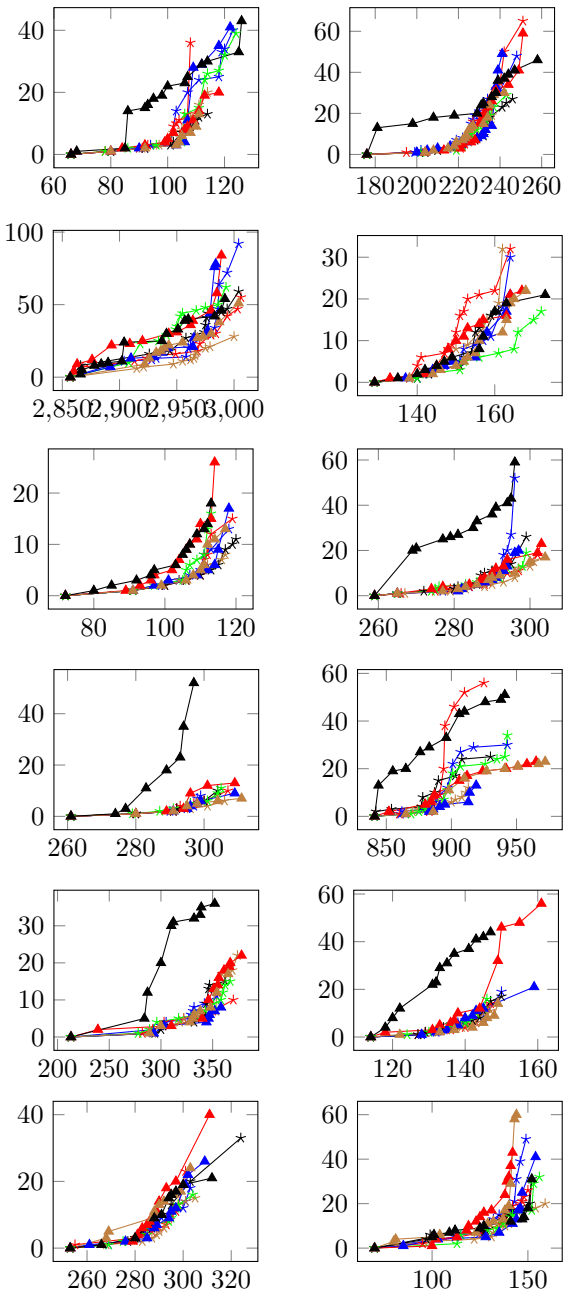

Figure B.9: AM-MGPSO Best Archives by Coefficient (x: Aligned Characters; y: Inserted Indels)

Appendix C

Acronyms

PSO Particle Swarm Optimization

MSA Multiple Sequence Alignment

BPSO Binary PSO

AMPSO Angular Modulated PSO

MGPSO Multi-guided PSO

MOP Multi-objective Problem

MOOP Multi-objective Optimization Problem

B-MGPSO Binary MGPSO

AM-MGPSO Angular Modulated MGPSO

Bibliography

[1] David B Fogel. "What is evolutionary computation?" In: *IEEE spectrum* 37.2 (2000), pp. 26–32.

[2] Lusheng Wang and Tao Jiang. "On the complexity of multiple sequence alignment". In: *Journal of computational biology* 1.4 (1994), pp. 337–348.

[3] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[4] Julie D Thompson, Desmond G Higgins, and Toby J Gibson. "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice". In: *Nucleic acids research* 22.22 (1994), pp. 4673–4680.

[5] Amarendran R Subramanian, Michael Kaufmann, and Burkhard Morgenstern. "DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment". In: *Algorithms for Molecular Biology* 3.1 (2008), pp. 1–11.

[6] Timo Lassmann and Erik LL Sonnhammer. "Kalign–an accurate and fast multiple sequence alignment algorithm". In: *BMC bioinformatics* 6.1 (2005), pp. 1–9.

[7] Kazutaka Katoh and Daron M Standley. "MAFFT multiple sequence alignment software version 7: improvements in performance and usability". In: *Molecular biology and evolution* 30.4 (2013), pp. 772–780.

[8] Robert C Edgar. "MUSCLE: a multiple sequence alignment method with reduced time and space complexity". In: *BMC bioinformatics* 5.1 (2004), pp. 1–19.

[9] Cédric Notredame, Desmond G Higgins, and Jaap Heringa. "T-Coffee: A novel method for fast and accurate multiple sequence alignment". In: *Journal of molecular biology* 302.1 (2000), pp. 205–217.

[10] Chuong B Do et al. "ProbCons: Probabilistic consistency-based multiple sequence alignment". In: *Genome research* 15.2 (2005), pp. 330–340.

[11] Julie D Thompson et al. "A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives". In: *PloS one* 6.3 (2011), e18093.

[12] Julie D Thompson, Frédéric Plewniak, and Olivier Poch. "A comprehensive comparison of multiple sequence alignment programs". In: *Nucleic acids research* 27.13 (1999), pp. 2682–2690.

[13] Cédric Notredame. "Recent progress in multiple sequence alignment: a survey". In: *Pharmacogenomics* 3.1 (2002), pp. 131–144.

[14] Soniya Lalwani, Rajesh Kumar, and Nilama Gupta. "A review on particle swarm optimization variants and their applications to multiple sequence alignment". In: *Journal of Applied Mathematics and Bioinformatics* 3.2 (2013), p. 87.

[15] Gary Pampara, Nelis Franken, and Andries Petrus Engelbrecht. "Combining particle swarm optimisation with angle modulation to solve binary problems". In: *2005 IEEE congress on evolutionary computation*. Vol. 1. IEEE. 2005, pp. 89–96.

[16] Christiaan Scheepers et al. "Multi-guided particle swarm optimization: A multi-objective particle swarm optimizer". PhD thesis. University of Pretoria, 2017.

[17] James Kennedy and Russell C Eberhart. "A discrete binary version of the particle swarm algorithm". In: *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*. Vol. 5. IEEE. 1997, pp. 4104–4108.

[18] Hai-Xia Long et al. "Multiple sequence alignment based on a binary particle swarm optimization algorithm". In: *2009 Fifth International Conference on Natural Computation*. Vol. 3. IEEE. 2009, pp. 265–269.

[19] Amrita Chakraborty and Arpan Kumar Kar. "Swarm intelligence: A review of algorithms". In: *Nature-Inspired Computing and Optimization* (2017), pp. 475–494.

[20] Riccardo Poli. *An analysis of publications on particle swarm optimization applications*. Tech. rep. Technical Report CSM-469, Department of Computer Science, University of Essex, 2007.

[21] Dongshu Wang, Dapei Tan, and Lei Liu. "Particle swarm optimization algorithm: an overview". In: *Soft Computing* 22.2 (2018), pp. 387–408.

[22] Shane Strasser et al. “A new discrete particle swarm optimization algorithm”. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. 2016, pp. 53–60.

[23] Mehmet Kaya, Abdullah Sarhan, and Reda Alhajj. “Multiple sequence alignment with affine gap by using multi-objective genetic algorithm”. In: *Computer methods and programs in biomedicine* 114.1 (2014), pp. 38–49.

[24] R Ranjani Rani and D Ramyachitra. “Multiple sequence alignment using multi-objective based bacterial foraging optimization algorithm”. In: *Biosystems* 150 (2016), pp. 177–189.

[25] Huazheng Zhu, Zhongshi He, and Yuanyuan Jia. “A novel approach to multiple sequence alignment using multiobjective evolutionary algorithm based on decomposition”. In: *IEEE journal of biomedical and health informatics* 20.2 (2015), pp. 717–727.

[26] Ekunda Lukata Ulungu and Jacques Teghem. “Multi-objective combinatorial optimization problems: A survey”. In: *Journal of Multi-Criteria Decision Analysis* 3.2 (1994), pp. 83–104.

[27] Patrick Ngatchou, Anahita 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.

[28] Rui Mendes, James Kennedy, and José Neves. “The fully informed particle swarm: simpler, maybe better”. In: *IEEE transactions on evolutionary computation* 8.3 (2004), pp. 204–210.

[29] Andries Petrus Engelbrecht. “Particle swarm optimization: Global best or local best?” In: *2013 BRICS congress on computational intelligence and 11th Brazilian congress on computational intelligence*. IEEE. 2013, pp. 124–135.

[30] Frans Van den Bergh and Andries Petrus Engelbrecht. “A study of particle swarm optimization particle trajectories”. In: *Information sciences* 176.8 (2006), pp. 937–971.

[31] Da-Fei Feng and Russell F Doolittle. “Progressive sequence alignment as a prerequisitetto correct phylogenetic trees”. In: *Journal of molecular evolution* 25.4 (1987), pp. 351–360.

[32] Kumar Chellapilla and Gary B Fogel. “Multiple sequence alignment using evolutionary programming”. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 1. IEEE. 1999, pp. 445–452.

[33] Mohammad Reza Bonyadi and Zbigniew Michalewicz. "Stability analysis of the particle swarm optimization without stagnation assumption". In: *IEEE Transactions on Evolutionary Computation* 20.5 (2015), pp. 814–819.

[34] Christiaan Scheepers, Andries P Engelbrecht, and Christopher W Cleghorn. "Multi-guide particle swarm optimization for multi-objective optimization: empirical and stability analysis". In: *Swarm Intelligence* 13.3 (2019), pp. 245–276.

[35] Kyle Erwin and Andries Engelbrecht. "Control parameter sensitivity analysis of the multi-guide particle swarm optimization algorithm". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2019, pp. 22–29.

[36] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. "Performance metrics in multi-objective optimization". In: *2015 Latin American Computing Conference (CLEI)*. IEEE. 2015, pp. 1–11.

[37] Julie D. Thompson, Frédéric Plewniak, and Olivier Poch. "BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs." In: *Bioinformatics (Oxford, England)* 15.1 (1999), pp. 87–88.