Multiple Sequence Alignment using Multi-Objective Particle Swarm Optimization

Quinton Weenink
dept. Computer Science
University of Pretoria
Pretoria, South Africa
u13176545@tuks.co.za

I. Introduction

This paper aims to investigate how to achieve multiple sequence alignment using a multi-objective *particle swarm optimisation algorithm* (PSO). While many multi-objective optimisation techniques exist this paper will investigate the performance of *vector evaluated particle swarm optimization* (VEPSO) to optimise the required objectives.

II. Background

Sequence alignment is a way of arranging the sequences to identify regions of similarity. For this research we will be attempting to align the sequences by identifying multiple objectives and attempting to solve them, namely:

- maximise the number of vertically aligned characters, while simultaneously
- minimising the number of leading spaces inserted.

An example of such a sequence is as follows:

a b c d e f
b b d h g
c a b f

A possible solution to the above sequence is:

– a b – c d e f – –
– – b b – d – – h g
c a b – – – – f – –

A. Fitness

The following describes how the fitness will be calculated for each of the objectives

1) Alignment fitness: The alignment fitness is calculated by incriminating the fitness each time the same character is aligned. Furthermore the fitness is decreased when different characters are aligned. The result is then negated in order to make both objectives minimisation problems. A solution who's characters are misaligned is considered an in-valid solution for this research.

2) Leading space fitness: The leading space fitness is calculated by adding the number of spaces that the solution requires.

The objective fitness for the above solution would be:

- Alignment fitness $= -(1 + 2 + 1 + 1) = -5$
- Leading space fitness $= 15$

B. Position

Each particle will have a position that corresponds to each of the characters. For the above sequence that means a position vector of size 15 where $x_j \in [0, 7]$. Positions are floored in order to get the number of spaces to be inserted before the characters.

III. Original Particle Swarm Optimization

A PSO is a machine learning algorithm where each particle moves around the search space attempting to find the optimal solution with the influence of its neighbouring particles. The position for each particle iteration is calculated as follows.

$$\vec{x}_i^t = \vec{x}_i^{t-1} + \vec{v}_i^t \tag{1}$$

Where:

- $\vec{x}$ is the particle position vector in n-dimensions
- $\vec{v}$ is the particle velocity vector, used to calculate the particles next position
- t is the time increment
- i is the particle number

For each iteration of the algorithm, a new location of the particle is calculated based on its previous location and velocity vector as seen in (1). The following describes a the velocity update algorithm.

$$\vec{v}_i^t = w.\vec{v}_i^{t-1} + c_1.\vec{r}_1^t.(\vec{x}_{pBest,i} - \vec{x}_i^{t-1}) + c_2.\vec{r}_2^t.(\vec{x}_{nBest,i} - \vec{x}_i^{t-1}) \tag{2}$$

Where:

- c_1 is the acceleration coefficient for the cognitive component
- c_2 is the acceleration coefficient for the social component
- r_1 and r_2 is a vector of random numbers in the range (0, 1)
- $\vec{x}_{pBest}$ is the personal best position of that particle
- $\vec{x}_{nBest}$ is the best position found in that particles neighbourhood

Particle's social component, the third term in (2), requires $nBest$, the best neighbouring particles position. The set of

neighbouring particles is determined by the topology used in the PSO. Different topologies could influence the performance of the PSO.

A. Social Network Topologies

In the above mentioned PSO the neighbourhood could be described in a variety of ways. One such method is the $gBest$ topology, where each particle is a neighbour of every other particle in the network. This means that the particle's $nBest$ is always the global best particle for the swarm.

Another social topology called $lBest$ connects particles in a ring topology, the neighbourhood is specified by a fixed size of n_s. The $nBest$ in this case is determined by the best particle within n_s neighbours from the current particle [1].

IV. Vector Evaluated Particle Swarm Optimization

In this multi-swarm concept, each objective function is optimised by a swarm of particles using the $gBest$ from another swarm [2].

While position for each particle is updated in the same way as an original PSO the velocity update function changes. Assuming a multi-objective problem with 2 objectives the following describes the velocity update for each of the swarms.

$$S_1.\vec{v}_i^t = w.S_1.\vec{v}_i^{t-1} + c_1.\vec{r}_1^t.(S_1.\vec{x}_{pBest,i} - S_1.\vec{x}_i^{t-1}) + c_2.\vec{r}_2^t.(S_2.\vec{x}_{nBest,i} - S_1.\vec{x}_i^{t-1}) \quad (3)$$

$$S_2.\vec{v}_i^t = w.S_2.\vec{v}_i^{t-1} + c_1.\vec{r}_1^t.(S_2.\vec{x}_{pBest,i} - S_2.\vec{x}_i^{t-1}) + c_2.\vec{r}_2^t.(S_1.\vec{x}_{nBest,i} - S_2.\vec{x}_i^{t-1}) \quad (4)$$

Where:

- S_1 and S_2 are the values associated with swarm 1 and swarm 2 respectively
- There rest remain the same as in the original pso velocity update function (2)

A. Swarm Social Network Topologies

Similarly to the social network topologies mentioned in for the original PSO the same topologies apply to VEPSO. In the same way the $lBest$ functions for particle social topologies the swarm discussed will share information to the next swarm in the same way.

$pBest$ and $nBest$ positions were only used as a local or global guide if they were non-dominated by a previous solution as well as a valid solution.

B. Objectives

Each of the swarm's error functions is the resulting fitness of one of the objectives. A good multi-objective optimisation algorithm should be able to solve its own objective while not forgetting about the other swarms objectives.

C. Boundary Constraint Handling Mechanisms

The the boundary constraint clamping approach used in this research to prevent particles from leaving the search space. If a particle violates a boundary constraint in a specific dimension, then the particles position was clamped to the edge of the constraint. For this research this was specified by the max length of the sequences.

V. Experiments

A. PSO configuration

In order to achieve viable results, each experiment was run as a mean over 30 samples. r_1 and r_2 sampled from a uniform distribution $(0, 1)$ as specified in equation 2.

A memory based PSO was used with a star or $gBest$ social topologie.

50 particles were used for all functions.

The constants $c_1 = c_2 = 1.4$ as well $w = 0.7$ were set as suggested in [1].

No V_{max} was used during this study.

VI. Results

TABLE I
Results after 2000 iterations. Means are reported over 30 samples with standard deviations in parenthesis

		f_1	f_2	f_3
S_1	$f1$	-4.2666666 (0.7272474)	-5.6333333 (1.0159833)	-3.93333333 (0.2494438)
	$f2$	40.6 (11.842297)	86.5 (14.6986393)	45.7 (10.030453)
S_2	$f1$	-0.866666 (4.74505590)	-2.6333333 (3.69218393)	-2.1 (3.02599845)
	$f2$	27.2 (5.3628350)	73.3 (7.6164296)	30.1 (4.7423622)

Figures (1, 2, 3) indicated the objective's fitness over the 2000 iterations that these solutions were trained over. It can be observed that each swarm is attempting to achieve result best for their objective. Each swarm being the best fitness for its particular objective.

One can also observe the objectives of each swarm approaching each other as their results are influenced by the global guides that are effecting their velocity.

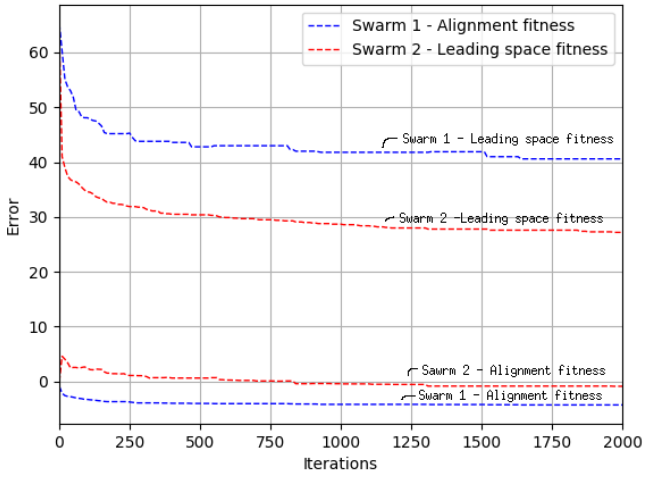

Fig. 1. f_1's fitness objectives over time

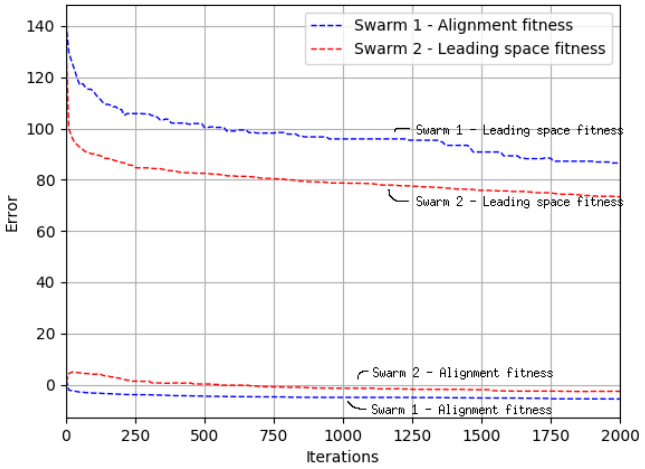

Fig. 2. f_2's fitness objectives over time

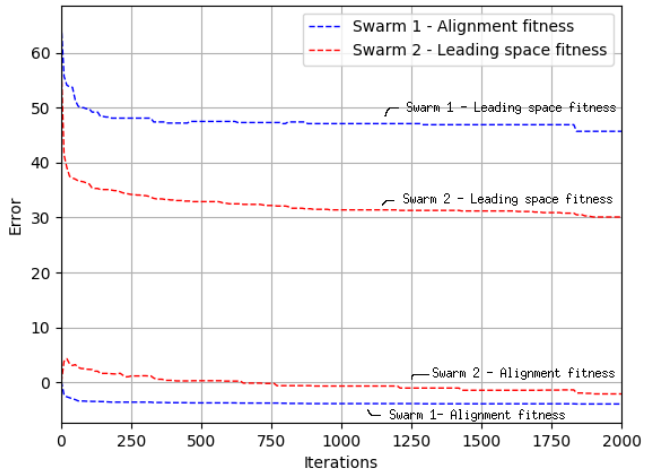

Fig. 3. f_3's fitness objectives over time

Figures (4, 5, 6) plot the solutions according to their swarm (x-axis: Alignment fitness, y-axis: Leading space fitness). For each of the testing sequences one can see that the swarms form what appears to be a Pareto front [2]. This being the place where the one objective can't improve more without negatively impacting the other.

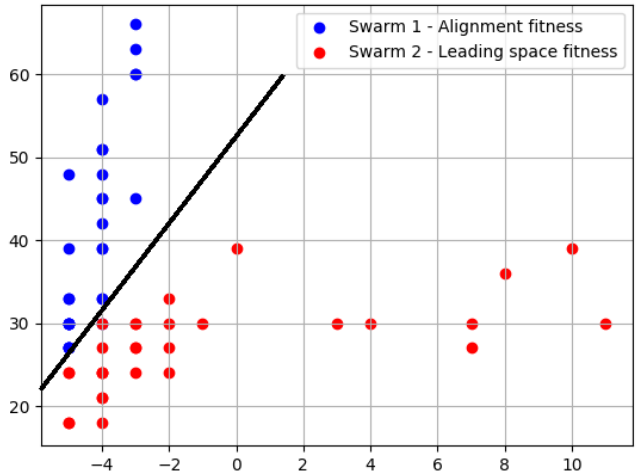

Fig. 4. f_1's non-dominated solutions

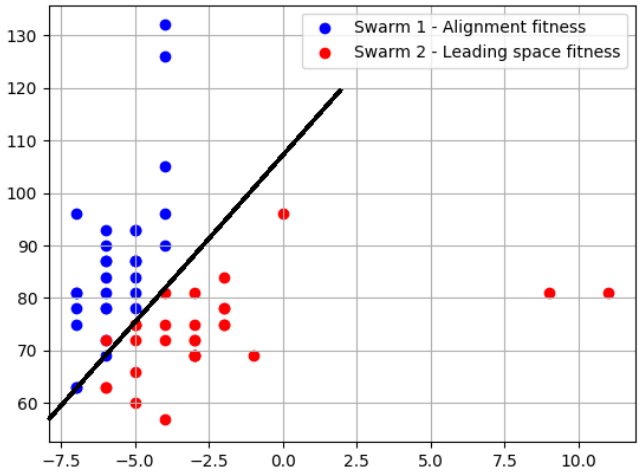

Fig. 5. f_2's non-dominated solutions

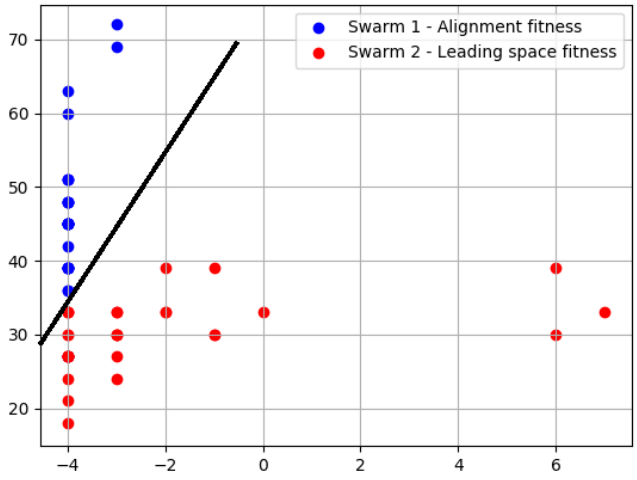

Fig. 6. f_3's non-dominated solutions

VII. Conclusion

In conclusion one can observe the ability of VEPSO's in solving multiple sequence alignment problems. VEPSO did however struggle with lager sequences like in f_2 and one can assume this is due to lack of exploration. The addition of achieve, as discussed in [2], would significantly benefit the optimisation as well as allow the addition of multiple solutions resulting in in increased exploration.

References

[1] A. B. van Wyk and A. P. Engelbrecht, "Overfitting by pso trained feedforward neural networks," in *IEEE Congress on Evolutionary Computation*, IEEE. IEEE, Jul. 2010. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/5586333/?section=abstract

[2] K. S. Lim and Z. Ibrahim, "Improving vector evaluated particle swarm optimisation by incorporating nondominated solutions," *ScientificWorldJournal*, vol. 5, no. 500, p. 3943, 1995. [Online]. Available: https://link.springer.com/article/10.1007/s00500-014-1222-z

[3] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Micro Machine and Human Science*, vol. 2013, no. 500, p. 3943, 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3662110/

[5] A. Rakitianskaia and A. Engelbrecht, "Measuring saturation in neural networks," in *Symposium Series on Computational Intelligence*, IEEE. IEEE, Dec. 2015. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7376778/

VIII. Appendix

f_1, sequence provided, where $x_j \in [0, 7]$

a b c d e f
b b d h g
c a b f

f_2, longer sequence, where $x_j \in [0, 10]$

a b c d e f a a h
f b b d h g h h
c a b f f e a

f_3, a sequence that would benefit from bounds bigger than the number of sequences, where $x_j \in [0, 7]$

a b c d e f
g h i j k
a b c d