An ant colony algorithm for multiple sequence alignment in bioinformatics

Jonathan Moss and Colin G. Johnson *

*Computing Laboratory University of Kent at Canterbury Canterbury, Kent, CT2 7NF, England. C.G.Johnson@ukc.ac.uk

Abstract

This paper describes a the application of *ant colony optimization* algorithms, which draw inspiration from the way ants organize themselves in searching for food, to the well-known bioinformatics problem of aligning several protein sequences.

1 Introduction

Swarm intelligence methods are computational techniques inspired by animals such as social insects acting together to solve complex problems. The main application of these techniques has been to combinatorial optimization problems. This paper discusses work-in-progress on the application of swarm intelligence ideas to a bioinformatics problem, *viz.* aligning multiple protein sequences which are believed to be related.

The paper begins with a brief survey of swarm intelligence and the multiple sequence alignment problem. The application of one to the other is then described, and some preliminary results are given both on synthetic problems and on real-world data.

2 Ant colony optimization and swarm intelligence

Ant colonies are able to organize their foraging behaviour in a seemingly efficient way without any centralized control [7]. This self-organizing structure is carried out via *stigmergic* communication, i.e. communication by changing the environment, in this case by laying down pheromone trails.

Initially ants have no idea of where food is in the environment, so they wander randomly, leaving a pheromone trail. When an ant finds food it wanders back to the nest. Initially these paths will be arbitrary, but when an ant follows a shorter path it will be able to follow that path more often within the same time period than an ant following a longer path, so there is a *positive reinforcement* process whereby the shorter paths get stronger.

A simple version of this is illustrated in figure 1, where ants have two possible routes from a nest to a food source. If two ants set out at the same time, one taking route A and one route B, which is twice as long, then the ant taking A will have travelled back and forth between the food source twice in the same time that the other ant has travelled back and forth once. Therefore there will be a stronger pheromone trail on route A compared to route B. This idea can be effectively scaled up to solving route finding problems such as the TSP, with performance as good as or better than existing heuristics [2, 3].

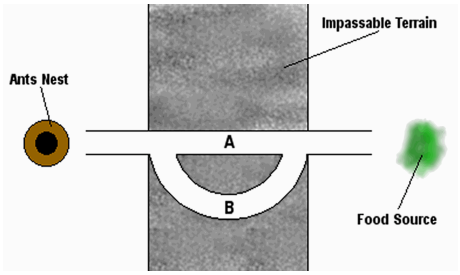

Fig. 1. A simple ant foraging problem.

3 Multiple sequence alignment

Proteins are complex molecules which consist of a long chain of *amino acids*. The sequence of these amino acids along the chain is specified by transscribing and translating the DNA sequence in the cell. These chains fold up into a complex three dimensional structure. Proteins are the basic building blocks of living organisms; most of the body is built from proteins of various kinds, and proteins are used to carry signals around the body and carry out the various actions which an organism needs to do to survive.

The fact that proteins consist of long linear sequences of simple subcomponents means that we can store this information easily on the computer. Finding relationships between such sequences is an important part of the subject known as *bioinformatics* [1, 5, 8].

Over the course of evolutionary history proteins

become modified as organisms evolve. Nonetheless enough commonality remains so that proteins with a common evolutionary history can be identified. There are a number of reasons for being interested in this, e.g.:

1. This information can be used to support the reconstruction of phylogenetic trees by giving an indication of how much time has passed since present organisms branched off from a common ancestor.

2. If several proteins have commonalities at the sequence level, this may correlate with commonalities in their three-dimensional structure, so this may contribute to the ongoing work on predicting three-dimensional structures of proteins.

3. Certain "families" of proteins are commonly found together. If two organisms have a strong alignment between certain proteins which belong to one of these families, then it is likely that the other proteins in the family will also be present [1].

There are a number of ways in which protein sequences can change. Firstly one amino acid can be substituted for another. In particular amino acids that have similar properties are more likely to be substituted, so alignment methods tend to incorporate measures of substitutability based either on data about known substitutions or biochemical properties. Secondly amino acids can be inserted or deleted from the sequence. Therefore one of the requirements for an alignment algorithm is to be able to include gaps in the sequence to enable a sequence which has lost or gained amino acids to be lined up against another sequence which hasn't. A number of methods have been applied in multiple sequence alignment such as hidden Markov models and dynamic programming (a good survey is [5]).

4 Applying ant colony optimization to multiple sequence alignment

We have developed a system, called *AntAlign* (summarized in figure 2), which applies the ant colony optimization techniques to the multiple sequence alignment problem.

The main idea of the system that ants take a subsequence and move in an interval associated with each sequence, strengthening a pheromone trail when a close match is found to a sequence at that position in other sequences. As the algorithm runs larger fragments of sequence are picked up by the ants. This is designed to encourage the removal of extraneous gaps later on in the process. The overall architecture of the program is indicated in figure 2.

Cycles	The number of cycles executed.
PopPerTime	The number of `Ant`s generated per cycle.
StartLen	The length of the `Ant`'s subsequence in the first cycle.
EndLen	The length of the `Ant`'s subsequence in the final cycle.
Evap	The rate of trail evaporation.
Intensity	The intensity of the trail.
Drift	The distance `Ant`s are allowed to drift from the current consensus and still score a match.
RndChance	The chance that an `Ant` will choose a random path rather than a matched trail.

Table I. Parameters for the algorithm.

The first type of object used to build the system is a "trail manager" (called an `ITrail`) which will organize the pheromone sequences. There is one `ITrail` for each sequence, and an additional `ITrail` in the system which will manage the emerging consensus sequence. The consensus sequence contains the strongest match so far at each position.

Objects of this `ITrail` class play a number of roles. The main role is in storing the pheromone trails which are created as the ants move along the sequences looking for a good match. Associated with each `ITrail` is an interval in which the pheromone trails will be placed, i.e. a number of locations (longer than the sequence itself) into which pheromone can be placed. The strength of these pheromone trails will eventually determine the sequence.

The `ITrail` object also manages the three main functions related to the pheromone trails (`PTrails`). Firstly when a new `Ant` is created the `ITrails` object compares the subsequence the `Ant` is carrying to those associated with trails already stored in the `ITrails` object to determine the possible paths the `Ant` could take. The second function it provides is the adding of `PTrails` into its interval, or enhancing existing `PTrails` if a matching `PTrail` is already present. The final role the `ITrail` plays is applying evaporation to the trail strengths at the end of each cycle.

The main driving program is contained in the `AntSystem` class. To start a run of the program a number of parameters are specified (table I). The

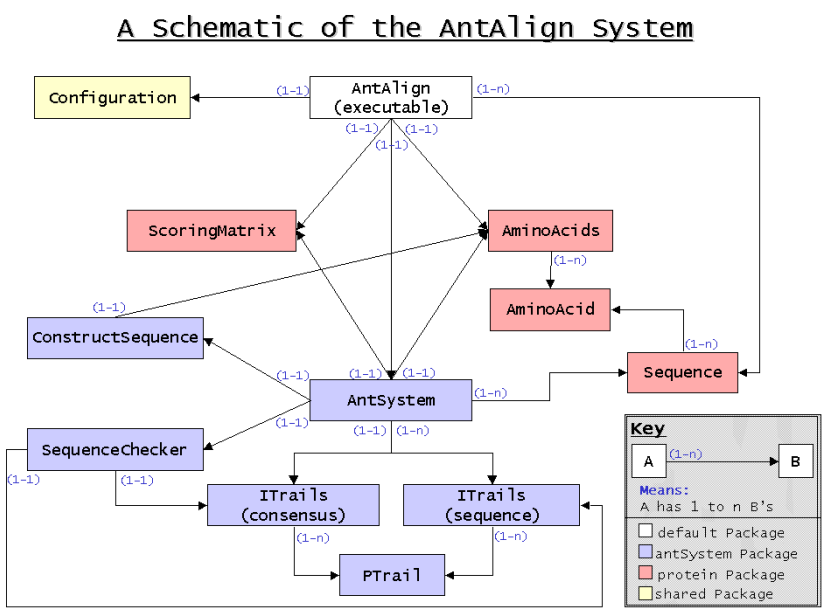

Fig. 2. An overview of the components of the *AntAlign* system and their interactions.

following pseudocode indicates the main algorithm used:

```
LOOP until number == cyclesRequired
  cargoLength := startingLength
        + (cycleNumber*rateOfLengthIncrease)
  LOOP though each sequence
    Generate number of ants required
          from current sequence
    LOOP through Ants
      Get matches of current Ant to
          consensus PTrails
      Determine match probabilities
      Determine path taken
      Generate alignmentScore for
          the path taken
      IF (alignmentScore>0.0)
        Put new PTrail into consensus trail
        Put new PTrail into current
            sequence trail
      ENDIF
    ENDLOOP
  ENDLOOP
ENDLOOP
```

The first few lines of the algorithm simply sets up loops through cycles, sequences and ants. During each iteration around the inner loop each `Ant` takes a random subsequence from its sequence and attempts to place it within the interval. Details of the core steps of the algorithm are as follows:

Get matches of `Ants` to consensus `Ptrails`. An important decision in creating the algorithm was that an `Ant` would have read access only to the pheromone trails related to the consensus sequence, not to the other sequences directly (it would be interesting future work to compare this with a variation where ants associated with a sequence have access to all other sequences). A match is determined to have happened if the subsequence carried by the `Ant` and the offset of the ant from the start of the `ITrail` interval is close to the offset of the `PTrail`. *Close* here means within the distance specified by the **Drift** parameter to the left or right.

Determine match probabilities. This determines the percentage probability of following each of the paths in the set $P = \{p_1, \ldots, p_n\}$ determined by the previous step in the algorithm, or another path chosen at random from all possible paths. The probability of a path p_i being chosen is

$$\frac{t_i}{\sum_{i=1}^{n} t_i} \times (100 - \textbf{RndChance})$$

where t_i is the strength of trail p_i and the **RndChance** is the parameter giving the percentage chance of a random path being followed instead of the path determined in the previous step of the algorithm.

Determine path taken. A path is chosen according to the probabilities calculated in the previous step of the algorithm.

Generate alignment score for path taken. A score is calculated based on the match between the current consensus sequence and the path taken by the ant. This is based on the well-known BLOSUM-62 matrix [6] which was calculated from data about probabilities of protein substitution from a large protein sequence database.

Because the subsequences which each `Ant` carries are generated at random, there is a chance that parts of the sequence are missed out or misaligned. To combat this a repair algorithm (the *SequenceChecker*) is invoked every 10 cycles, which tidies up inconsistencies. However as can be seen in the results below this is not ideal. Current work is attempting to improve the repair algorithm, and a new version of the algorithm is being implemented which divides the sequence completely between ants which are not allowed to move out of sequence order.

5 Results

Firstly we show the alignment of six copies of the same sequence (figure 3). With a small exception, these are lined up perfectly by the algorithm. The one exception is at the end of the third sequence, where an amino acid has been missed out. This demonstrates the need for improvements to the repair algorithm.

Secondly we take six randomly generated sequences (figure 4). It is unsurprising that these sequences are all bunched close to each other; there is minimal commonality between them which could form the basis for spreading out the sequences.

The third experiment worked with real data. This consisted of six variations on the chaperonin protein Cpn60 taken from six different bacterial species. The results from this are illustrated in figure 5. As would be expected from a highly conserved sequence there is little spread, the representation is rather compact. However several small improvements were not discovered by the algorithm, e.g. sequence number 1 lags behind a much stronger position by one position for much of the sequence.

The final experiment uses a less highly conserved set of sequences, consisting of six serine proteases from various species. These were chosen because they have small regions which are active in the function of the protein and much of the remainder has drifted over evolutionary timescales. These results are shown in figure 6. A particular point of interest is towards the end of the sequences, where a large gap has been left in the first sequence to make it line up strongly against the 4th and 5th sequences.

There is some evidence that the algorithm scales well as the number of sequences being aligned is increased. Results showing the time taken to align various numbers of sequences are given in figure 7. Adding a new sequence increases the time taken by approximately a factor of 1.5.

6 Conclusions and future work

This paper has presented the current state of work on this project. One important improvement which needs to be made is to improve the repair algorithm or replace the random assignment of subsequence payloads to `Ant`s with a more consistent method. Another important area of work is carrying out a more formal comparison with other algorithms, including both heuristics developed specifically for multiple sequence alignment and metaheuristics adapted for this problem.

References

[1] T.K. Attwood and D.J. Parry-Smith. *Introduction to Bioinformatics*. Addison Wesley Longman, 1999.

[2] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence*. Oxford University Press, 1999.

[3] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics—Part B*, 26(1):29–41, 1996.

[4] Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors. *Ant Algorithms: Third International Workshop, ANTS 2002*. Springer, 2002.

[5] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[6] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA*, 89:10915–10919, 1992.

[7] B. Hölldobler and E.O. Wilson. *The Ants*. Springer, 1990.

[8] Arthur M. Lesk. *Introduction to Bioinformatics*. Oxford University Press, 2002.

Fig. 3 **Fig. 3.** Results from the alignment of identical sequences.

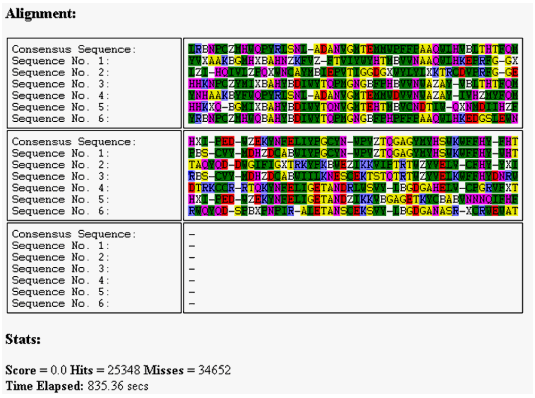

Fig. 4 **Fig. 4.** Results from the alignment of random sequences.

Fig. 5 **Fig. 5.** Results from the alignment of highly conserved sequences.

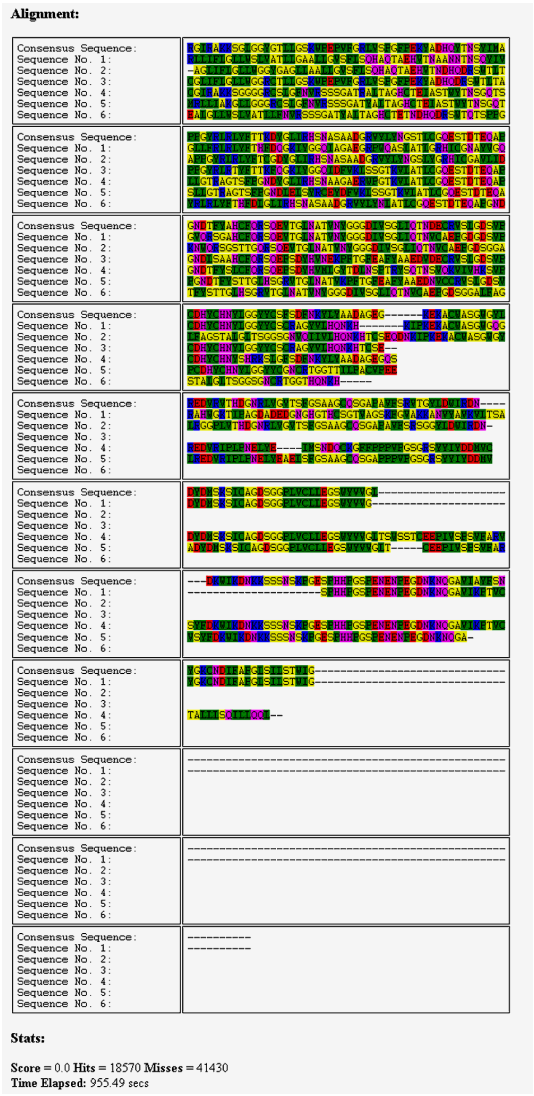

Fig. 6 **Fig. 6.** Results from the alignment of less highly conserved sequences.

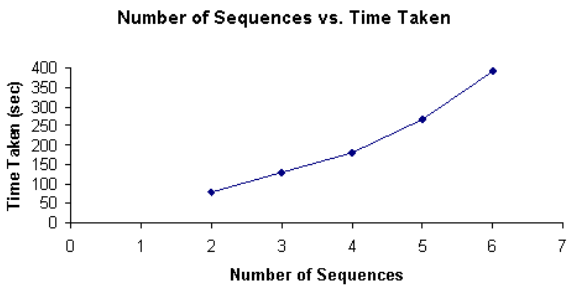

Fig. 7. Time taken vs. number of sequences.