# Sequence Alignment Using Nature-Inspired Metaheuristic Algorithms

Muhammad Luthfi Shahab and Mohammad Isa Irawan

***Abstract*—The most basic process in sequence analysis is sequence alignment, usually solved by dynamic programming Needleman-Wunsch algorithm. However, Needleman-Wunsch algorithm has some lack when the length of the sequence which is aligned is big enough. Because of that, sequence alignment is solved by metaheuristic algorithms. In the present, there are a lot of new metaheuristic algorithms based on natural behavior of some species, we usually call them as nature-inspired metaheuristic algorithms. Some of those algorithm that are more efficient are firefly algorithm, cuckoo search, and flower pollination algorithm. In this research, we use those algorithms to solve sequence alignment. The results show that those algorithms can be used to solve sequence alignment with good result and linear time computation.**

***Index Terms*—Nature-inspired metaheuristic algorithms, sequence alignment.**

## I. Introduction

NEW DNA, RNA and protein sequences usually develop from pre-existing sequences rather than get invented by nature from scratch. This fact is the foundation of any sequence analysis [1], [2]. And the most basic process in sequence analysis is sequence alignment, and usually solved by dynamic programming Needleman-Wunsch algorithm. However, Needleman-Wunsch algorithm needs a lot of memories when the length of the sequence which is aligned is big enough.

Calculation of the dynamic programming matrices for a pair of sequences takes a substantial amount of time and memory. However, as we have seen, the areas of interest in these matrices (that is, the areas formed by the cells that take part in a traceback procedure) are minuscule in comparison to the entire areas of the matrices. The FASTA algorithm is designed to limit the areas of the matrices that the dynamic programming examines. But FASTA can miss optimal alignments [1].

In the present, there are a lot of new metaheuristic algorithms based on natural behavior of some species, we usually call them as nature-inspired metaheuristic algorithms. Those algorithms can be used to solve many problems. Earlier algorithms in nature-inspired metaheuristic algorithms are genetic algorithm, ant algorithm, particle swarm optimization, and another.

Nature-inspired metaheuristic algorithms, especially those based on swarm intelligence, have attracted much attention in the last ten years. Firefly algorithm, developed in 2008, cuckoo search, developed in 2009, and flower pollination algorithm, developed in 2012 [3].

Firefly algorithmis based on the flashing patterns and behavior of fireflies. Cuckoo search is based on the brood parasitism of some cuckoo species. And flower pollination algorithm is based on the pollination process of flowering plant. For an optimization problem, those algorithms are more efficient than earlier algorithms [3]. However, those algorithms not yet used to solve sequence alignment problem.

In this research, we will use firefly algorithm, cuckoo search, and flower pollination algorithm to do sequence alignment.

## II. Literature Review

### A. Sequence Alignment

When sequences evolve starting from a common ancestor, their residues can undergo substitutions (when residues are replaced by some other residues). Apart from substitutions, during the course of evolution sequences can accumulate a number of events of two more types: insertions (when new residues appear in a sequence in addition to the existing ones) and deletions (when some residues disappear). Therefore, when one is trying to produce the best possible alignment between two sequences, residues must be allowed to be aligned not only to other residues but also to gaps. The presence of a gap in an alignment represents either an insertion or deletion event. Consider, for example, the following two short nucleotide sequences, each consisting of only seven residues

$$\begin{aligned} x &: TACCAGT \\ y &: CCCGTAA \end{aligned} \tag{1}$$

The sequences are of the same length, and there is only one way to align them, if one does not allow gaps in alignments

$$\begin{aligned} x &: TACCAGT \\ y &: CCCGTAA. \end{aligned} \tag{2}$$

However, if we allow gaps, there are many possible alignments. In particular, the following alignment seems to be much more informative than the preceding one

$$\begin{aligned} x &: TACCAGT-- \\ y &: --CCCGTAA. \end{aligned} \tag{3}$$

The most popular scoring schemes assume independence among the columns in an alignment and set the total score of the alignment to be equal to the sum of the scores of each column. Therefore, for such schemes one only needs to specify the scores $s(a,b) = s(b,a)$ and the gap penalty $s(-,a) = s(a,-)$, with $a,b \in Q$, where $Q = A,C,G,T$ [1].

Manuscript received February 3, 2017; accepted February 22, 2017.

The authors are with the Department of Mathematics, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia. Email: shahab.luthfi@gmail.com, mii@its.ac.id

As an example of a scoring scheme one can set $s(a,a) = 1$ (the score of a match), $s(a,b) = -1$, if $a \neq b$ (the score of a mismatch), and $s(-,a) = s(a,-) = -2$ (the gap penalty). We will use this scoring scheme.

## *B. Needleman-Wunsch Algorithm*

Needleman-Wunsch algorithm always finds all optimal global alignments (there are frequently more than one such alignments).

The idea is to produce an optimal alignment from optimal alignments of subsequences. Algorithms that achieve optimization by means of performing optimization for smaller amounts of data (in this case subsequences) are generally called dynamic programming algorithms. Suppose we are given two sequences $x = x_1x_2...x_ix_n$ and $y = y_1y_2...y_jy_m$. We construct an $(n+1) \times (m+1)$ matrix $F$. Its $(i,j)$-th element $F(i,j)$ for $i = 1,...,n$, $j = 1,...,m$ is equal to the score of an optimal alignment between $x_1x_2...x_i$ and $y_1y_2...y_j$. The element $F(i,0)$ for $i = 1,...,n$ is the score of aligning $x_1x_2...x_i$ to a gap region of length $i$. Similarly, the element $F(0,j)$ for $j = 1,\cdots,m$ is the score of aligning $y_1y_2...y_j$ to a gap region of length $j$. We build $F$ recursively, initializing it by the condition $F(0,0) = 0$ and then proceeding to fill the matrix from the top left corner to the bottom right corner. If $F(i-1,j-1)$, $F(i-1,j)$ and $F(i,j-1)$ are known, $F(i,j)$ is clearly calculated as follows

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i,y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{4}$$

Indeed, there are three possible ways to obtain the best score $F(i,j): x_i$ can be aligned to $y_j$ (see the first option in the above formula), or $x_i$ is aligned to a gap (the second option), or $y_j$ is aligned to a gap (the third option).

Calculating $F(i,j)$ we keep a pointer to the option from which $F(i,j)$ was produced. When we reach $F(n,m)$ we trace back the pointers to recover optimal alignments. The value $F(n,m)$ is exactly their score. Note that more than one pointers may come out of a particular cell of the matrix which results in several optimal alignments [1].

## *C. Firefly Algorithm*

Firefly algorithm use the following three idealized rules [3]:

- All fireflies are unisex, so one firefly will be attracted to other fireflies regardless of their sex.
- Attractiveness is proportional to a fireflys brightness. Thus for any two flashing fireflies, the less brighter one will move toward the brighter one. The attractiveness is proportional to the brightness, both of which decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly.
- The brightness of firefly is affected or determined by the landscape of the objective function.

For a maximization problem, the brightness can simply be proportional to the value of the objective function. Based on these three rules, the basic steps of the firefly algorithm can be summarized as the pseudo code shown in Fig. 1.

## *D. Cuckoo Search*

Cuckoo search is one of the latest nature-inspired metaheuristic algorithms, developed in 2009 by Xin-She Yang and Suash Deb. Cuckoo search is based on the brood parasitism of some cuckoo spesies.

Cuckoo search use the following three idealized rules [3]:

- Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest.
- The best nests with high-quality eggs will be carried over the next generations.
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in (0,1)$. In this case, the host bird can either get rid of the egg or simply abandon the nest and build a completely new nest.

As a further approximation, this last assumption can be approximated by replacing a fraction $p_a$ of the $n$ host nests (with new random solutions). For a maximization problem, the quality or fitness of a solution can simply be proportional to the value of the objective function.

From the implementation point of view, we can use the following simple representations that each egg in a nest represents a solution, and each cuckoo can lay only one egg (thus representing one solution). The aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. Obviously, this algorithm can be extended to the more complicated case where each nest has multiple eggs representing a set of solutions. Here we use the simplest approach, where each nest has only a single egg. In this case, there is no distinction between an egg, a nest, or a cuckoo, since each nest corresponds to one egg, which also represents one cuckoo.

Based on these three rules, the basic steps of the cuckoo search can be summarized as the pseudo code shown in Fig. 2.

## *E. Flower Pollination Algorithm*

Flower pollination algorithm use the following three idealized rules [3]:

- Biotic and cross-pollination can be considered processes of global pollination.
- For local pollination, abiotic pollination and self-pollination are used.
- Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
- The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0,1]$, slightly biased toward local pollination.

Based on these four rules, the basic steps of the flower pollination algorithm can be summarized as the pseudo code shown in Fig. 3.

```
Generate initial population of $n$ fireflies
Define initial attractiveness $\alpha$
while $t < maxGeneration$
    for $i = 1 : n$
        for $j = 1 : n$
            if $I_i < I_j$
                Move firefly $i$ towards firefly $j$
            end
        Update $\alpha$
        end
    end
    Rank the fireflies and find the current global best $g_*$
    Update $t$
end
Show the results
```

Fig. 1. Pseudo code of firefly algorithm

```
Generate initial population of $n$ nests
Find the current best nest $g_*$
Define a probability $p_a \in (0,1)$
while $t < maxGeneration$
    Get a nest randomly
    Generate a new nest using the global random walk
    Evaluate its solution quality or objective value $f_i$
    Chose a nest among $n$ (say nest $j$) randomly
    if $f_i < f_j$
        Replace nest $j$ by the new nest $i$
    end
    A fraction $p_a$ of worse nests are abandoned
    New nests are generated using local random walk
    Rank the nests and find the current global best $g_*$
    Update $t$
End
Show the results
```

Fig. 2. Pseudo code of cuckoo search

```
Generate initial population of $n$ flowers
Find the current best flower $g_*$
Define a switch probability $p \in [0,1]$
while $t < maxGeneration$
    for $i = 1 : n$
        if $rand < p$
            Generate new flower using global pollination
        else
            Generate new flower using local pollination
        end
        Evaluate the objective value of new flower
        if new solution is better
            Update flower $i$ by new solution
        end
    end
    Rank the flowers and find the current global best $g_*$
    Update t
End
Show the results
```

Fig. 3. Pseudo code of flower pollination algorithm

## III. Firefly Algorithm, Cuckoo Search, and Flower Pollination Algorithm for Sequence Alignment

### A. *Solution Representation*

Solution representation for firefly algorithm, cuckoo search, and flower pollination algorithm is the solution of sequence alignment it self. It means that the solution is the two initial sequences those are inserted by gap symbols until the length of the both sequences are equal. This solution representation is easier to processed and we dont need to do a transformation from the solution representation to the sequence alignment solution [4].

### B. *Generate New Solution*

In this paper, we use two schemes to produce new solutions. In the first scheme, we choose a gap symbol randomly and then put it to a new place randomly in the same sequence.

In the second scheme, first, we choose two solutions randomly. The first solution is cut straight at some randomly chosen position and the second one is tailored so that the right and the left pieces of each solution can be joined together while keeping the original sequence. Any void space that appears at the junction point is filled with gap symbols. Because of the specificity of this junction point, where rearrangements can occur, this operator combines both the traditional properties of a crossover and those of a local rearrangement mutation [4].

### C. *Implementation*

The implementation in this research was written in Java language programming using NetBeans IDE 8.2. For each case, we do five independent runs and record them, and then pick the best score as the score of algorithms. And we take the average time of that five runs.

Firefly algorithm uses 10 solutions and 2000 iterations. Cuckoo search uses 100 solutions and 1000 iterations. And flower pollination algorithm uses 100 solutions and 1000 iterations.

### D. *Some Modifications*

In the firefly algorithm, firefly $i$ will move toward firefly $j$ if firefly $j$ is better than firefly $i$. We modify it, although firefly $j$ is not better than firefly $i$, firefly $i$ will move toward firefly $j$. However, that new solution will bi accepted if that new solution is better than the old one.

In the cuckoo search, global random walk will be accepted if the new solution is better than the old one. That mecanism will be replaced by elitism replacement with filtration [4], that will make all of better solutions are not abandoned. And the new solution from local random walk will not always replace the old one. That solution will be accepted if is better than the old one.

In the flower pollination algorithm, new solution will be accepted if the new solution is better than the old one. That mecanism will be replaced by elitism replacement with filtration.

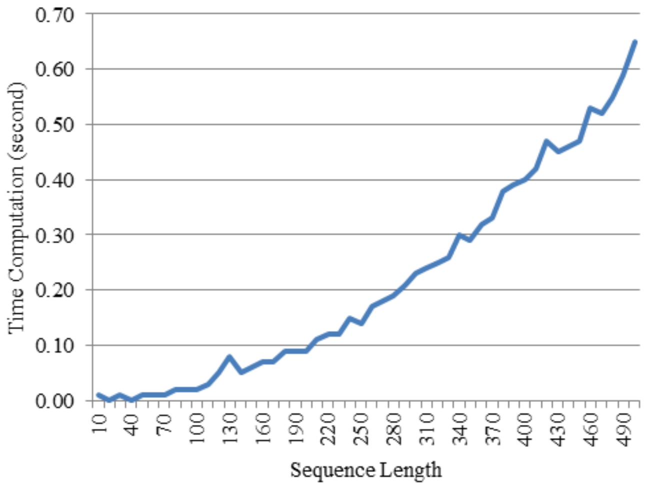

Fig. 4. Time computation of Needleman-Wunsch Algorithm

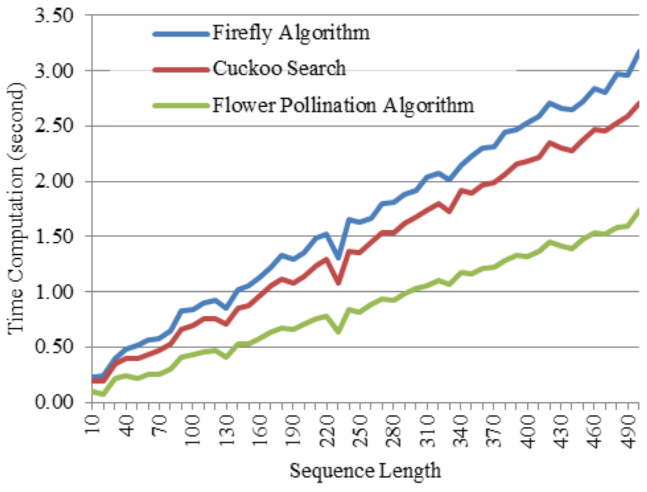

Fig. 5. Time computation of Firefly Algorithm, Cuckoo Search, and Flower Pollination Algorithm

## IV. Results and Discussions

### A. Sequence

We create 50 pairs of sequences as a experimental data. To create a pair of sequences, first we create a parent sequence randomly. Then we create the first and second sequence by take a mutations to the parent sequence. We use 4 types of mutations [5]. We create pairs of sequences which have length vary from 10 to 500.

### B. The Comparison of Needleman Wunsch Algorithm, Firefly Algorithm, Cuckoo Search, and Flower Pollination Algorithm

To do an alignment, we create some sequences as an input. We create pair of sequences which have length vary from 10 to 500.

The score of sequence alignment from Needleman-Wunsch algorithm, firefly algorithm, cuckoo search, and flower pollination algorithm is showed in Table 1.

From Table 1, we can see that firefly algorithm, cuckoo search, and flower pollination algorithm can find optimal solutions of sequence alignment because the score of those algorithms is equal with the score of Needleman Wunsch algorithm.

TABLE I
Scores of Needleman-Wunsch Algorithm, Firefly Algorithm, Cuckoo Search, and Flower Pollination Algorithm

| Needleman-Wunsch Algorithm | Firefly Algorithm | Cuckoo Search | Flower Pollination Algorithm |
|---|---|---|---|
| 3 | 3 | 3 | 3 |
| 9 | 9 | 9 | 9 |
| 7 | 7 | 7 | 7 |
| 16 | 16 | 16 | 16 |
| 25 | 25 | 25 | 25 |
| 20 | 20 | 20 | 20 |
| 30 | 30 | 30 | 30 |
| 32 | 32 | 32 | 32 |
| 28 | 28 | 28 | 28 |
| 44 | 44 | 44 | 44 |
| 49 | 49 | 49 | 49 |
| 49 | 49 | 49 | 49 |
| 57 | 57 | 57 | 57 |
| 61 | 61 | 61 | 61 |
| 60 | 60 | 60 | 60 |
| 59 | 59 | 59 | 59 |
| 72 | 72 | 72 | 72 |
| 81 | 81 | 81 | 81 |
| 82 | 82 | 82 | 82 |
| 82 | 82 | 82 | 82 |
| 83 | 83 | 83 | 83 |
| 100 | 100 | 100 | 100 |
| 88 | 88 | 88 | 88 |
| 114 | 114 | 114 | 114 |
| 114 | 114 | 114 | 114 |
| 107 | 107 | 107 | 107 |
| 121 | 121 | 121 | 121 |
| 119 | 119 | 119 | 119 |
| 117 | 117 | 117 | 117 |
| 123 | 123 | 123 | 123 |
| 132 | 132 | 132 | 132 |
| 143 | 143 | 143 | 143 |
| 130 | 130 | 130 | 130 |
| 138 | 138 | 138 | 138 |
| 145 | 145 | 145 | 145 |
| 151 | 151 | 151 | 151 |
| 160 | 160 | 160 | 160 |
| 174 | 174 | 174 | 174 |
| 159 | 159 | 159 | 159 |
| 182 | 182 | 182 | 182 |
| 177 | 177 | 177 | 177 |
| 177 | 177 | 177 | 177 |
| 205 | 205 | 205 | 205 |
| 188 | 188 | 188 | 188 |
| 185 | 185 | 185 | 185 |
| 194 | 194 | 194 | 194 |
| 192 | 192 | 192 | 192 |
| 201 | 201 | 201 | 201 |
| 219 | 219 | 219 | 219 |
| 222 | 222 | 222 | 222 |

Time computation for Needleman-Wunsch algorithm can be seen in Fig. 4. We can see that the time computation function does not follow the linear functions.

The average time computation of firefly algorithm is 1.675 second. The average time computation of cuckoo search is 1.431 second. The average time computation of flower pollination algorithm is 0.876 second. The increase of time computation along with the increase of the sequence length follow the linear functions. Time computation for firefly algorithm, cuckoo search, and flower pollination algorithm can be seen in Fig. 5.

## REFERENCES

[1] A. Isaev, *Introduction to mathematical methods in bioinformatics*. Springer Science & Business Media, 2006.

[2] M. Shahab, D. Utomo, and M. Irawan, "Decomposing and solving capacitated vehicle routing problem (CVRP) using two-step genetic algorithm (TSGA)," *Journal of Theoretical and Applied Information Technology*, vol. 87, no. 3, pp. 461–468, 2016.

[3] X.-S. Yang, *Nature-inspired optimization algorithms*. Elsevier, 2014.

[4] C. Notredame and D. Higgins, "Saga: sequence alignment by genetic algorithm," *Nucleic acids research*, vol. 24, no. 8, pp. 1515–1524, 1996.

[5] S. Shen, *Theory and Mathematical methods in Bioinformatics*. Springer Science & Business Media, 2008.