World Academy of Science, Engineering and Technology
International Journal of Bioengineering and Life Sciences
Vol:11, No:6, 2017

BeamGA Median: A Hybrid Heuristic Search Approach

Ghada Badr, Manar Hosny, Nuha Bintayyash, Eman Albilali, Souad Larabi Marie-Sainte

***Abstract*—**The median problem is significantly applied to derive the most reasonable rearrangement phylogenetic tree for many species. More specifically, the problem is concerned with finding a permutation that minimizes the sum of distances between itself and a set of three signed permutations. Genomes with equal number of genes but different order can be represented as permutations. In this paper, an algorithm, namely BeamGA median, is proposed that combines a heuristic search approach (local beam) as an initialization step to generate a number of solutions, and then a Genetic Algorithm (GA) is applied in order to refine the solutions, aiming to achieve a better median with the smallest possible reversal distance from the three original permutations. In this approach, any genome rearrangement distance can be applied. In this paper, we use the reversal distance. To the best of our knowledge, the proposed approach was not applied before for solving the median problem. Our approach considers true biological evolution scenario by applying the concept of common intervals during the GA optimization process. This allows us to imitate a true biological behavior and enhance genetic approach time convergence. We were able to handle permutations with a large number of genes, within an acceptable time performance and with same or better accuracy as compared to existing algorithms.

***Keywords*—**Median problem, phylogenetic tree, permutation, genetic algorithm, beam search, genome rearrangement distance.

I. Introduction

IN early 1970s, since the beginning stages of the studies of computational molecular biology, scientists and researchers have been focusing on the DNA and amino acid arrangements and how to analyze them. In this field, gene prediction, similarity searching, and phylogeny reconstruction are the most tackled problems [1]. Change in evolution of genes is the key element of the solutions to these problems. Through evolutionary history, the modifications of genomes and the relationship of genome construction and role in various biological species or subspecies are being studied in comparative genomics. Comparative genomics is one of the fields of computational molecular biology, which is basically based on studying variations in the order and content of genes in the genomes of associated organisms. Comparative genomics has many applications such as building comparative genomic maps, rebuilding phylogenetic associations between organisms, and estimating the comparative frequencies of genome rearrangement methods [1].

This paper focuses on devising a heuristic approach for solving the median problem, which is a well-known problem in bioinformatics. We examine the opportunities to overcome the limitations of other approaches with respect to speed and/or accuracy. The previous approaches handled a limited number of genes in each permutation, due to the exponential time complexity of the problem. In our approach, we aim to handle a large number of genes within an acceptable time performance with a similar accuracy rate as compared to existing algorithms. The proposed approach is a heuristic initialization step followed by a Genetic Algorithm. To the best of our knowledge, this approach was not applied before for this biological problem. The computational results show that the algorithm is comparable to previous solution approaches with respect to quality of the obtained solution.

The paper organization is as follows: Section II presents the state of the art studies related to the problem under discussion. Section III describes the median problem. Section IV introduces the proposed method BeamGA. Section V addresses the experimental results and finally Section VI concludes this research work.

II. Related Studies

Computing the distances between genomes is quite difficult, and has been proved to be computationally intensive and suspected to be NP-hard [3]. However, the Median problem has two important properties that could help in finding good solutions with reasonable computational effort: the non-uniqueness of the problem solution, and the probability to find the median on or near the XX gene orders rather than the center.

Sankoff and Blanchette [4] proposed an algorithm for the median problem based on breakpoint distance. They reduced the breakpoint median problem to a particular case of the Traveling Salesman Problem (TSP). The breakpoint distance measure has some disadvantages such as lacking simple biological explanation, as it does not associate in a straight line to any real rearrangement method, even though it is helpful as a heuristic measure. Given these disadvantages, Siepel [1] devised an efficient branch-and-bound exact algorithm for the reversal median problem that deals with the inversion rearrangement method. Siepel's algorithm relies on bounds that are calculated with the metric feature of reversal distance only. Although they cost more to compute, when reversal

G. Badr is with faculty of Engineering, University of Ottawa, Canada (e-mail: gbadr@uottawa.ca). She is also with RI, The City of Scientific Research and Technological Applications, University and Research District, P.O. 21934 New Borg Alarab, Alex, Egypt.

M. Hosny and E. Albilali are with Computer Science department, College of Computer and Information Sciences, King Saud University, Riyadh, KSA.

N. Bintayyash is with Information Technology department, College of Computer and Information Sciences, King Saud University, Riyadh, KSA.

S. Larabi Marie-Sainte is with Computer Science department, College of Computer and Information Sciences, Prince Sultan University, Riyadh, KSA (e-mail: slarabi@psu.edu.sa).

medians are compared to breakpoint medians regarding the inversion distance, breakpoint medians are weaker [1]. Moreover, Eriksen [6] proved that the reversal median algorithm cannot provide efficient results when the distance between the species is reasonably large. This result has conducted the researchers toward two directions. To solve the median problem, they used another evolutionary distance and/or applied the rearrangement operations on a common interval. Common intervals detect sets of genes that take place successively in all input gene orders. Matthias Bernt et al. [5] proposed an exact algorithm to solve reversal median problem using common interval for three given gene orders. The authors used the preserving minimum reversal distance between two gene orders which is an NP-hard problem (whereas the same problem with reversals that are not necessarily preserving can be solved in polynomial time). The preserving reversal distance is the minimum number of reversals that preserve all common intervals between two given gene orders.

Bader [7] addressed a new method to solve the weighted reversal and transposition medians. It consists of extending Caprara's median solver by using a new branch and bound algorithm.

Using breakpoint distance, reversal distance or any evolutionary distance, the median problem is known to be NP-hard [1], [2]. Hence, exact algorithms become prohibitive for large problem sizes, and heuristic approaches are considered as an attractive alternative in such cases.

Rajan et al. [8] proposed greedy heuristic based on DCJ median. The proposed method can deal with large size genomes. However, it cannot handle the length of inversions.

Among the heuristic approaches, Evolutionary Algorithms are well known techniques to efficiently solve complex and hard problems due to their good exploration and exploitation of the search space. Our idea is to solve the rearrangement genome median problem using an evolutionary algorithm, Genetic Algorithm, and taking into account the common interval. This direction has only been recently treated by [9] in his thesis at the end of 2014. The author proposed a method based on DCJ sorting and Indel operations using Genetic Algorithm, called GA-DCJ for solving unequal genomes median problem (without duplication). Our proposed method uses reversal distance to solve equal genomes median problem.

III. The Median Problem

Input: given three signed permutations π_1, π_2, and π_3 that represent three different taxa.
Output: finding the fourth permutation (median) π_ϕ, with the smallest possible distance score $S(\phi)$ from the three original permutations as shown in Fig. 1 and explained as follows:
$S(\phi) = d_{\pi_1,\phi} + d_{\pi_2,\phi} + d_{\pi_3,\phi}$,
$S(\phi) \geq M_{min}$, $S(\phi) \leq M_{max}$
$M_{min} = [\frac{d_{\pi_1,\pi_2}+d_{\pi_1,\pi_3}+d_{\pi_2,\pi_3}}{2}\ ]$
$M_{max} = min\{(d_{\pi_1,\pi_2}+d_{\pi_2,\pi_3})+(d_{\pi_1,\pi_2}+d_{\pi_1,\pi_3})+(d_{\pi_2,\pi_3}+d_{\pi_1,\pi_3})\}$

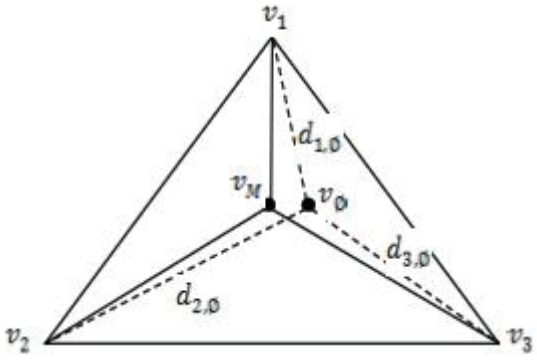

Fig. 1 The Median Problem representation [1]

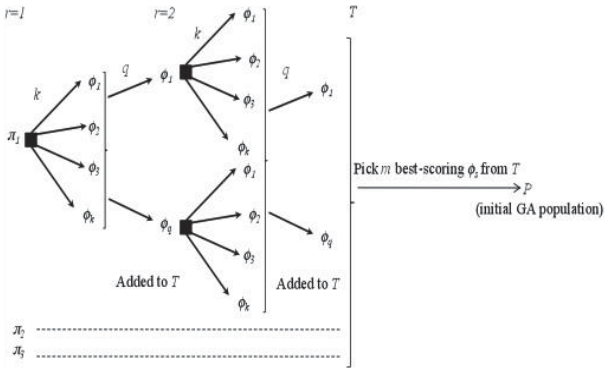

Fig. 2 Initialization Phase

IV. Method: BeamGA

The BeamGA is based on two phases, the Initialization phase, where a beam search algorithm is applied in order to generate the initial population. This is followed by an optimization phase, where a genetic algorithm is applied on common intervals in order to enhance the median score.

A. Initialization Phase: Beam Search

Given three signed permutations π_1, π_2, and π_3, and:

- i: The number of random reversals that can be applied to get the three original permutations from the identity permutation for every taxon (2, 6, or 10) π.
- $i/2$: The maximum levels number of reversals to be applied on π.
- $r \leq i/2$: the current level.
- k: The number of feasible neighbors f_s that can be generated from π.
- $q < k$: The number of best-scoring neighbors f_s that are added to priority queue T.
- m: maximum population size.
- n: number of genes.

In the initialization phase, we generate a population of size m by using a beam search [10]. The search starts by initializing a priority queue T to be empty. For each π_i, we generate k neighbors by performing one rearrangement operation on the previous permutation, and put the best generated q from k neighbors in T based on their median scores. The process is repeated for r levels as displayed in Fig.

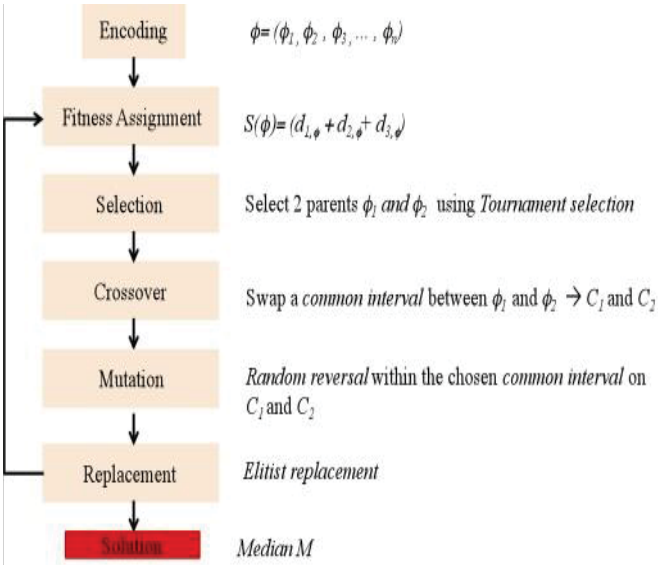

Fig. 3 Genetic Algorithm

2. Finally, we pick m best-scoring $S(\phi)$ from T and add them to P (the initial GA population in the optimization phase).

B. Optimization Phase: Genetic Algorithm (GA)

Given an initial population P, each individual in the population (a possible median ϕ) is assigned a fitness value based on its median score $S(\phi)$ with respect to the three original permutations. Then a genetic algorithm is applied. Parents are selected using tournament selection, crossover is performed by swapping a randomly chosen common intervals between the selected parents to produce two children, and mutation is done on each child by a random rearrangement operation (e.g reversal) within a randomly chosen common interval. To generate the new population, elitist replacement chooses the best individuals from the new and the old generations. The GA is repeated until the perfect median is reached or no improvement can be achieved for a number of iterations. If both conditions cannot be satisfied, the GA continues for a pre-specified number of iterations. The best solution in the final population is returned as the best median. Fig. 3 shows the different steps of GA and Algorithm1 presents the pseudocode of BeamGA.

```
/*———— Initialization phase ————*/
π1, π2, and π3
T = 0
for πi i : 1 to 3 do
    for r : 1 to i/2 do
        for j : 1 to k do
            S(φ)=GRO (πi)
            T=push S(φ)
        end
        Pick best q from k neighbors in T based on GRD
    end
end
P=m best-scoring S(φ) from T
/*———— Optimization Phase ————*/
Repeat
for i : 1 to m do
    (φ1,φ2)=Select parents (m)
    Crossover Px: (φ1,φ2)
        1) cm ← Find CM(φ1,φ2)
        2) x ← random CM(cm)
        3) (C1, C2) ← Swap CM (φ1,φ2)
        4) Mutation Pm: (C1,C2)
            a) cm ← Find CM(C1,C2)
            b) x ← random CM(cm)
            c) (C1, C2) ← GRO with in CM x (C1,C2)
        5) New ←(C1,C2)
end
P ← Elitisim strategy (P,New)
Until (iteration = 100 Or S(φ)= Mmin Or iteration with out
 improvment=10)
```

Algorithm 1: BeamGA pseudocode

V. Emperical Results

A. Experimental Setup

Extensive experiments are conducted on simulated data to evaluate the performance of our algorithm compared with previous algorithms. In order to evaluate our algorithm, performance analysis is performed and median accuracy score is compared with other algorithms. The computer used in these experiments has an Intel(R) CORE(TM) i5-3337U CPU with clock speed of 1.80 GHz 2 Cores 4 logical Processors, 64-bit operating system, and 4 GB RAM memory and the code was implemented using MATLAB R2010a.

In initialization phase, test data set of three signed permutation is constructed by applying random reversals on identity permutation. This simulates the biological evolution such that each set is a three taxa generated by applying reversal on shared ancestors. Three parameters are used to control generated data sets n, s and i. n represents the number of genes in each permutation. s represents the number of data set generated and i represent the number of reversal applied to generate every taxon. The number of reversals applied on original set to generate neighbors are kept i/2 throughout the experiments.

The experiments are divided into two types:

1) Testing variation of n and i: test data consists of three different set of three signed permutation (taxon) with different size of n and different values of i. i represents the number of reversal applied on the identity permutation to generate the three texa. n represents the number of genes in each permutation which can hold 25, 50, 75, 100 and 125. The value of i for each n is 2, 6 and 10.
2) Testing variation of i for the same n: test data consists of three different sets of three signed permutations (taxon) where n is 25 and i have the values: 2, 3, 4, 5 and 6.

The probability of crossover p_x and the probability of mutation p_m were tuned until most accurate median with respect to time. p_x was set to 0.8 while p_m was set to 0.05 throughout the whole experiments. Moreover, some other parameters k, q and T values are altered in each experiment. k represents the numbers of neighbors generated from each permutation by applying random reversal. Initially, k was set to be 50% of n, while q which represents the best scoring

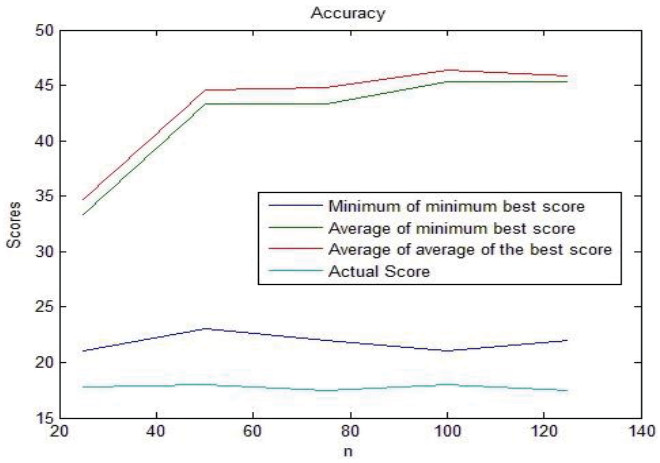

Fig. 4 The accuracy of the median score found through five experiments of i=6 with five different values of *n: 25, 50, 75, 100 and 125*, where $k = 14$, $q = 7$, the size of priority queue T is $3 * q * r$

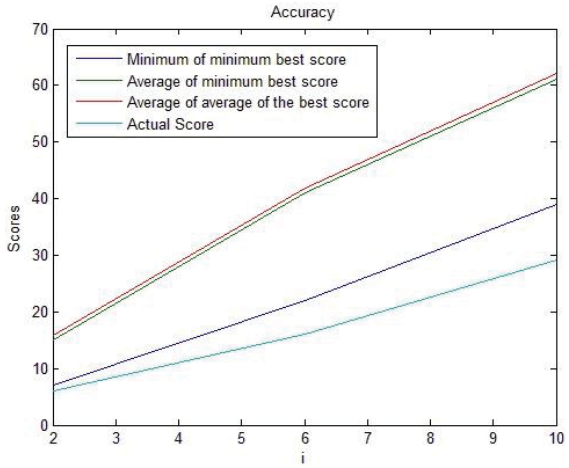

Fig. 5 The accuracy of the median score found through three experiments of *n=50* with three different values of *i: 2, 6, and 10*, where $k = 14$, $q = 7$, the size of priority queue T is $3 * q * r$

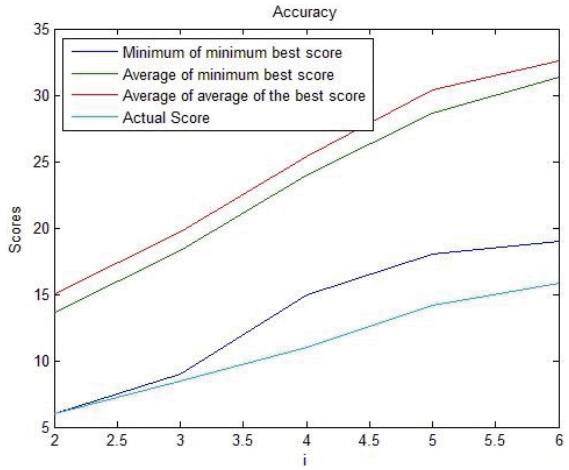

Fig. 6 The accuracy of the median score found through five experiments of *n=25* with five different values of *n: 2, 3, 4, 5 and 6*, where $k = 14$, $q = 7$, the size of priority queue T is $3 * q * r$

permutation is set to 50% of k (beam width). After that, we change the value of k and q to 14 and 7 respectively to reduce the initialization phase time. As a result, the performance increased and the accuracy measures remained the same. Throughout the experiments we evaluate the accuracy and the performance using various measures:

1) The actual score which is the score obtained from the identity permutation used to generate the set.
2) The minimum of the minimum best score median found for 10 runs of the three data sets.
3) The average of minimum best score median found for 10 runs of the three data sets.
4) The average of the best score median found for 10 runs of the three data sets.
5) The average time in seconds for the initialization time, GA time and the total time through the 10 runs of the three data sets.

The experiments are divided into two stages:

1) The number of neighbours generated k is equal to 14, beam width q is set to 7, the size of the priority queue T is set to $q * r * 3$ and the number of iteration without improvement is set to n.
2) The parameter k is set to 14, q is set to 7, the size of the priority queue T is set to 30 and the number of iteration without improvement is set to n.

B. Experimental Results

Fig. 4 depicts the accuracy of the median score when *i=6* and *n* have five different values: *25, 50, 75, 100 and 125*. For each value of *n*, three sets are generated such that each set run 10 times. After that, the average is calculated for the 10 runs of each set. Then, the average of the three sets is computed. The figure shows the minimum of minimum best score is close to the actual score for all values of *n*.

Fig. 5 demonstrates the accuracy of the median score when *n=50* and *i* have three different values: 2, 6 and 10. When *i=2* the minimum of minimum best score almost matches the actual score. The distance between them increases as the value of *i* increases.

Fig. 6 demonstrates the accuracy of median score of five experiments of *i= 2, 3, 4, 5 and 6* and *n=25*. The minimum of minimum best score is identical to the actual score when *i=2*. As the value of *i* increases the difference between them increases.

Fig. 7 shows that our algorithm requires less than 1.8 seconds when *n=25*. Moreover, the time decreases to 1.733 when *n=50* .After that, the time increases proportional to the value of *n*. initialization phase requires around 1 second for various values of *n* while GA time increases gradually as the value of n increases.

Fig. 8 depicts the performance of our algorithm where *i=2, 6 and 10* and $n = 50$. GA time remains steady throughout the experiment. The total time follows the pattern of the initialization time which increases proportional to the value of *i*.

Initialization phase outperforms the GA phase in performance when *n=25* and *i=2,3,4,5 and 6* as depicted in

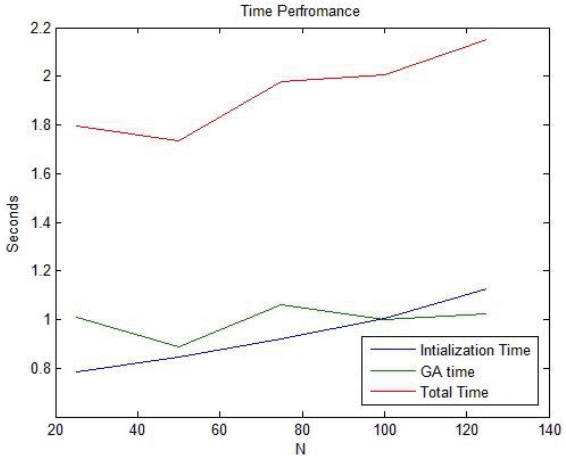

Fig. 7 The time performance of the median score found through five experiments of i=6 with five different values of *n: 25, 50, 75, 100 and 125*, where $k = 14$, $q = 7$, the size of priority queue T is $3 * q * r$

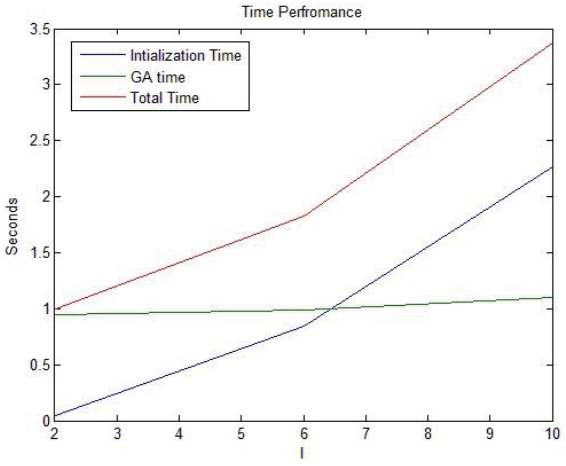

Fig. 8 The time performance of the median score found through three experiments of *n=50* with three different values of *i: 2,6, and 10*, where $k = 14$, $q = 7$, the size of priority queue T is $3 * q * r$

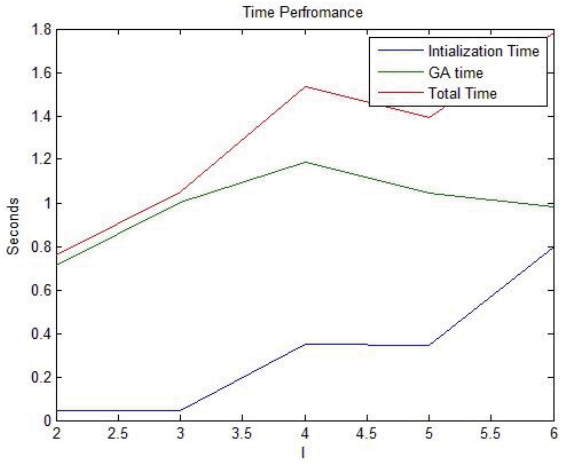

Fig. 9 The time performance of the median score found through five experiments of *n=25* with five different values of *n: 2, 3, 4, 5 and 6*, where $k = 14$, $q = 7$, the size of priority queue T is $3 * q * r$

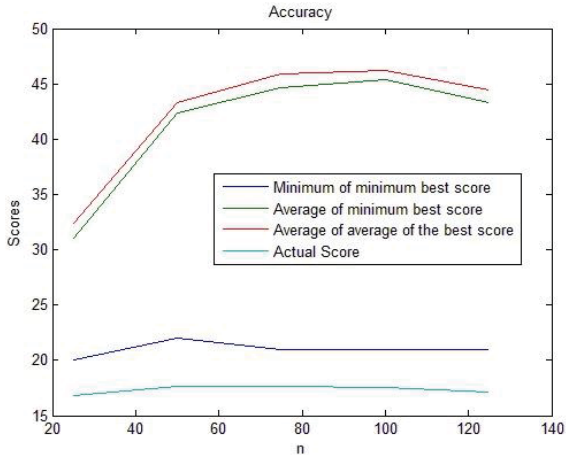

Fig. 10 The accuracy of the median score found through five experiments of i=6 with five different values of *n: 25, 50, 75, 100 and 125*, where $k = 14$, $q = 7$, the size of priority queue T is 30

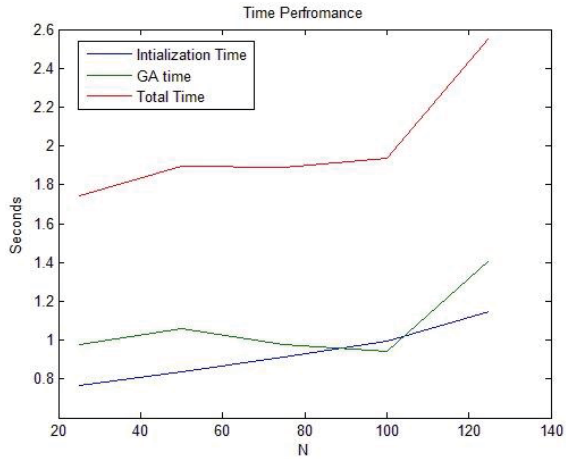

Fig. 11 The time performance of the median score found through five experiments of i=6 with five different values of *n: 25, 50, 75, 100 and 125*, where $k = 14$, $q = 7$, the size of priority queue T is 30

Fig. 9. The algorithm runs in 0.8 seconds to 1.8 second for different values of i.

The accuracy in Figs. 4 and 10 are quite identical even when the priority queue differs. However, the performance measures show better results when $T=3*q*r$ as demonstrated in Figs. 7 and 11.

For various values of i the accuracy measures are identical when $T = 3 * q * r$ and $T=30$ as shown in Figs. 5 and 12. However, when GA performance remains steady around 1 second throughout the experiment when $T = 3 * q * r$ as depicted in Fig. 8, the GA time decreases from less than 1.5 seconds when i=2 to less the 1 second when $i=6$ and $T=30$ as demonstrated in Fig. 13. After that, the time remains steady.

The minimum of minimum best score is near to the actual score in Fig. 14 than in Fig. 6 when $n=25$ and $i=2, 3, 4, 5$ *and 6*. In addition, the performance measures follow the same pattern with better results when $T=30$ as illustrated in Figs. 9 and 15.

Figs. 9 and 11 show the time performance recorded for the previous experiments. In Fig. 11, the total time recorded

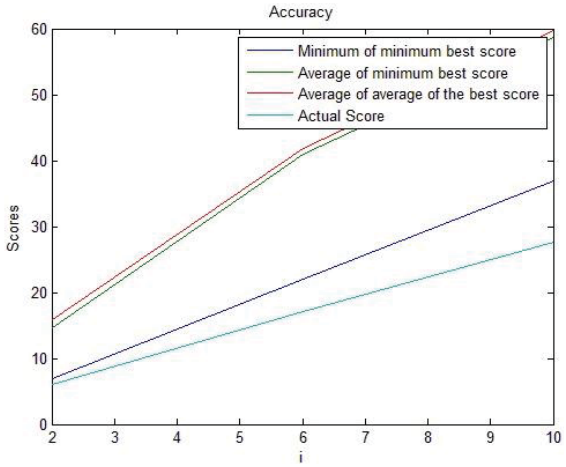

Fig. 12 The accuracy of the median score found through three experiments of *n=50* with three different values of *i: 2,6, and 10*, where *k = 14, q = 7*, the size of priority queue T is 30

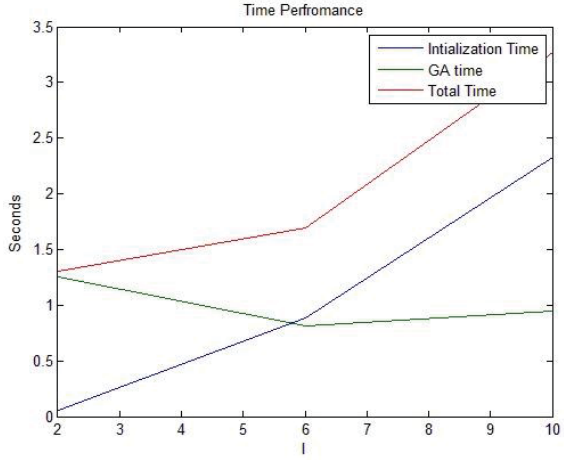

Fig. 13 The time performance of the median score found through three experiments of *n=50* with three different values of *i: 2,6, and 10*, where *k = 14, q = 7*, the size of priority queue T is 30

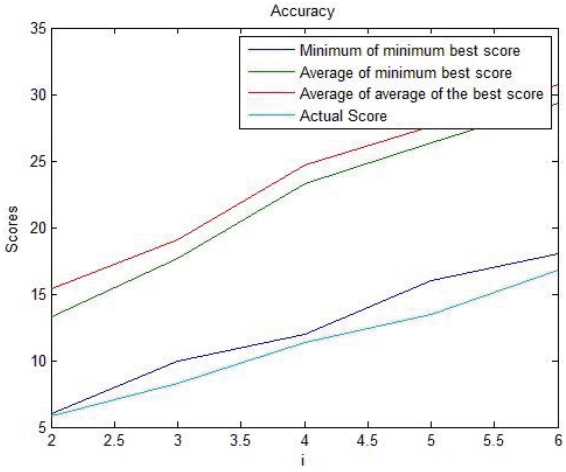

Fig. 14 The accuracy of the median score found through five experiments of *n=25* with five different values of *i: 2, 3, 4, 5 and 6*, where *k = 14, q = 7*, the size of priority queue T is 30

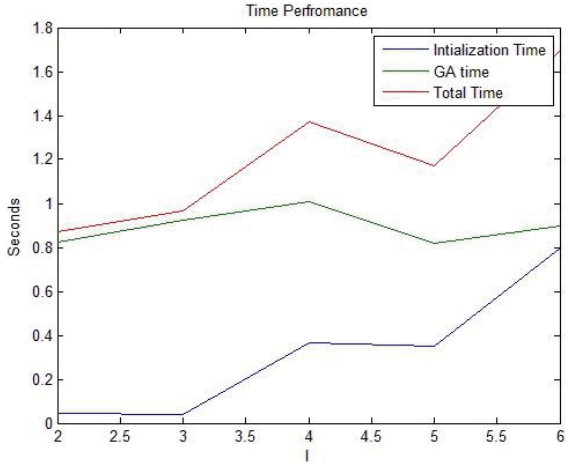

Fig. 15 The time performance of the median score found through five experiments of *n=25* with five different values of *i: 2, 3, 4, 5 and 6*, where k = 14, q = 7, the size of priority queue T is 30

is less than 4.5 second while the time increased directly proportional with the growing values of *n*. Fig. 9 shows that the time of initialization phase and total time increase when i values increase but GA time starts with 1 second for *i=3* then decreases to almost .8 for *i=6* and remains steady.

VI. Conclusion

We proposed a heuristic approach for solving the median problem. The accuracy of BeamGA for the median score is excellent when compared to the actual score. Further optimization can be added to the initialization phase to enhance its time performance. For more accurate assessment, the performance needs to be tested on real biological data.

References

[1] A. C. Siepel, *Exact Algorithms for the Reversal Median Problem*, Master's thesis, University Of New Mexico, Albuquerque, New Mexi-co, December, 2001.

[2] E. Ohlebush, M. I. Abouelhoda, K. Hockel and J. Stallkamp, *The Median Problem for the Reversal Distance in Circular Bacterial Ge-nomes*, in Proceedings of the 2005 international conference on Com-parative genomics, Ottawa, Canada, pp. 240-251, 2005.

[3] J. P. P. Zanetti, P. Biller and J. Meidanis, *On the Matrix Median Problem*. In Algorithms in Bioinformatics, pp. 230-243, Springer Berlin Heidelberg, 2013.

[4] D. Sankoff and M. Blanchette, *Multiple genome rearrangement and breakpoint phylogeny*, Journal of Computational Biology, Vol. 5(3), pp. 555-570, 1998.

[5] M. Bernt, D. Merkle and M. Middendorf, *Genome Rearrangement Based on Reversals that Preserve Conserved Intervals*, IEEE/ACM Trans. Comput. Biol. Bioinformatics 3, Vol. 3, 275-288, 2006. DOI=10.1109/TCBB.2006.38 http://dx.doi.org/10.1109/TCBB.2006.38

[6] N. Eriksen, *Reversal and transposition medians*, Theoretical Computer Science, Vol. 374(1-3), pp. 111-126, 2007.

[7] M. Bader, *On Reversal and Transposition Medians*, Ulmer Informatik-Berichte, 2009.

[8] V. Rajan, A. Xu, Y. Lin, K. M. Swenson and B. ME Moret, *Heuristics for the inversion median problem*, BMC Bioinformatics, 2010.

[9] N. Gao, *Using Genetic Algorithm to solve Median Problem and Phylogenetic Inference*, Doctoral dissertation, Retrieved from: http://scholarcommons.sc.edu/etd/2825

[10] D. Furcy and S. Koenig, *Limited Discrepancy Beam Search*, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, UK, 2005.