A Genetic Approach with a Simple Fitness Function for Sorting Unsigned Permutations by Reversals

José Luis Soncco Álvarez
Department of Computer Science
University of Brasilia
Brasilia, D.F., Brazil
Email: josesoal@hotmail.com

Mauricio Ayala-Rincón
Departments of Computer Science and
Mathematics, University of Brasilia
Brasilia, D.F., Brazil
Email: ayala@unb.br

***Abstract*—Sorting unsigned permutations by reversals is an important and difficult problem in combinatorial processing of permutations with important applications in bio-informatics for the interpretation of the evolutionary relationship between organisms. Since it was shown that the problem is NP-hard many approximation and a few evolutionary algorithms were proposed. In this paper we propose a new genetic algorithm approach that uses modified crossover and mutation operators adapted to the problem. Instead previous genetic algorithmic approaches, the proposed algorithm uses a very simple fitness function that can be linearly computed in the size of the permutation and updated in constant time, for each individual in each generation. In order to compare the accuracy of the computed solutions, an 1.5 approximation ratio algorithm was developed by fixing Christie's approximation method. The results showed that on average the proposed genetic approach produces competitive results in relation with the ones given by the 1.5-approximation algorithm. Additionally, it has been observed that for permutations of all sizes, that were randomly generated, it was alway possible to compute better solutions with the genetic than with the approximate approach and that the difference obtained for these cases is greater than the ones obtained in the cases in which the genetic approach has a worse behavior than the approximate one.**

I. Introduction

The evolutionary relationship of two organisms can be determined by comparing two biological sequences, but classical algorithms only take into account local mutations (deletions, insertions and substitutions) and do not consider global rearrangements (reversals, transpositions, translocations, etc). Reversals are the most commonly observed mechanisms of genome rearrangements to transform one genome into another.

The order of genes in a genome can be represented as a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of the set $\{1...n\}$ where n is the number of genes. There are two types of permutations: signed and unsigned permutations. On signed permutations, each π_i has a positive or negative sign reflecting its orientation within the genome.

Given two permutations we wish to determine the minimum number of reversals to transform one permutation into another, that is the reversal distance between two permutations. If one of the permutations is the identity permutation (permutation sorted in increasing order), the problem is known as sorting by reversals.

For the problem of sorting signed permutation, initially, Kececioglu and Sankoff [1] conjectured that the problem was NP-hard and proposed a 2-approximation algorithm. Afterwards, Bafna and Pevzner [2] improved the radio to 1.5, by using the data structure of breakpoint graphs. Finally, Hannenhalli and Pevzner [3] gave an exact polynomial ($O(n^4)$) algorithm. Further, more efficient algorithms have been introduced (e.g., [4], [5]).

For unsigned permutations, that are the ones treated in this paper, the problem was shown to be NP-hard by Caprara [6]. Before the complexity was known, Kececioglu and Sankoff [1] gave a 2-approximation algorithm. Later on, the radio was improved to 1.5 by Christie [7] and then to 1.375 by Berman, Hannenhalli and Karpinski [8]. The last approximation algorithm is of theoretical interest being its practical implementation of great difficulty. Thus, in this paper we have fixed some imprecisions in Christie 1.5-approximation approach, that were not pointed out by other authors dealing with this approximation algorithm as a control mechanism for genetic solutions. The resulting 1.5-approximation algorithm has been directly implemented in order to have an accurate mechanism of control of the solutions computed by the proposed genetic approach.

Because of its complexity, exploration of evolutionary algorithms for the problem of sorting unsigned permutations by reversals is of great interest. Auyeung and Abraham [9] suggested a genetic algorithmic (GA) approach to solve the problem of sorting unsigned permutations by reversals based on mapping unsigned permutations of size n into a subset of the 2^n possible signed versions of the permutations. For a given unsigned permutation, a set of signed permutations is generated by randomly assigning either a positive or a negative signal to each component of the permutation. An exact sorting solution of a signed version of a permutation corresponds to an approximate sorting solution of the unsigned permutation. The fitness function of each signed permutation is given by its exact reversal distance that is computed by Hannenhalli's et al. method. Using GA techniques this combinatorial space is explored. Subsequently improvements to Auyeung's et al method were published in [10], but without changing the central premisses of this approach. More recently, Ghaffarizadeh, Ahmadi and Flann [11] used a modified version of

the standard GA using individuals of different sizes to reduce the runtime of the algorithm. All these approaches have been reported to improve the results obtained applying Christie's 1.5-approximation algorithm, but no mention was given on the imprecisions presented neither in Christie's original paper nor in its subsequently published erratum.

In this paper we propose a new GA approach, which initially, uses individuals of the same "size" to find a good solution and, after that, the mutation operator is used generating permutations of different "sizes" in order to improve the first computed solution. The distinguished feature of this GA approach, is that individuals correspond to partial approximate sorting by reversal solutions. Since incomplete solutions need to be compared, a simple fitness function is given by the addition of the number of "breakpoints" in the current permutation plus the number of reversals that were applied in order to obtain the current unsigned permutation. This fitness function can either be computed from scratch in linear time or updated in constant time in the size of the permutation. In this way, the expensive runtime necessary to compute the fitness function applied in the approaches in [9], [10], [11] is avoided. In order to compare the computed sorting solutions, Christie's 1.5-approximation algorithm was adequately implemented fixing some cases that were not considered in [7]. Although Berman's et al algorithm has a better approximation ratio, its implementation is not of practical interest, as previously mentioned, because it requires a great deal of effort without producing a competitive runtime mechanism of control. Experiments showed that on average the results obtained though the proposed GA approach are similar to the ones obtained through the 1.5-approximation algorithm. For all size of permutations checked, cases for which the GA approach works better and worse than the 1.5-approximation algorithm were found, but in average the computed sorting solutions are in favor of the GA approach.

The paper is organized in the following sections: in Section II, the necessary notations and notions as well as the 1.5-approximation algorithm implemented as control are given; in Section III, the proposed GA approach is presented; in Section IV, experiments and results are given; in Section V, the method and the results are discussed and; finally concluding remarks are discussed in Section VI.

II. Background

A. Terminology

All definitions and terminology presented below, were introduced by Bafna and Pevzner in their seminal paper [2].

Given a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ in the symmetry group S_n, we extend its definition by adding initial and final pivots $\pi_0 = 0$ and $\pi_{n+1} = n + 1$. A reversal $\rho(i, j)$ of an interval $[i, j]$, for $1 \leq i \leq j \leq n$, transforms the extended permutation π into $\pi' = (\pi_0, \ldots, \pi_{i-1}, \pi_{\mathbf{j}}, \ldots, \pi_{\mathbf{i}}, \ldots, \pi_{n+1})$. For example, consider the permutation

$$\pi = (0, 3, \mathbf{1}, \mathbf{5}, \mathbf{2}, 4, 6)$$

The reversal $\rho(2, 4)$ transforms π into

$$\pi' = (0, 3, \mathbf{2}, \mathbf{5}, \mathbf{1}, 4, 6)$$

Note that the reversal reverts the interval $[2, 4]$ of π.

Given two permutations π and σ, the *reversal distance problem* is the problem of finding a shortest sequence of reversals needed to transform π into σ. The *reversal distance* between π and σ is the minimum number of reversals required to transform π into σ.

By simple algebraic properties of symmetry groups, the reversal distance between π and σ is equal to the reversal distance between $\sigma^{-1}\pi$ and the identity permutations, that is denoted as id. In fact, notice that if $\rho_1 \ldots \rho_k$, is a sequence of (reversal) permutations, then it holds that $\pi\rho_1 \ldots \rho_k = \sigma$, if and only if $(\sigma^{-1}\pi)\rho_1 \ldots \rho_k = \sigma^{-1}\sigma = id$. Thus, the problem of sorting by reversals corresponds to find the reversal distance between a permutation π and the identity permutation id, that is denoted as $d(\pi)$.

Let $i \sim j$ denote the property $|i - j| = 1$. Given two consecutive elements π_i and π_j of π, for $0 < i < n + 1$ and either $j = i - 1$ or $j = i + 1$,

- they are said to be *adjacent* if $\pi_i \sim \pi_j$ and
- they are said to form a *breakpoint* if $\pi_i \nsim \pi_j$.

Observe that the identity permutation is the unique permutation without breakpoints. The number of breakpoints in π is denoted by $b(\pi)$.

Let ρ be a reversal that transforms π into π', then $b(\pi) - b(\pi') \in \{-2, -1, 0, 1, 2\}$.

Given a permutation π, one defines a *cycle graph* (also called as *breakpoint graph*) , $G(\pi)$ as a undirected edge-colored graph derived from the adjacency and breakpoint relations in π with $n+2$ vertices labeled by $0, 1, \ldots, n, n+1$. Two vertices i and j are joined by a *black edge* if (i, j) is a breakpoint of π. Two vertices i and j are joined by a *gray edge* if $i \sim j$ and i, j are not consecutive in π. An example of a cycle graph is shown in Fig.1.

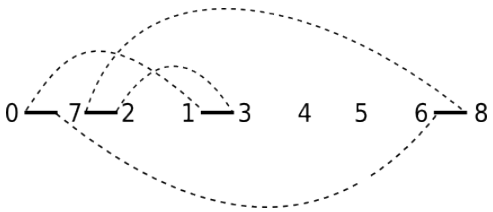

Fig. 1. Cycle graph $G(\pi)$ for the permutation $\pi = (7, 2, 1, 3, 4, 5, 6)$

Note that for all permutations π, $G(\pi)$ can be completely decomposed into disjoint cycles of alternated colored edges, since each node has an equal number of black and gray incident edges. However, there are probably many different cycle decompositions of $G(\pi)$ of alternated colored edges. For simplicity, cycles of alternating colored edges will be called either alternating cycles or simply cycles.

B. 1.5-Approximation algorithm for sorting by reversals

The development of the 1.5-approximation algorithm requires another graph, called *reversal graph*, that is used in order to find the sequence of sorting reversals. Given a permutation π, and a particular cycle decomposition *C* of the cycle graph, the reversal graph *R(C)* is constructed correspondingly to the cycle decomposition *C* following the method presented in [7].

Each vertex i of the reversal graph represents a possible reversal embedded in the gray edge $(i, i+1)$ of the cycle graph. A vertex of the reversal graph is colored either blue or red. The color of a vertex is blue if applying the associated reversal does not eliminate breakpoints, otherwise it is colored red. Two vertices are joined by an edge, if the grey edges that they represent in the cycle graph are *interleaved*. Two vertices are interleaved if the positions of the elements of the grey edges that they represent are interleaved; i.e., given the positions (a, b) and (c, d) of two gray edges, they are interleaved if $a < c < b < d$. Also, two vertices are interleaved if the rightmost positions of the black edges that belong to the gray edges that they represent are interleaved; i.e., given the rightmost positions (a, b) and (c, d) of black edges that belongs to two grey edges, they are interleaved if $a < c < b < d$.

During the implementation of Christie's algorithm, it was found a counterexample to one of the most fundamental lemmas (Lemma 4.1), that should describe how the reversal graph can be used in order to update in a straightforward manner its structure after applying reversals corresponding to its vertices. The properties presented in this lemma are the basic ones that guarantee the soundness of the approximation algorithm. Lemma 4.1 as given in [7] is presented below.

Lemma 4.1 ([7]): Let *u* be a vertex of *R(C)* that arises from a cycle of *C*. Then $R_u(C)$, that is the reversal graph obtained after applying the reversion corresponding to vertex u, can be derived directly from *R(C)* by making the following changes to *R(C)*:

- *(i)* The colour of a vertex *v* is flipped if and only if $\{u,v\}$ is an edge in R(C).
- *(ii)* For all pairs of vertices *v* and *w* in *R(C)*, such that $\{u,v\}$ and $\{u,w\}$ are edges in *R(C)*, then $\{v,w\}$ is and edge in $R_u(C)$ if and only if it is not an edge in *R(C)*.
- *(iii)* If *u* is a red vertex then it becomes an isolated blue vertex, and otherwise it is unchanged.

As counterexample for the well-functioning of the updating proposed in this lemma, consider the permutation $\pi = (7, 2, 1, 3, 4, 5, 6)$ whose cycle graph was shown in Fig.1 and whose reversal graph is shown in Fig.2.

Let the vertex 0 of the reversal graph, that corresponds to the revesal $\rho(1, 3)$, be the vertex u of the Lemma 4.1, then after applying part (ii) we obtain the resulting reversal graph shown in Fig.3

The cycle graph after applying the reversal represented by the vertex 0 is shown in Fig.4.

The resulting reversal graph should correspond to the resulting cycle graph, but it can be observed that in the resulting

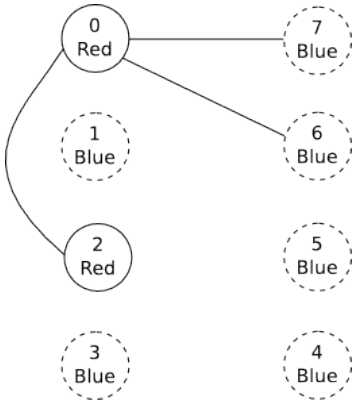

Fig. 2. Reversal graph of the counterexample $\pi = (7, 2, 1, 3, 4, 5, 6)$

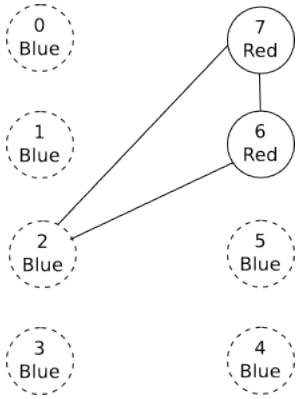

Fig. 3. The resulting reversal graph

reversal graph there is an edge between the vertices 7 and 9. And this relationship between these vertices does not exist since it does not correspond to the resulting reversal cycle graph. Thus, Lemma 4.1 fails to determine how reversal graphs should be updated after applying reversals, which is fundamental for the soundness of the approximation algorithm. In general, this kind of counterexample is possible if and only if two grey edges share the same vertex and the same black edge as was the case for vertex 7 in Fig.4.

Among other necessary modifications, the necessary adjusts were made in item (ii) of the Lemma 4.1. obtaining the following sound lemma.

Lemma 4.1 (modified): Let *u* be a vertex of *R(C)* that arises from a cycle of *C*. Then $R_u(C)$ can be derived from *R(C)* by making the following changes to *R(C)*:

- *(i)* The colour of a vertex *v* is flipped if and only if $\{u,v\}$ is an edge in $R(C)$.
- *(ii)* For all pairs of vertices *v* and *w* in *R(C)*, such that $\{u,v\}$ and $\{u,w\}$ are edges in *R(C)*, then $\{v,w\}$ is and edge in $R_u(C)$ if and only if it is not an edge in *R(C)* ***and the edges that represent the vertices*** v ***and*** w***, do not share exactly one vertex and one black edge in the cycle graph***.
- *(iii)* If *u* is a red vertex, then it becomes an isolated blue vertex, and otherwise it is unchanged.

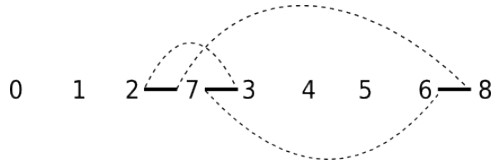

Fig. 4. Cycle graph after applying the reversal represented by vertex 0

Applying the modified version of the Lemma 4.1, the resulting reversal graph corresponds to the resulting cycle graph, in general. For the counterexample, the correct resulting reversal graph is shown in Fig.5.

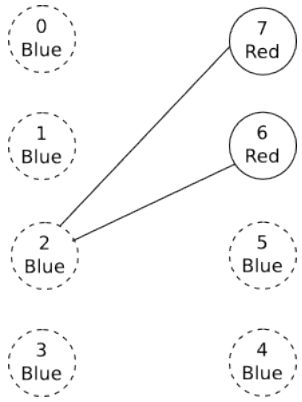

Fig. 5. The correct resulting reversal graph.

III. The GA approach for sorting by reversal unsigned permutations

Our method uses a modified version of the standard genetic algorithm. The search space consists only of reversals that eliminate 0, 1 or 2 breakpoints. Each individual of the population initially has size 0, after each generation the size of each individual of the population is increased by one. In this first part of the algorithm, it is only applied the modified crossover operator, until we find a valid solution that totally eliminates all breakpoints, at this point we only have a population with individuals of the same size. Once an initial solution is found, the mutation operator is applied to the population, decreasing the size of each individual, in order to improve the solution initially found.

There are three important operators in the modified version of the standard genetic algorithm, adapted to the problem of sorting by reversals: the increment operator, the modified crossover, the modified mutation, that are described below.

The increment operator is the responsible for increasing, in every generation, each individual in the population with a new gene. This new gene is a selected reversal that eliminates 0, 1 or 2 breakpoints.

The modified crossover chooses the best individuals of the population and duplicate them for replacing individuals with worse fitness. This is done with the aim of providing more opportunities to the individuals that represent good solutions.

The modified mutation is applied after finding a valid solution, and it is intended to improve the solution already found. After applying this operator, the length of the individuals is reduced and the population remains with individuals of different size.

The selection operator sorts the individuals of the population by their fitness value, which is the sum of the number of breakpoints and the length of the solution. This can be either computed in time $O(n)$, where n is the size of the permutation, or updated in constant time, for each individual.

The pseudo-code of our proposed genetic algorithm is shown in Algorithm 1.

Algorithm 1: Modified Genetic Algorithm

```
Input: A permutation π
Output: A sequence of reversals to sort permutation π
generate initial population;
evaluate fitness of initial population;
for i = 2 to number of generations do
    selection;
    save the best solution if found;
    crossover;
    if valid solution found then
        mutation;
    increment new genes;
    evaluate fitness of current population;
```

We consider *n* as the increased size of the initial permutation. We fixed the initial population size as $n \log n$, each individual contains initially only a reversal that eliminates 0, 1 or 2 breakpoints. The reversals are taken from the breakpoint graph of the initial permutation.

The algorithm used to sort the population, in the *selection*, is the quicksort that is well-known to take runtime in $\Theta(n\,(log\,n)^2)$ since we have to order *n log n* elements.

In each generation the increment of new genes takes runtime in $\Theta(n^2\,log\,n)$, the crossover takes runtime in $O(n^2\,log\,n)$ and the mutation takes time in $O(n^2\,log\,n)$ as well.

The genetic algorithm finishes after *n* generations, then the overall time complexity is $O(n^3 log n)$.

IV. Experiments and Results

In order to compare properly the 1.5-approximation algorithm with the proposed GA approach, experiments were performed with $(n \log n)$ permutations generated randomly with size $i = 10, 20, ...150$. Both algorithms were implemented in C language and executed in OS X platforms with Intel core I5, I7 processors and other similar platforms.

For each size *i* of permutations, 100 permutations were randomly generated. Then, it was calculated the average over these 100 permutations for both algorithms. We also calculated the best result for the genetic algorithm and the worst result against the genetic algorithm.The results of this experiment are shown in Table I.

TABLE I
COMPARISON OF THE 1.5-APPROXIMATION AND THE GA APPROACH

n	Size of pop.	Avg. 1.5-approx.	Avg. GA	Best Result (1.5-approx. vs GA)	Worst Result (1.5-approx. vs GA)
10	33	5.95	5.87	3 (9 vs 6)	0 (5 vs 5)
20	86	13.46	13.41	2(15 vs 13)	-1(14 vs 15)
30	147	21.59	21.81	4(25 vs 21)	-2(18 vs 20)
40	212	30.31	30.71	5(35 vs 30)	-1(30 vs 31)
50	282	39.36	39.99	3(41 vs 38)	-2(38 vs 40)
60	354	48.32	48.94	6(52 vs 46)	-2(48 vs 50)
70	429	57.13	58.17	4(59 vs 55)	-3(54 vs 57)
80	505	66.8	67.92	6(71 vs 65)	-3(64 vs 67)
90	584	75.94	77.11	6(80 vs 74)	-3(74 vs 77)
100	664	85.44	86.5	6(87 vs 81)	-3(80 vs 83)
110	745	94.67	96.02	9(99 vs 90)	-4(90 vs 94)
120	828	104.03	105.56	7(106 vs 99)	-4(94 vs 98)
130	912	113.77	115.13	7(118 vs 111)	-5(108 vs 113)
140	998	123.59	124.65	8(129 vs 121)	-5(116 vs 121)
150	1084	132.88	134.34	7(139 vs 132)	-4(129 vs 133)

V. DISCUSSION

Due to the high combinatorial and computational complexity of the problem of sorting by reversals unsigned permutations, it is very difficult to adapt genetic algorithms to this problem, so we had found it necessary to modify the standard model of genetic algorithms profiting in this way of a population given by partial solutions of the problem for which a simple fitness function was computed from the notion of breakpoints and the number of applied reversals. In other previous proposals such as the ones presented in [9] and [10], this was made through a mapping of the problem of sorting by reversals for unsigned permutations to corresponding signed permutations among the set of 2^n possible signed permutations. Although this scenario is more suitable in order to apply standard genetic algorithm approaches, especially because crossover and mutation operators arise in a natural manner, these approaches are more runtime expensive than the proposed here, because the complexity involved in updating the fitness function.

Since we only explore a valid and restricted search space, that is the one given by reversals that remove only 0,1, or 2 breakpoints, our algorithm often converges to premature solutions. Although it computes solutions where all breakpoints are eliminated, they are not necessarily optimal solutions. This problem was solved after incorporating the mutation operator to improve solutions already found.

From the experiments that were performed we can see that our proposed GA approach computes similar results on average to the ones obtained by application of the 1.5-approximation algorithm. It is worth mentioning that in the majority of the cases, the best solution found by our GA proposal is better than the worst solution found.

VI. CONCLUSION

Sorting permutations by reversals is an important problem in bio-informatics to help understanding the evolutionary relationship between different organisms. In this paper we proposed a new modified genetic algorithm for sorting by reversals unsigned permutations, that is well-known to be an NP-hard problem. In order to control the computed solutions, it was necessary to implement an 1.5-approximation algorithm that was guaranteeing to achieve properly its 1.5-approximation ratio after fixing some fundamental imprecisions in its original conception. Experimental results showed that our GA method produces similar results to the ones given by the1.5-approximation algorithm, and that our modified GA approach is a promising methodology for solving the problem. A distinguished feature of the proposed approach is that it uses a simplified fitness function that can be linearly computed and updated in constant time, in contrast to the complexity of the fitness function applied in previous GA approaches to resolve this problem, that were based on computing exact sorting solutions for associated signed permutations.

Several questions arise when one is dealing with permutations. The combinatorial complexity of permutations makes it unclear whether randomly generation of permutations is in fact a good choice in order to evaluate the behavior of this kind of algorithmic approaches over a representative sample of this sophisticated data structure. Additional experiments are proposed building permutations in a more controlled manner; namely, since the number of necessary reversion to order permutations of size n is less than n, another way to build interesting samples of input permutations of size n is through applications of linear sequences of randomly selected reversions applied to the identity permutation. Additional experiments will be done for samples of permutations generated in this controlled manner.

Using the number of breakpoints as a measure to compare permutations in our GA approach, was motivated from the early work of Bafna and Pevzner [2] in which the notion of breakpoint was coined and the data structure of breakpoint graph was introduced in order to present approximate solutions. From this point of view, the proposed approach is not a standard GA approach. Thus, as a future work, we will combine our current approach with other GA methods and approximation algorithms, in order to further improve the accuracy of the computed results. Good alternatives would be the 1.5-approximation algorithm since it has been checked

that it obtains solutions very close to the optimal ones for small or nearly sorted permutations. Also GA methods based on translating the problem to signed permutations are of interest when this translation is not applied in each step or generation, but instead, only when mutations and crossover operations can be used to avoid premature convergence. Also, experiments will be done with a larger number of generations, but keeping the linear order, to increase the runtime of the mutation operator.

ACKNOWLEDGMENT

The authors would like to thank the Brazilian Coordination for the Improvement of Higher Education Personnel from the Ministry of Education CAPES and the Brazilian National Counsel of Technological and Scientific Development from the Ministry of Science and Technology CNPq for grants that made possible the involvement in this interesting field of research.

REFERENCES

[1] J. Kececioglu and D. Sankoff, "Exact and approximation algorithms for the inversion distance between two chromosomes," in *Combinatorial Pattern Matching*, ser. Lecture Notes in Computer Science, A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, Eds. Springer Berlin / Heidelberg, 1993, vol. 684, pp. 87–105, 10.1007/BFb0029799. [Online]. Available: http://dx.doi.org/10.1007/BFb0029799

[2] V. Bafna and P. Pevzner, "Genome rearrangements and sorting by reversals," in *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, nov 1993, pp. 148 –157.

[3] S. Hannenhalli and P. Pevzner, "Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals," in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, ser. STOC '95. New York, NY, USA: ACM, 1995, pp. 178–189. [Online]. Available: http://doi.acm.org/10.1145/225058.225112

[4] P. Berman and S. Hannenhalli, "Fast sorting by reversal," in *CPM*, ser. Lecture Notes in Computer Science, D. S. Hirschberg and E. W. Myers, Eds., vol. 1075. Springer, 1996, pp. 168–185.

[5] D. A. Bader, B. M. E. Moret, and M. Yan, "A linear-time algorithm for computing inversion distance between signed permutations with an experimental study," in *WADS*, ser. Lecture Notes in Computer Science, F. K. H. A. Dehne, J.-R. Sack, and R. Tamassia, Eds., vol. 2125. Springer, 2001, pp. 365–376.

[6] A. Caprara, "Sorting by reversals is difficult," in *Proceedings of the first annual international conference on Computational molecular biology*, ser. RECOMB '97. New York, NY, USA: ACM, 1997, pp. 75–83. [Online]. Available: http://doi.acm.org/10.1145/267521.267531

[7] D. A. Christie, "A 3/2-approximation algorithm for sorting by reversals," in *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '98. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998, pp. 244–252. [Online]. Available: http://dl.acm.org/citation.cfm?id=314613.314711

[8] P. Berman, S. Hannenhalli, and M. Karpinski, "1.375-approximation algorithm for sorting by reversals," in *Proceedings of the 10th Annual European Symposium on Algorithms*, ser. ESA '02. London, UK, UK: Springer-Verlag, 2002, pp. 200–210. [Online]. Available: http://dl.acm.org/citation.cfm?id=647912.740832

[9] A. Auyeung and A. Abraham, "Estimating genome reversal distance by genetic algorithm," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 2, dec. 2003, pp. 1157 – 1161 Vol.2.

[10] M. Zhongxi and Z. Tao, "An improved genetic algorithm for problem of genome rearrangement," *Wuhan University Journal of Natural Sciences*, vol. 11, pp. 498–502, 2006, 10.1007/BF02836651. [Online]. Available: http://dx.doi.org/10.1007/BF02836651

[11] A. Ghaffarizadeh, K. Ahmadi, and N. Flann, "Sorting unsigned permutations by reversals using multi-objective evolutionary algorithms with variable size individuals," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, june 2011, pp. 292 –295.