2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Memetic Algorithm for Sorting Unsigned Permutations by Reversals

José Luis Soncco-Álvarez
Department of Computer Science
University of Brasilia
70910-900 Brasilia, D.F., Brazil
Email: josesoal@hotmail.com

Mauricio Ayala-Rincón
Departments of Computer Science and Mathematics
University of Brasilia
70910-900 Brasilia, D.F., Brazil
Email: ayala@unb.br

***Abstract*—Sorting by reversals unsigned permutations is a problem exhaustively studied in the fields of combinatorics of permutations and bioinformatics with crucial applications in the analysis of evolutionary distance between organisms. This problem was shown to be $\mathcal{NP}$-hard, which gave rise to the development of a series of approximation and heuristic algorithms. Among these approaches, evolutionary algorithms were also proposed, from which to the best of our knowledge a parallel version of the first proposed genetic algorithm computes the highest quality results. These solutions were not optimized for the case when the population reaches a degenerate state, that is when individuals of the population remain very similar, and the procedure still continues consuming computational resources, but without improving the individuals. In this paper, a memetic algorithm is proposed for sorting unsigned permutations by reversals, using the local search as a way to improve the fitness function image of the individuals. Also, the entropy of the population is controlled, such that, when a degenerate state is reached the population is restarted. Several experiments were performed using permutations generated from biological data as well as hundreds of randomly generated permutations of different size, from which some ones were chosen and used as benchmark permutations. Experiments have shown that the proposed memetic algorithm uses more adequately the computational resources and gives competitive results in comparison with the parallel genetic algorithm and outperforms the results of the standard genetic algorithm.**

I. Introduction

Computing the minimum number of reversals to transform one permutation into another is motivated in molecular biology by the evolutionary process of some organisms whose genomes differ in just a few number of reversals of its genes [1]. On the one hand, when the genes in the genome sequence of an organism are abstracted without any orientation in the sequence, the representation of the genome corresponds to an unsigned permutation in which each symbol is associated with a gene. On the other hand, if the orientation (positive or negative) of the genes inside the genome is considered, the genome is modeled as a signed permutation in which each gene has a positive or negative sign according to its orientation within the genome.

Calculating the minimum number of reversals to transform a permutation into another is known as the problem of sorting by reversals and is equivalent to the problem of transforming a permutation into the *identity permutation* that is the permutation sorted in increasing order for the case of unsigned permutations, and to the same permutation with positive orientation, for the signed case.

In the earliest stages of research of solutions for this problem, for the case of sorting signed permutations by reversals, approximation algorithms were proposed in [2] and in [3], before discovering that the problem is of polynomial complexity. The first polynomial time algorithm was proposed in [4] and further improved obtaining a linear time algorithm in [5], when restricted to computing only the reversal distance, and of running time $O(n^2)$ for explicitly computing the minimal sequence of necessary reversals.

For the case of sorting unsigned permutations, the problem was shown to be $\mathcal{NP}$-hard [6] and many approximation algorithms were proposed (e.g., [2], [3], [7]) until the better known approximation ratio of 1.375 was reached in [8]. The 1.375 approximation algorithm is of great theoretical interest, but its implementation of great difficulty and for this reason adaptations of the 1.5 approximation algorithm proposed in [7] was referenced as control mechanism for heuristic approaches and approximate algorithms. These approximation algorithms are based on the analysis of the graph structure of permutations in which *breakpoints* are contiguous elements in the permutations that are not relatively well-ordered. Thus, the central idea of these approximation algorithms is to eliminate breakpoints through applications of reversals, giving priority to the application of reversals that simultaneously can eliminate two breakpoints: one in each limit of the subsequence being reversed in the permutation (cf. [3]).

For the case of unsigned permutations, evolutionary algorithms were proposed. The first Genetic Algorithm (GA) was proposed by Auyeung and Abraham [9] and is based on the exploration of the space of solutions of signed permutations built from the given unsigned permutations, taking advantage of the fact that the reversal distance of signed permutations can be computed in linear time. Using as fitness function the reversal distance of signed versions of the unsigned permutation is adequate since any sequence of reversals that sorts a signed permutation also sorts the permutation without the orientations, including redundancies when reversals are applied to change the orientation of a unique gene. Subsequently, other genetic methods were proposed in [10], [11] and [12]. In particular, in the last paper a fixed implementation of the 1.5 approximation algorithm was implemented as control mechanism, that presented fairer and better results that the approximation algorithms used in the previous references. With this implementation of the 1.5 approximation algorithm a more accurate analysis of the results provided by GAs was possible making evident that these genetic techniques, as implemented, did not provided better results than the fixed 1.5 approximation algorithm. Further, a hybrid GA which combines the GA approach by Auyeung and Abrabaham with an initial phase of elimination of two breakpoints, which can be seen as an initial phase of local optimizations, was proposed in [13] and its parallel version was also reported in [14]. Both these approaches gave better results than the fixed 1.5 approximation algorithm and to the best of our knowledge the latter parallel GA provided the best known results.

In this paper it is proposed a Memetic Algorithm (MA) for the problem of sorting unsigned permutations by reversals which is also based on the GA approach proposed by Auyeung and Abraham, but without the initial phase of elimination of two breakpoints as in the hybrid approach. The main feature of the new method is that it embeds the local search in some stages, and implements a specialized control over the convergence of the population. Several experiments were performed for finding adequate parameter settings, which provided competitive results. The quality of the results was compared with other approaches computing reversal distances for hundred of randomly generated permutations and for single permutations proposed as benchmarks, that will be also used for future comparisons. Also, unsigned permutations were generated from the mitochondrial genome of some organisms taken from GeneBank and their reversal distance was calculated using the MA. In particular, despite the local searching in the proposed mechanism is much simpler to implement than the two breakpoints elimination in the hybrid approach, the MA also provides better results than Auyeung and Abraham GA as implemented in [13], but with adjusted parameters, and is slightly outperformed only by the parallel GA as implemented in [14]. It's important to stress that this parallel GA does not take advantage of the computational resources when a degenerate state is reached, unlike the proposed MA.

The rest of the paper is organized as follows: Section II presents the necessary definitions and terminology. Then, Section III introduces the pseudo-code of the proposed memetic algorithm and Section IV gives all the performed experiments. After that, Section V discusses the results of the experiments, and finally Section VI concludes and presents future work.

II. Definitions and Terminology

Most of the notations and definitions related to unsigned permutations that are given in this section were taken from [3] or [2].

The order of the genes in a genome is represented as a permutation $\pi = \pi_1, \pi_2, \ldots, \pi_n$ that is a bijection of the set $\{1...n\}$ into itself, being n the number of elements of the permutation.

A reversal $\rho^{i..j}$ over a permutation π is an operation that reverses all the elements of a permutation at positions in the interval from position i to position j.

The problem of determining the minimum number of reversals to transform a permutation into the identity permutation $\imath$, defined as $\imath(k) = k$ for all $k = 1, \ldots, n$,

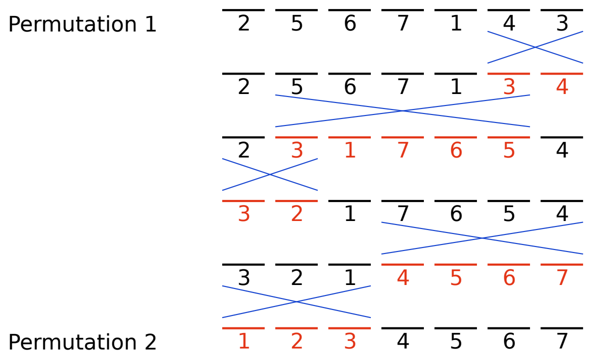

Fig. 1. Application of a sequence of reversals to transform one permutation into the identity permutation

is known as the problem of sorting permutations by reversals. In Fig. 1 it can be seen how the sequence of reversals: $(\rho^{6..7}, \rho^{2..6}, \rho^{1..2}, \rho^{4..7}, \rho^{1..3})$ sorts the permutation $2, 5, 6, 7, 1, 4, 3$.

Let $i \sim j$ denote the property $|i - j| = 1$ and extend the permutation with the initial and final pivots 0 and $n+1$. Given two consecutive elements π_i and π_j of π, for $0 < i < n+1$, such that either $j = i - 1$ or $j = i + 1$, one says that:

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

For instance, in the the permutation 1 of Fig. 1 the consecutive elements 5 and 6, 6 and 7 and 4 and 3 are adjacent and the consecutive elements 2 and 5, 7 and 1 and 1 and 4 form breakpoints.

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 it is easy to observe that $b(\pi) - b(\pi') \in \{-2, -1, 0, 1, 2\}$. We define an i-reversal as a reversal that reduces the number of breakpoints by i. Approximate algorithms give preference to the inclusion of 2-reversals in a sorting sequence (cf. [2], [3], [7]). This is also used as an heuristic in the hybrid GA introduced by the authors in [13].

In a signed permutation π its elements are either positive $(+\pi_i)$ or negative $(-\pi_i)$. A standard manner to transform π into an unsigned permutation is by replacing each positive element $+\pi_i$ by the pair $(2\pi_i - 1, 2\pi_i)$, and each negative element $-\pi_i$ by the pair $(2\pi_i, 2\pi_i - 1)$. In this way a signed permutation of length n becomes an unsigned permutation of contiguous inseparable pairs of sorted elements of length $2n$. For instance, the permutation $-2, -5, +6, -7, +1, +4, -3$ is transformed into $(4, 3), (10, 9)(11, 12), (14, 13), (1, 2), (7, 8), (6, 5)$ and the identity $+1, +2, +3, +4, +5, +6, +7$ corresponds to $(1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 12), (13, 14)$. Also, an initial and a final pivot 0 and $2n + 1$, respectively, are added. This construction or similar ones are used by running time polynomial algorithms for solving the problem of sorting signed permutations by reversals such as the ones proposed in [4] and [5].

Note that, it can be built signed permutations based on an unsigned permutation π by assigning positive or negative signs to each of its elements. This signed permutations form the search space of evolutionary algorithms such as [9].

III. Memetic Algorithm

The Memetic Algorithms (MA) are the combination of Evolutionary Algorithms (EA) with Local Search (LS) heuristics [15]. Additionally, this class of algorithms may include exact methods and approximation algorithms.

The main idea of this metaheuristic is the individual improvement of the elements of a population and also the population-cooperation to find better solutions by imitating the evolutionary process [16]. The proposed memetic algorithm is based on the standard GA approach introduced by Auyeung and Abraham in [9] in which for each unsigned permutation of length n, the search space consists of the 2^n signed permutations obtained by assigning either a positive or negative sign to each element of the unsigned permutation. The population is a subset of the 2^n signed permutations' search space. The fitness of an individual is calculated as the reversal distance of a signed permutation using Bader et al.'s linear time algorithm [5], being the desired best result the smallest reversal distance found in the population. This approach of using a subset of the search space of signed permutations for each individual as well as the fitness calculation is also used by the proposed MA. The main steps of the standard GA are sketched in Algorithm. 1.

Algorithm 1 Sorting by Reversals' **standard GA**

```
Input: An unsigned permutation $\pi$
Output: Number of reversals to sort the permutation $\pi$
1: generate initial population of signed permutations for $\pi$
2: evaluate fitness for each individual of the initial population
3: for $i = 2$ to number of generations do
4:     selection
5:     crossover
6:     mutation
7:     evaluate fitness of offspring
8:     replacement
9: end for
```

The correctness of the standard GA approach is justified by three facts:

- the problem of sorting **signed** permutations can be solved using a linear time algorithm [5];
- a sorting solution for a signed permutation is also a feasible solution for its unsigned version;
- the solution of one of the 2^n signed permutations generated from an unsigned permutation is an optimal solution for the unsigned permutation.

So there is a search space of 2^n signed permutations that makes the problem of finding the optimal sorting reversal of an unsigned much too difficult than the difficult of searching solutions for a sole signed permutation (namely, NP-hard as previously mentioned). Using just the standard GA approach gives good results, but since this approach does not consider the local searching, here local searching was included at certain stages of the GA mechanism to improve the fitness of the individuals and thus the quality of the results.

Algorithm 2 Sorting by Reversals' **MA**

```
Input: An unsigned permutation $\pi$
Output: Number of reversals to sort the permutation $\pi$
1: generate initial population of signed permutations for $\pi$
   improved with local search
2: evaluate fitness for each individual of the initial population
3: for $i = 2$ to number of generations do
4:     selection
5:     crossover
6:     mutation
7:     evaluate fitness of offspring
8:     replacement
9:     apply local search to the current population
10:    if entropy threshold is reached then
11:        restart population and improve with local search
12:    end if
13: end for
```

The proposed memetic algorithm is the combination of the standard GA mentioned before with the mechanism of local search, which is applied in following stages of the standard GA:

- Generation of the initial population.
- Restarting the population.

Also, a new stage is added after the breeding cycle, that consists in applying the local search operator over the new population.

The pseudo-code of the memetic algorithm is shown in Algorithm 2. Let n be the length of the input permutation. Then, the number of individuals of the population is fixed as $n\ log\ n$. The number of generation is set to n. The stages of the lines 1, 5, 6, 7, 8, 9 and 11 have complexity of $O(n^2\ log\ n)$. So the overall time complexity is $O(n^3\ log\ n)$.

The entropy threshold is a parameter that must be optimized to give the best results, and it was done that way as mentioned in the Sec. IV of experiments. After the entropy threshold is reached a percentage of the population is restarted and improved with local searching. This percentage is also another parameter that must be optimized.

Algorithm 3 applies local searching over a single signed permutation.

Because of the use of local searching, the converge of the population to a degenerate state must also be controlled, so that a better use of the computational resources is obtained [15]. To do it, the Shannon Entropy was used as a measure of the diversity of the population. A degenerate state happens whenever all elements of a population have high similarity.

The shannon entropy [17] is defined as follows:

$$H(S) = -\sum_i p_i\ log_2\ p_i$$

where S is a set of different elements taken from the population of the MA. Let i be an element of S, then p_i is the value obtained from dividing the number of occurrences of i in the population by the number of elements of the population.

Algorithm 3 Local Search over a signed permutation

```
Input: A signed permutation $\pi$
 1: number of iterations = 2
 2: best fitness = calculate fitness of $\pi$
 3: for $i = 1$ to number of iterations do
 4:     generate a random position $j$ in $\pi$
 5:     modify the sign of element in position $j$
 6:     fitness = calculate fitness of $\pi$
 7:     if fitness $<$ best fitness then
 8:         update new fitness in $\pi$
 9:         break
10:     end if
11: end for
```

IV. Experiments and Results

A. Setting the parameters of the memetic algorithm

Since there are many parameters to be established optimally, we have to test all the possibilities, that is, all combinations of parameters to get the best possible result (in terms of number of reversals) and therefore the best possible configuration of parameters.

For example, the crossover probability has values in the interval $[0, 1]$, and for sake of tractability one can be consider the discrete interval $\{0.1, 0.2, \ldots, 0.9\}$. But, even with this restriction we would have a great amount of combinations of parameters. So, first we have to refine this discrete interval for each parameter.

The following procedure was applied for setting the parameters:

1) All the parameters were divided into three groups
   - Group 1:
     - p_1: **Crossover** probability
     - p_2: **Mutation** probability
     - p_3: Number of points of crossover
   - Group 2:
     - p_4: Percentage of population for **selection**
     - p_5: Percentage of population to be **replaced**
   - Group 3:
     - p_6: Percentage of population on which **local searching** is applied
     - p_7: Percentage of population that will be preserved after **restarting** the population
     - p_8: Maximum **entropy** threshold
2) For each group the following was done:
   a) For each parameter in a group, a discrete interval was generated (e.g. discrete interval for crossover probability $\{0.1, 0.2, \ldots, 0.9\}$);
   b) Each discrete interval was reduced based on the results of experiments (see third column of Table I);
   c) For each group all possible permutations of parameters of their discrete **reduced** intervals were generated;
   d) The best configuration (best values, i.e. a permutation of parameters) was chosen based on the results of experiments (see fourth column of Table I).

The parameters of the groups 1 and 2 are related with the standard GA as well as with the MA, being the standard GA used in this case for the experiments. For the group 3 the MA was used since the parameters of this group are related only with this algorithm.

TABLE I. Reduced intervals and corresponding best values for each parameter

	Parameter	**Reduced Interval**	**Best Value**
p_1	Crossover probability	$\{0.96, 0.98\}$	0.98
p_2	Mutation probability	{0.01,0.02,0.03,0.04,0.05}	0.01
p_3	Num. crossover points	$\{1, 2, 3, 4, 5\}$	1
p_4	% for selection	$\{0, 94, 0.96, 0.98\}$	0.96
p_5	% for replacement	$\{0.6, 0.65, 0.7\}$	0.6
p_6	% for local search	$\{0.94, 0.96, 0.98\}$	0.94
p_7	% of preservation	$\{0.94, 0.96, 0.98\}$	0.98
p_8	Max. entropy threshold	$\{0.1, 0.2, 0.3, 0.4\}$	0.2

The experiments pointed out in step 2.b) for reducing a discrete interval for one parameter p, were performed in the following way:

- First, a set of hundred unsigned permutations was generated for each length $i \in \{10, 50, 100, 150\}$.
- Then, a value for p was taken from its discrete interval and the standard GA (or MA in case of group 3) was executed 10 times for each unsigned permutation of length i. This was done for each value of p in its discrete interval.
- After that, the average of the results of these 10 executions was calculated. This represents the result (in terms of the number of reversals) for one unsigned permutation of length i for a given value of the parameter p;
- Next, the average of the results of hundred permutations of length i for a given value of p was calculated. This represents the result for a set of hundred permutation of length i for a given value of p.
- Finally, the results of each set of hundred permutations of length i with the same value of p were averaged. The values of p that correspond to the best averages represent the reduced discrete interval.

It is important to stress that the remaining parameters that were not varying were fixed with an estimate value.

The experiments in step 2.d) to choose the best configuration of parameters, were performed in the following way:

- First, a set of hundred unsigned permutations was generated for each length $i \in \{10, 50, 100, 150\}$.
- Then, all possible permutations of parameters from the reduced discrete intervals was generated.
- Next, for each unsigned permutation of length i with a given permutation of parameters, the standard GA (or MA in case of group 3) was executed 10 times.
- The average of the results of these 10 executions was calculated. This represents the result (in terms

TABLE II. AVERAGE OF NUMBER OF REVERSALS FOR HUNDRED UNSIGNED PERMUTATIONS

Perm. Length	**AA (from [13])**	**PGA (from [14])**	**GA**	**MA**	GA - MA
10	5.84	5.81	5.848	$5,762$	0.086
20	13.69	12.94	13.176	13.042	0.134
30	21.88	$20,589$	20.965	20.735	0.23
40	30.27	28.254	28.768	28.426	0.342
50	39.64	36.291	37.14	36.647	0.493
60	48.45	44.633	45.422	44.748	0.674
70	57.56	52.949	53.713	53.016	0.697
80	66.66	60.887	62.467	61.662	0.805
90	75.86	69.555	71.188	70.359	0.829
100	85.93	78.096	80.046	79.221	0.825
110	94.03	86.702	88.155	87.332	0.823
120	104.37	95.258	97.293	96.508	0.785
130	113.38	104.582	105.872	105.098	0.774
140	123.15	113.539	114.869	114.341	0.528
150	132.76	122.671	123.554	123.116	0.438

of number of reversals) for one unsigned permutation for a given permutation of parameters.

- The average of the results of hundred unsigned permutations for each length i for a given permutation of parameters represents the result for this set of hundred unsigned permutations of length i and the given configuration of parameters.
- Finally, the results of each set of hundred unsigned permutations of length i with the same permutations of parameters were averaged. The configuration of parameters (permutations of parameters) that has the best average is chosen as the best setting of parameters.

B. Experiments with hundred unsigned permutations for comparison with other algorithms

In order to carry out an accurate comparison of the MA with the standard GA, several experiments using hundred unsigned permutations were performed in the following way:

- First, hundred unsigned permutations were generated for each length of $10, 20, \ldots, 150$
- For each permutation the MA and the standard GA were executed 10 times with the same set of random seeds.
- The average of the results of these 10 executions for both algorithms were calculated. These averages represent the result (in terms of number of reversals) for each unsigned permutation of a given length.
- Finally, for each length the average of the results of the hundred permutation of the same length was calculated.

The results of these experiments are shown in the Table II. The fourth column represents the difference between the standard GA and the MA.

C. Experiments with specific unsigned permutations for comparison with other algorithms

To compare the results of different evolutionary algorithms fairly it is desirable to use benchmarks. Previous EA approaches for sorting permutations by reversals reported just experiments with a benchmark permutation of length 36 in [2] [10] that was generated by considering the mitochondrial genomes of mammals and the flatworm *Ascaris suum* maintaining only genes in the intersection. Besides, we generated other permutations based on the mitochodrial genomes of *Homo sapiens* (37 genes), *Drosophila melanogaster* (fruit fly, 37 genes), *Crocodylus mindorensis* (Philippine crocodile, 35 genes), *Sibon nebulatus* (clouded snake, 37 genes), *Caretta caretta* (loggerhead sea turtle, 36 genes). All these genomes were taken from *GeneBank* a genetic sequence database of the National Center for Biotechnology Information (NCBI). For generating a permutation between a pair o genomes only genes in the intersection were considered, the overall number of deleted genes for each pair is shown in Table III. Additionally, in table IV the results for the MA using these biologically-based permutations are shown, where each cell represents the result for the permutation generated from the organisms in the respective row and column.

TABLE III. NUMBER OF DELETED GENES FOR BUILDING A PERMUTATION BETWEEN TWO ORGANISMS

	Hom.	*Dro.*	*Cro.*	*Sib.*	*Car.*
Homo sapiens	0	4	4	4	1
Drosophila melanogaster	-	0	4	4	3
Crocodylus mindorensis	-	-	0	4	3
Sibon nebulatus	-	-	-	0	3
Caretta caretta	-	-	-	-	0

TABLE IV. MA' RESULTS USING BIOLOGICALLY-BASED PERMUTATIONS

	Hom.	*Dro.*	*Cro.*	*Sib.*	*Car.*
Homo sapiens	0	16	3	2	0
Drosophila melanogaster	-	0	15	17	16
Crocodylus mindorensis	-	-	0	5	3
Sibon nebulatus	-	-	-	0	2
Caretta caretta	-	-	-	-	0

Since these biologically-based permutations are relatively short, additionally random permutations of different lengths were generated and proposed as benchmarks.

For all these specific permutations the experiments were performed in the following way:

- First, two random permutations were generated for each length $10, 50, 100, 150$. These permutations are named `iRPLn` where $i = 1, 2$ for the first or second permutation and n for its length, thus `2RPL100` is the second permutation of length 100.

- For each permutation the memetic algorithm and the other related algorithms were executed 30 times with the same set of random seeds.
- The best, the worst, the average and the median results of these 30 executions were calculated for each unsigned permutation and each different algorithm.

The results of these experiments are shown in the Table V.

V. Discussion

In the known related works (i.e., [9], [11], [10], [12], [13], and [14]) in which evolutionary algorithms were used to solve the reversal distance problem, any specific methodology applied to establish the parameter settings was reported and it seems to be that just empirical observations without exhaustive experiments support the selected settings. In Section IV a procedure to establish the parameter settings guided by the results of exhaustive experiments is proposed. Because of this, the confidence on the selection of adequate parameter settings is higher than in previous approaches.

From the experiments using hundred permutations one can see that the MA overcomes the results of the standard GA (see Table II). Also, a comparison with related work was done as presented in Fig. 2 for permutations of length 150, 140, 100, 90, 50, 40, 20 and 10. From this figure it can be seen that the proposed MA outperforms the results of all other algorithms except those computed by the parallel version of the standard GA, although they both compute very similar results that do not differ so much. And it is relevant to mention that the resources used by the parallel GA are higher than the ones of the MA since the former works in parallel with a population that is the same of the MA ($n\ log(n)$) times the number of processors.

From the results using biologically-based permutations several interpretations could arise. For instance, the mitochondrial genome of the *Homo sapiens* is far from the *Drosophila melanogaster* by 16 reversals and 4 deletions. Despite only a specialist could propose a consistent interpretation based on this data, at least can be concluded that reversals and deletions represent in this case an evolutionary divergence between both organisms.

From experiments using specific permutations proposed as benchmarks, it can be observed that:

- With respect to the **best** values, the MA and GA have almost the same results.
- With respect to the **worst** values the MA has the best results.
- With respect to the **mean** values the MA overcomes the results of the GA.
- With respect to the **median** values the MA has almost all the best results.

In all cases the MA has at least the same values as the GA, otherwise the MA always overcome the GA.

The Wilcoxon Signed Rank Test of the results in Table V are presented in Table VI in which the standard GA and the MA are compared for all the benchmark permutations.

TABLE VI. Statistical Comparison between the standard GA and the MA using the Wilcoxon Signed Rank Test

Benchmark	**p**	**h**	**zval**	**signedrank**
1RPL10	1	**0**	$-$	0
2RPL10	1	**0**	$-$	0
1RPL50	0.0073	**1**	-2.6833	25.5
2RPL50	0.0122	**1**	-2.507	20.5
1RPL100	$2.0588e-04$	**1**	-3.7117	18
2RPL100	0.0048	**1**	-2.8197	53.5
1RPL150	0.0303	**1**	-2.1664	68
2RPL150	0.3931	**0**	-0.8541	131.5

From this results it can be concluded that in the majority of the cases, that is when h = 1, both algorithms have different behavior at 5% of significance level. In the other cases (for permutations 1RPL10, 2RPL10, and 2RPL150), when h=0, the null hypothesis that there is no difference between both algorithms can not be rejected. This statistical test was performed using the statistical toolbox provided by Matlab.

VI. Conclusions

A memetic algorithm for sorting unsigned permutations by reversals was proposed. The algorithm is based on the approach firstly presented by Auyeung and Abraham, that consists in generating an initial population of signed permutations built from the unsigned permutation by assigning either positive or negative signs to its elements. Additionally, since solutions for signed versions of the unsigned permutation are also feasible solutions of this permutation and the reversal distance of signed permutations can be computed in linear time, the fitness of the unsigned permutation is efficiently computed as the minimum reversal distance of its signed versions. In addition to these characteristics, other relevant feature of the proposed memetic algorithm is the application of local search in the next cases:

- when generating the initial population of signed permutations;
- when restarting the population after a entropy threshold is reached;
- and, as a new stage, after the breeding cycle.

The local search consists in modifying the sign of an element of a signed permutation at a random position and verifying if the fitness is improved. Since the individuals of the population are improved by local search, it was necessary to control the converge of the population to a degenerate state and to do so, it was used the Shannon entropy. The population is restarted after it reaches a degenerate state, that is when the individuals are very similar, and thus, using adequately the computational resources with the aim of improving the results.

Exhaustive experiments were performed:

- to determine a parameter setting more adequate than in previous works;
- for comparing the standard GA solution and the MA using sets of hundred randomly generated permutations of different lengths;
- and, for comparing the standard GA and the MA using single permutations proposed as benchmarks.

TABLE V. Best, Worst, Mean and Median Number of Reversals for Single Unsigned Permutations

	Standard Genetic Algorithm				Memetic Algorithm			
Benchmark	Best	Worst	Mean	Median	Best	Worst	Mean	Median
1RPL10	5	5	5.000	5	5	5	5.000	5
2RPL10	7	8	7.033	7	7	7	7.000	7
1RPL50	37	39	37.567	38	37	38	37.167	37
2RPL50	36	39	37.333	37	36	38	36.733	37
1RPL100	79	84	81.867	82	79	82	80.700	81
2RPL100	77	81	79.600	80	77	80	78.767	79
1RPL150	122	126	124.200	124	121	125	123.533	123
2RPL150	123	129	125.767	126	122	128	125.500	126

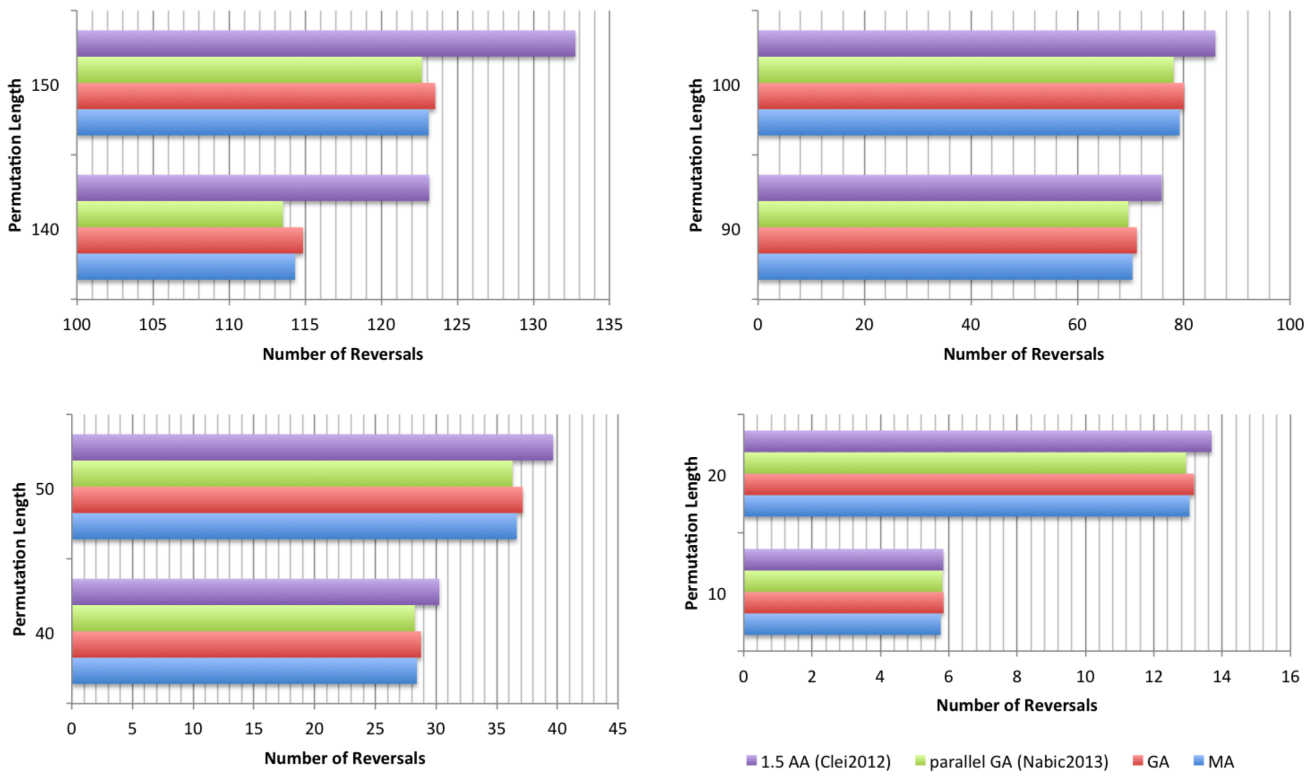

Fig. 2. Comparison with related work for permutations of lengths 150,140,100,90,50,40,20,10

Also, promising experiments were performed with biological data, namely mitochondrial DNA of different organisms, in order to point out the real capabilities of the method to give eventual support to the analysis of evolutionary diversity.

Results of the experiments with randomly generated sets of hundred permutations and with single permutations shown that the MA outperforms the standard GA, but not yet a parallel version of the standard GA which uses a population higher that the MA (number of processors times the size of the population used by the MA, that is $24\ O(n\ log(n))$).

As a future work we plan to:

- combine the MA with the heuristic of elimination of 2-breakpoints as used in the hybrid GA introduced in [13];
- combine the MA with the 1.5 approximation algorithm;
- and, to include in the restarting population stage the generation of opposite permutations, that is a permutation with all their signs changed, for finding better solutions in another part of the search space.
- to adapt the method to provide consistent support in the construction of phylogenetic trees, that would be based on the reversal distance as well as on other operations over permutations built from the mitochondrial DNA of organisms.

Appendix

A. List of Benchmark Permutations

1RPL10 = {3, 2, 6, 5, 8, 1, 4, 9, 7, 10}

2RPL10 = {7, 1, 9, 2, 6, 8, 4, 5, 10, 3}

1RPL50 = {19, 37, 48, 17, 42, 15, 31, 27, 7, 50, 2, 43, 12, 36, 32, 35, 13, 29, 47, 18, 23, 10, 6, 34, 46, 38, 44, 33, 3, 26, 21, 39, 40, 41, 5, 14, 16, 20, 8, 22, 49, 9, 11, 4, 45, 25, 28, 30, 1, 24}

2RPL50 = {44, 48, 10, 32, 3, 25, 49, 27, 37, 7, 47, 5, 13, 17, 39, 16, 14, 46, 6, 8, 36, 42, 18, 33, 40, 26, 12, 23, 9, 19, 24, 28, 1, 4, 50, 35, 30, 34, 15, 31, 11, 41, 22, 2, 21, 20, 43, 38, 29, 45}

1RPL100 = {57, 37, 38, 36, 34, 6, 42, 44, 40, 14, 82, 99, 3, 20, 16, 39, 9, 17, 86, 23, 54, 45, 92, 26, 35, 59, 22, 11, 70, 31, 73, 51, 46, 68, 33, 80, 53, 100, 66, 47, 43, 71, 8, 10, 97, 13, 96, 61, 94, 74, 27, 4, 24, 67, 25, 19, 48, 15, 65, 95, 98, 93, 7, 88, 12, 85, 91, 81, 5, 58, 79, 56, 83, 77, 18, 78, 76, 55, 87, 1, 49, 21, 75, 60, 84, 63, 29, 69, 50, 2, 64, 62, 72, 30, 32, 41, 28, 90, 52, 89}

2RPL100 = {100, 55, 61, 44, 59, 80, 31, 82, 28, 15, 33, 96, 97, 87, 73, 3, 75, 22, 40, 62, 45, 90, 99, 27, 21, 63, 52, 70, 58, 12, 13, 42, 7, 30, 17, 49, 34, 94, 4, 91, 18, 8, 5, 88, 11, 35, 66, 95, 25, 29, 71, 20, 83, 69, 57, 14, 53, 23, 74, 51, 6, 78, 48, 60, 41, 50, 64, 54, 67, 76, 32, 19, 86, 92, 89, 2, 24, 26, 85, 36, 1, 10, 16, 72, 38, 47, 81, 46, 65, 39, 43, 77, 93, 37, 84, 79, 68, 9, 98, 56}

1RPL150 = {149, 91, 114, 79, 54, 47, 42, 63, 124, 119, 72, 103, 80, 31, 95, 33, 32, 129, 24, 26, 67, 4, 60, 98, 105, 117, 59, 50, 18, 27, 65, 8, 88, 144, 125, 83, 74, 57, 82, 48, 138, 81, 131, 30, 118, 29, 109, 93, 128, 78, 120, 2, 3, 5, 10, 35, 61, 41, 102, 108, 139, 49, 20, 13, 116, 137, 73, 84, 44, 150, 70, 6, 90, 25, 1, 38, 142, 143, 7, 45, 17, 146, 46, 36, 121, 53, 122, 107, 145, 85, 134, 68, 37, 101, 77, 86, 132, 94, 106, 87, 11, 100, 43, 55, 66, 34, 40, 135, 104, 75, 136, 92, 110, 21, 113, 115, 23, 15, 89, 52, 130, 56, 141, 148, 147, 76, 96, 111, 51, 140, 12, 69, 71, 58, 9, 126, 133, 99, 16, 97, 123, 127, 62, 28, 112, 64, 14, 39, 19, 22}

2RPL150 = {124, 56, 123, 60, 18, 68, 46, 128, 28, 137, 8, 37, 84, 43, 36, 69, 41, 97, 119, 19, 14, 23, 31, 54, 35, 40, 117, 104, 94, 103, 99, 57, 79, 77, 49, 16, 126, 95, 135, 53, 1, 4, 55, 38, 122, 125, 39, 58, 116, 73, 44, 93, 80, 88, 107, 51, 136, 146, 120, 142, 143, 78, 25, 83, 129, 11, 139, 30, 45, 81, 17, 108, 112, 149, 7, 89, 92, 109, 132, 2, 47, 102, 63, 61, 144, 130, 150, 29, 141, 115, 33, 140, 131, 134, 15, 98, 22, 67, 106, 5, 10, 50, 105, 91, 62, 111, 133, 71, 82, 87, 24, 21, 147, 127, 20, 9, 13, 70, 65, 110, 72, 76, 113, 66, 101, 75, 52, 100, 90, 138, 6, 114, 74, 118, 148, 86, 42, 34, 27, 145, 59, 26, 48, 121, 96, 85, 32, 3, 12, 64}

Acknowledgment

This work was funded by CNPq universal and FAPDF PRONEX grants. Authors are resp. fully and partially supported by CAPES and CNPq.

References

[1] V. Bafna and P. A. Pevzner, "Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of x chromosome," *Mol. Biol. and Evol*, vol. 12, pp. 239–246, 1995.

[2] J. Kececioglu and D. Sankoff, "Exact and approximation algorithms for the inversion distance between two chromosomes," in *Combinatorial Pattern Matching*, ser. LNCS. Springer, 1993, vol. 684, pp. 87–105.

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

[4] S. Hannenhalli and P. Pevzner, "Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals," in *Proc. of the 27th annual ACM symposium on Theory of computing*, ser. STOC'95, 1995, pp. 178–189.

[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. LNCS, 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. ACM, 1997, pp. 75–83.

[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. Society for Industrial and Applied Mathematics, 1998, pp. 244–252.

[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. Springer, 2002, pp. 200–210.

[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, 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.

[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*, 2011, pp. 292 –295.

[12] J. L. Soncco-Álvarez and M. Ayala-Rincón, "A genetic approach with a simple fitness function for sorting unsigned permutations by reversals," in *Computing Congress (CCC), 7th Colombian*. IEEE Xpress, 2012, pp. 1–6.

[13] ——, "Sorting permutations by reversals through a hybrid genetic algorithm based on breakpoint elimination and exact solutions for signed permutations," *Electr. Notes Theor. Comput. Sci.*, vol. 292, pp. 119–133, 2013.

[14] J. Soncco-Alvarez, G. Marchesan Almeida, J. Becker, and M. Ayala-Rincon, "Parallelization and virtualization of genetic algorithms for sorting permutations by reversals," in *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*, 2013, pp. 29–35.

[15] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms," 1989.

[16] ——, "A gentle introduction to memetic algorithms," in *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003, pp. 105–144.

[17] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July, October 1948. [Online]. Available: http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf