Geometric Nelder-Mead Algorithm for the Permutation Representation

Alberto Moraglio and Julian Togelius

***Abstract*— The Nelder-Mead Algorithm (NMA) is an almost half-century old method for numerical optimization, and it is a close relative of Particle Swarm Optimization (PSO) and Differential Evolution (DE). In recent work, PSO, DE and NMA have been generalized using a formal geometric framework that treats solution representations in a uniform way. These formal algorithms can be used as templates to derive rigorously specific PSO, DE and NMA for both continuous and combinatorial spaces retaining the same geometric interpretation of the search dynamics of the original algorithms across representations. In previous work, a geometric NMA was derived for the binary string representation. In this paper, we advance this line of research and derive formally a specific NMA for the permutation representation. The result is a Nelder-Mead Algorithm searching the space of permutations by acting directly on this representation. We present initial experimental results for the new algorithm on the Traveling Salesman Problem. The peculiar geometry of the permutation space seems to affect the performance of the geometric NMA that does not perform as well as the NMA for the binary string representation. We present a discussion about the nature of permutation spaces that seeks to explain this phenomenon. Further study is required to understand if this is a fundamental limitation of the application of the geometric NMA to permutation spaces.**

I. Introduction

The Nelder-Mead Algorithm published by Nelder and Mead in 1965 [12], also known as the simplex method, is a numerical optimization method which is, despite its age, the method of choice for many practitioners. Contrasted with the majority of classic methods for numerical optimization, it only uses the values of the objective function without any derivative information. The search done by NMA is based on geometric operations (reflection, expansion, contraction and shrinking) on a current set of points, seen as the corners of a n-dimensional polygon (a simplex), to determine what points in space to evaluate next. The overall behaviour of the NMA expands or focuses the search adaptively on the basis of the topography of the fitness landscape.

Interestingly, the NMA can be seen as a form of (population-based) evolutionary algorithm with special selection and reproduction operators [17]. Also, there are similarities between the search operators employed by the NMA and those of DE [16] and PSO [3] that have led a number of authors to propose hybrid approaches (see for example [19] and [4]). As the original versions of DE and PSO, NMA requires the search space to be continuous and the points in space to be represented as vectors of real numbers.

There are few extensions of DE and PSO to combinatorial spaces [16] [15] [14] [2] [1] and to the space of genetic programs [13]. Some of these works recast the search in discrete spaces as continuous search via encoding the candidate solutions as vectors of real numbers and then applying the traditional search algorithms to solve these continuous problems. Other works present PSO and DE algorithms defined on combinatorial spaces acting directly on the original solution representation that, however, are only loosely related to the traditional algorithms in that the original geometric interpretation is lost in the transition from continuous to combinatorial spaces. Furthermore, in the latter approaches every time a new solution representation is considered, the search algorithm needs to be rethought and adapted to the new representation. To the authors's best knowledge, apart from very recent work of one of the authors [8], there are no generalizations of the NMA to combinatorial spaces.

The searches done by PSO, DE and NMA have natural geometric interpretations and can be understood as the motion of points in space obtained by different but related linear combinations of their current and past positions to determine their new positions. Geometric Particle Swarm Optimization (GPSO) [6], Geometric Differential Evolution (GDE) [11] and Geometric Nelder-Mead Algorithm (GNMA) [8] are recently devised formal generalizations of PSO, DE and NMA that, in principle, can be specified to any solution representation while retaining the original geometric interpretation of the dynamics of the points in space across representations. In particular, these formal algorithms can be applied to any search space endowed with a distance and associated with any solution representation to derive formally specific PSO, DE and NMA for the target space and for the target representation.

Specific GPSOs were derived for different types of continuous spaces and for the Hamming space associated with binary strings [7], for spaces associated with permutations [10] and for spaces associated with genetic programs [18]. GDE was specialized to the space of binary strings [11] and, very recently, to the space of genetic programs [9]. GNMA was specialized to the space of binary strings [8]. The derived algorithms performed satisfactorily in experimental results. This suggests that the generalization methodology employed is a promising one. In this paper, we continue this line of research and derive the Geometric Nelder-Mead Algorithm for the space of permutations. Preliminary experimental results indicate that the permutation-based GNMA does not perform as well as the GNMA for binary strings, as unlike the latter, the former is not really competitive with a genetic algorithm and a stochastic local search based on the same search space. The reason behind it seems to be related with the peculiar geometric properties of the permutation space that forces the

GNMA towards a degenerate dynamic. Further investigation is needed to elucidate if this is a fundamental limitation of the application of the GNMA to permutation spaces.

The remaining part of the paper is organized as follows. Section II reviews the original Nelder-Mead Algorithm and Section III presents its formal geometric generalization. Section IV presents specific GNMA search operators for permutations. Section V reports experiments for GNMA for permutations on the Traveling Salesman Problem and in Section VI presents an analysis and a discussion. Section VII presents the conclusions and future work.

II. Classic Nelder-Mead Algorithm

Algorithm 1 Nelder-Mead Algorithm

Input: f: the objective function to minimize
Input: $n+1$: number of points in the simplex
Input: α, ρ, γ, σ: reflection, expansion, contraction and shrink coefficients
Output: x^*: the best solution found

$S \leftarrow createPop(n+1)$
while stop criterion not met **do**
  $S \leftarrow sortPop(S, f)$
  // Center of mass: determine the center of mass of the n best points
  $m \leftarrow \frac{1}{n} \sum_{i=0,n-1} S[i]$
  // Reflection: reflect the worst point over m
  $r \leftarrow m + \alpha(m - S[n])$
  if $f(S[0]) < f(r) < f(S[n])$ **then**
    $S[n] \leftarrow r$
  else
    if $f(r) \leq f(S[0])$ **then**
      // Expansion: try to search farther in this direction
      $e \leftarrow r + \gamma(r - m)$
      if $f(e) < f(r)$ **then**
        $S[n] \leftarrow e$
      else
        $S[n] \leftarrow r$
      end if
    else
      $b \leftarrow true$
      if $f(r) \geq f(S[n-1])$ **then**
        // Contraction: a test point between r and m
        $c \leftarrow \rho r + (1-\rho)m$
        if $f(c) < f(r)$ **then**
          $S[n] \leftarrow c$
          $b \leftarrow false$
        end if
      end if
      if b = true **then**
        // Shrink towards the best solution candidate $S[0]$
        for i from n down to 1 **do**
          $S[i] \leftarrow S[0] + \sigma(S[i] - S[0])$
        end for
      end if
    end if
  end if
end while
return $S[0]$

In this section, we describe the traditional NMA [12] (see Algorithm 1). The NMA uses $n+1$ points in $\mathbf{R}^n$. These points form a type of n-dimensional polygon, a simplex, which has $n+1$ points as vertices in $\mathbf{R}^n$. For example, the simplex is a triangle in $\mathbf{R}^2$ and a tetrahedron in $\mathbf{R}^3$. The initial simplex has to be non-degenerate, i.e., the points must not lie in the same hyperplane. This allows the NMA to search in all n dimensions. The method then performs a sequence of transformations of the simplex, which preserve non-degeneracy, aimed at decreasing the function values at its vertices. At each step, the transformation is determined by computing one or more test points and comparing their function values. In Figure 1, we illustrate the NMA transformations for the two-dimensional case, where the simplex S consists of three points.

The optimization process described by Algorithm 1 starts with creating a sample of $n+1$ random points in the search space. Notice that apart from the creation of the initial simplex, all further steps are deterministic and do not involve random choices. In each loop iteration, the points in the simplex S are arranged in ascending order according to their corresponding objective values. Hence, the best solution candidate is $S[0]$ and the worst is $S[n]$. We then compute the center m of the n best points and then reflect the worst candidate solution $S[n]$ through this point, obtaining the new point r as also illustrated in Fig. 1(a). The reflection parameter α is usually set to 1. In the case that r is neither better than $S[0]$ nor as worse as $S[n]$, we directly replace $S[n]$ with it. If r is better than the best solution candidate $S[0]$, we expand the simplex further into this promising direction. As sketched in Fig. 1(b), we obtain the point e with the expansion parameter γ set to 1. We now take the best of these two points to replace $S[n]$. If r is no better than $S[n]$, the simplex is contracted by creating a point c somewhere in between r and m. In Fig. 1(c), the contraction parameter ρ was set to $1/2$. We substitute $S[n]$ with c only if c is better than r. When everything else fails, we shrink the whole simplex by moving all points (except $S[0]$) into the direction of the current optimum $S[0]$. The shrinking parameter σ normally has the value $1/2$, as is the case in the example outlined in Fig. 1(d).

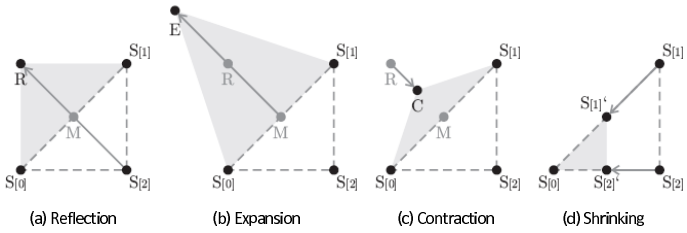

Fig. 1. One step of the NMA in R^2 (figure modified from [20])

III. Geometric Nelder-Mead Algorithm

In this section, we present how the general Geometric Nelder-Mead Algorithm [8] (Algorithm 2) was derived from the classic Nelder-Mead Algorithm (Algorithm 1). The generalization was obtained using a methodology to generalize search algorithms for continuous spaces to combinatorial spaces [11] based on the geometric framework introduced by Moraglio [5], sketched in the following.

1) Given a search algorithm defined on continuous spaces, one has to recast the definition of the search operators expressing them explicitly in terms of Euclidean distance between parents and offspring.
2) Then one has to substitute the Euclidean distance with a generic metric, obtaining a formal search algorithm generalizing the original algorithm based on the continuous space.

3) Next, one can consider a (discrete) representation and a distance associated with it (a combinatorial space) and use it in the definition of the formal search algorithm to obtain a specific instance of the algorithm for this space.
4) Finally, one can use this geometric and declarative description of the search operator to derive its operational definition in terms of manipulation of the specific underlying representation.

This methodology was used to generalize PSO, DE and NMA to any metric space, obtaining GPSO, GDE and GNMA, then to derive the specific search operators for a number of specific representations and distances. In particular for GNMA, the generalization of the classic Nelder-Mead Algorithm to general metric spaces was done by recasting the search operations described in the previous section (reflection, expansion, contraction and shrinking) as functions of the distance of the underlying search space, thereby obtaining their abstract geometric definitions, as explained below. Then, the specific GNMA for the Hamming space associated with binary strings was derived [8]. In Section IV, we derive the specific GNMA for the space of permutations with swap distance by plugging this distance in the abstract definition of the search operators.

Algorithm 2 Formal Nelder-Mead Algorithm

```
1: Input: f: the objective function to minimize
2: Input: n + 1: number of points in the simplex
3: Input: α, ρ, γ, σ: reflection, expansion, contraction and shrink coefficients
4: Output: x*: the best solution candidate found
5:
6: S ← createPop(n + 1)
7: while stop criterion not met do
8:    S ← sortPop(S, f)
9:    // Center of mass: determine the center of mass of the n best points
10:   m ← CM(S[0], S[1], ..., S[n − 1])
11:   // Reflection: reflect the worst point over m
12:   r ← ER(S[n], m) with weights (α/(1+α), 1/(1+α))
13:   if f(S[0]) < f(r) < f(S[n]) then
14:      S[n] ← r
15:   else
16:      if f(r) ≤ f(S[0]) then
17:         // Expansion: try to search farther in this direction
18:         e ← ER(m, r) with weights (1/γ, (γ−1)/γ)
19:         if f(e) < f(r) then
20:            S[n] ← e
21:         else
22:            S[n] ← r
23:         end if
24:      else
25:         b ← true
26:         if f(r) ≥ f(S[n − 1]) then
27:            // Contraction: a test point between r and m
28:            c ← CX(r, m) with weights (ρ, 1 − ρ)
29:            if f(c) < f(r) then
30:               S[n] ← c
31:               b ← false
32:            end if
33:         end if
34:         if b = true then
35:            // Shrink towards the best solution candidate S[0]
36:            for i from n down to 1 do
37:               S[i] ← CX(S[0], S[i]) with weights (1 − σ, σ)
38:            end for
39:         end if
40:      end if
41:   end if
42: end while
43: return S[0]
```

A. Geometric Generalization of the Nelder-Mead Algorithm

Using the notion of convex combination CX, extension ray ER and center of mass CM we can generalize all search operators of the classical Nelder-Mead Algorithm from the Euclidean case to generic metric spaces because, as we will see in the following section, these are geometric elements well-defined on any metric space.

The graphical description of the search operations of NMA (Fig. 1) leads directly to their geometric interpretation in terms of convex combination and extension ray, as follows. The reflection of the worst point $S[n]$ over M can be seen as picking a point beyond M on the extension ray originating in $S[n]$ and passing through M. The expansion operation can be seen as picking a point beyond R on the extension ray originating in M and passing through R. The contraction operation can be seen as picking a point in the segment between R and M. The shrink of all points $S[i]$ towards the best in the population $S[0]$ can be seen as replacing each point $S[i]$ with a point in the segment between $S[i]$ and $S[0]$.

In the following, we rewrite the algebraic definitions of the search operations of NMA to determine the weights of the corresponding convex combination or extension ray combination.

The definition of the reflection operation is $r = m + \alpha(m - S[n])$ (see Algorithm 1, line 12) and it can be rewritten as $m = \frac{\alpha}{1+\alpha}S[n] + \frac{1}{1+\alpha}r$. Since the coefficients of $S[n]$ and r are positive and sum up to 1 (for $\alpha \in [0, 1]$), this equation says that m is the convex combination of $S[n]$ and r with those coefficients. However, since r is the unknown and $S[n]$ and m are given, we can determine r as the inverse operation of the convex combination above, which is the extension ray combination with origin in $S[n]$ passing through m and keeping the same weights $(\frac{\alpha}{1+\alpha}, \frac{1}{1+\alpha})$ of the convex combination.

The definition of the expansion operation is $e = r + \gamma(r - m)$ (see Algorithm 1, line 18) and it can be rewritten as $r = \frac{1}{\gamma}m + \frac{\gamma-1}{\gamma}e$, which for $\gamma > 1$ is a convex combination of m and e returning r. Analogously as the reflection operation, since e is unknown and m and r are given, we can determine e by the extension ray combination with origin in m passing through r with weights $(\frac{1}{\gamma}, \frac{\gamma-1}{\gamma})$.

The definition of the contraction operation is $c = \rho r + (1 - \rho)m$ (see Algorithm 1, line 28), which for $\rho \in [0, 1]$ is a convex combination of r and m with weights $(\rho, 1 - \rho)$ returning c.

The definition of the shrink operation for a point $S[i]$ is $S[i]' = S[0] + \sigma(S[i] - S[0])$ (where $S[i]'$ denotes $S[i]$ at the next time step) (see Algorithm 1, line 37). This can be rewritten as $S[i]' = (1-\sigma)S[0] + \sigma S[i]$, which for $\sigma \in [0, 1]$ is a convex combination of $S[0]$ and $S[i]$ with weights $(1-\sigma, \sigma)$ returning $S[i]'$.

By replacing in Algorithm 1 the original operations defined on the Euclidean space with their generalized definitions we obtain the definition of a Formal Nelder-Mead Algorithm valid for any metric space (see Algorithm 2).

B. Convex Combination, Extension Ray and Center of Mass

Center of mass, segments and extension rays in the Euclidean space and their weighted extensions can be expressed in terms of distances, hence, these geometric objects can be naturally generalized to generic metric spaces by replacing the Euclidean distance with a generic metric.

Let (S, d) be a metric space. A (metric) segment is a set of the form $[x; y] = \{z \in S | d(x, z) + d(z, y) = d(x, y)\}$ where $x, y \in S$. The notion of convex combination in metric spaces was introduced in the GPSO framework [6]. The convex combination $C = CX((A, W_A), (B, W_B))$ of two points A and B with weights W_A and W_B (positive and summing up to one) in a metric space endowed with distance function d returns the set of points C in the segment $[A; B]$ such that $d(A, C)/d(A, B) = W_B$ and $d(B, C)/d(A, B) = W_A$ (the weights of the points A and B are inversely proportional to their distances to C). When specified to Euclidean spaces, this notion of convex combination coincides with the traditional notion of convex combination of real vectors.

The extension ray $ER(A, B)$ in the Euclidean plane is a semi-line originating in A and passing through B (note that $ER(A, B) \neq ER(B, A)$). The notion of extension ray in metric spaces was introduced in the GDE framework [11]. The weighted extension ray ER is defined as the inverse operation of the weighted convex combination CX, as follows. The weighted extension ray $ER((A, w_{ab}), (B, w_{bc}))$ of the points A (origin) and B (through) and weights w_{ab} and w_{bc} returns those points C such that their convex combination with A with weights w_{bc} and w_{ab}, $CX((A, w_{ab}), (C, w_{bc}))$, returns the point B.

The notion of center of mass was generalized to generic metric spaces in the GNMA framework [8], as follows. The center of mass CM of a set of points $p_1, ..., p_n$ in a metric space (S, d) is the point $p \in S$ that minimizes its average distance to that set of points, i.e. $CM(p_1, ..., p_n) = argmin_{p \in S} \frac{\sum_{i=1...n} d(p_i, p)}{n}$.

IV. GNMA Search Operators for Permutations

In this section, we derive formally specific convex combination, extension ray recombination and center of mass operator for the space of permutations. In this paper, we use the swap distance between permutations as basis for the GNMA. These specific operators can then be plugged in the formal GNMA (algorithm 2) to obtain a specific GNMA for the space of permutations, the permutation-based GNMA. Notice, however, that in principle, we could choose any other distance between permutations (e.g., adjacent swap distance, reversal distance, insertion distance, etc.) as a basis of the GNMA. In that case, for each distance, we would obtain a different permutation-based GNMA.

A. Swap distance

The swap distance between two permutations is the minimum number of swaps needed to order one permutation into the order of the other permutation. It can be implemented as in Algorithm 3.

Algorithm 3 Swap distance

```
inputs: permutations p_a and p_b
set dist = 0
for all position i in the permutations do
  if p_a(i) ≠ p_b(i) then
    find p_a(i) in p_b and be j its position in p_b
    swap contents of p_b(i) and p_b(j)
    dist = dist + 1
  end if
end for
return dist
```

B. Convex combination

Algorithm 4 presents a recombination operator for permutations that was introduced in the GPSO framework [7]. This operator produces an offspring by sorting by swaps the two parent permutations one towards the other until they converge to the same permutation. Which of the two permutations has to be sorted toward the other at each position is controlled by the contents of a random recombination mask generated using the parents weights interpreted as probabilities of the outcome of tossing a biased coin being the respective parent. This operator is called 'convex combination' because it is allegedly a convex combination for permutations under swap distance. The following two theorems prove it.

Algorithm 4 Convex combination

```
inputs: permutations p_a and p_b, and their weights w_a and w_b
generate a recombination mask m randomly with 'a' and 'b' with probabilities w_a and w_b
for all position i in the permutations do
  if p_a(i) ≠ p_b(i) then
    if m(i) = a then
      find p_a(i) in p_b and be j its position in p_b
      swap contents of p_b(i) and p_b(j)
    else
      find p_b(i) in p_a and be j its position in p_a
      swap contents of p_a(i) and p_a(j)
    end if
  end if
end for
return p_a as offspring
```

Theorem 1: The convex combination in Algorithm 4 is a geometric crossover under swap distance [7].

Additionally, in previous work [7], it was shown that the distances of the parents to the offspring are decreasing functions of their weights in the convex combination. In the following, we give a stronger result that says that that these distances are inversely proportional to the corresponding weights, as required by the refined definition of convex combination introduced in the GNMA framework [8].

Theorem 2: The convex combination in Algorithm 4 is (in expectation) a convex combination in the space of permutations endowed with swap distance.

Proof: The convex combination for permutations is a geometric crossover under swap distance. Therefore, the offspring of the convex combination are in the segment between parents as required to be a convex combination. To complete the proof, we need to show that the weights w_a and w_b of the convex combination are inversely proportional to the expected distances $E[SD(p_a, p_c)]$, $E[SD(p_b, p_c)]$ from the parents p_a

and p_b to their offspring p_c, as follows.

The recombination mask m contains a set of independently generated choices. The effect of each choice is sorting p_a a single swap towards p_b with probability w_b and sorting p_b a single swap towards p_a with probability w_a, when p_a and p_b differ at the current position. When p_a and p_b are equal at the current position, the effect of the choice is to leave p_a and p_b unchanged. When all choices in the mask m have been applied p_a and p_b have become equal in all positions, hence converged to the offspring p_c. Since the convex combination operator is a geometric crossover, the offspring p_c is on a shortest path between p_a and p_b (shortest sorting trajectory by swaps). The expected number of swap moves on the shortest path from p_a toward p_b to reach p_c, i.e., $E[SD(p_a, p_c)]$, is given by the number of swap moves on the shortest path, i.e., $SD(p_a, p_b)$, multiplied by the probability that any swap move on the shortest path was obtained by ordering p_a toward p_b, i.e., w_b. Hence $E[SD(p_a, p_c)] = SD(p_a, p_b) \cdot w_b$. Analogously for the other parent we obtain: $E[SD(p_b, p_c)] = SD(p_a, p_b) \cdot w_a$. Therefore, the expected distances of the parents to the offspring are inversely proportional to their respective weights. ■

C. Extension ray

Algorithm 5 presents a recombination operator that is allegedly the extension ray recombination for permutations under swap distance. This operator produces an offspring permutation by sorting by swaps parent permutation p_b away from parent permutation p_a. The number of swaps away is calculated in a way to obtain consistency between weights and distances of the offspring to the parents as required from the general definition of extension ray recombination in metric space. The following theorem proves that this is indeed an extension ray recombination for permutations under swap distance.

Algorithm 5 Extension ray recombination

1: inputs: parent p_a (origin point of the ray) and p_b (passing through point of the ray), with corresponding weights w_{ab} and w_{bc} (both weights are between 0 and 1 and sum up to 1)
2: output: a single offspring p_c (a point on the extension ray beyond p_b on the ray originating in p_a and passing through p_b)
3: compute the swap distance $SD(p_a, p_b)$ between p_a and p_b
4: set $SD(p_b, p_c) = SD(p_a, p_b) \cdot w_{ab}/w_{bc}$ (compute the distance between p_b and p_c using the weights)
5: set $p = SD(p_b, p_c)/(n - 1 - SD(p_a, p_b))$ (the probability p of swapping elements away from p_a and p_b beyond p_b)
6: set $p_c = p_b$
7: **for all** position i in the permutations **do**
8: **if** $p_c(i) = p_a(i)$ and random(0,1) $\leq p$ **then**
9: select at random a position j
10: swap contents of $p_c(i)$ and $p_c(j)$
11: **end if**
12: **end for**
13: return p_c as offspring

Theorem 3: The extension ray recombination in Algorithm 5 is (in expectation) an extension ray operator in the space of permutations endowed with swap distance.

Proof: First we prove that $p_c = ER(p_a, p_b)$ by proving that p_b is in the segment between p_a and p_c under swap distance. Then we prove that the expected distances $E[SD(p_a, p_b)]$ and $E[SD(p_b, p_c)]$ are inversely proportional to the weights w_{ab} and w_{bc}, respectively.

Every swap move applied to p_b that increases the Hamming distance between p_a and p_b generate a permutation p'_b such that p_b is on a swap shortest path between p_a and p'_b. This is because (i) p'_b is a swap away from p_b, i.e., $SD(p_b, p'_b) = 1$ and (ii) p'_b is a swap further away from p_a since $HD(p_a, p'_b) > HD(p_a, p_b)$, i.e., $SD(p_a, p_b) + 1 = SD(p_a, p'_b)$. Hence $SD(p_a, p_b) + SD(p_b, p'_b) = SD(p_a, p'_b)$. This construction can be continued applying a swap move to p'_b obtaining a p''_b such that p'_b and p_b are on a swap shortest path between p_a and p''_b. Analogously, for any further reiteration, we obtain $p_b^{(n)}$ such that p_b is on a swap shortest path between p_a and $p_b^{(n)}$. Since the operator ER constructs the offspring p_c (corresponding to $p_b^{(n)}$) from parents p_a and p_b following the above procedure, we have that p_b is in the segment between p_a and p_c under swap distance.

The probability p is the probability of applying a swap away from p_a for each position i, for which p_a equals p_b. The wanted distance $SD(p_b, p_c)$ to have distances and weights of parents inversely proportional is calculated from the weights w_{ab} and w_{bc}, and the known distance $SD(p_a, p_b)$. The probability p is then set to $SD(p_b, p_c)$ over the number of positions for which p_a equals p_b. This number is well estimated by the maximum number of swaps away from p_a that can be applied to p_b. The last number is given by the length of the diameter of the space (maximum swap distance between any two permutations), which is $n - 1$ where n is the number of elements in the permutation, minus the swap distance between p_a and p_b. Hence, the expected number of swaps away from p_b done equals the wanted distance $SD(p_b, p_c)$. ■

The extension ray recombination operator for permutations cannot return points which are farther away than the diameter of the space. When input weights require this, the point actually returned by the operator is the farthest away point on the extension ray.

D. Center of Mass

When specified to the Hamming space on binary strings the center of mass CM coincides with the multi-parental recombination that returns the offspring by taking position-wise the majority vote of the parents [8]. This result generalizes straightforwardly to non-binary strings under Hamming distance. Permutations can be seen as a particular type of non-binary strings with interdependencies between elements. The Hamming distance and the swap distance are intimately related distances as a single swap to a permutation can change its Hamming distance to another fixed permutation at most of two. This may suggest that the center of mass operator for permutations under swap distance, similarly to those for binary strings, is a form of majority vote on permutations which keeps adequately into account the interdependencies between different dimensions in the permutation. A multi-parental operator following this line of thinking is reported

Algorithm 6 Center of Mass Operator

inputs: parent permutations $P_1, P_2, ..., P_n$
make a copy of the parents
for all position i in the parent permutations **do**
  find the most frequent element MFE_i and its frequency F_i at position i
end for
while all positions have not yet been considered **do**
  let $ElMax$ be the element among MFE_i with the highest frequency at a single position and be $PosMax$ its position (break ties at random)
  for all parents P_i **do**
    do a swap in P_i to position element $ElMax$ in position $PosMax$
  end for
  update the tables of most frequent element MFE_i and their frequency F_i
  mark $PosMax$ as a considered position
end while
return any of the parent P_i (they are all converged to the same permutation)

in Algorithm 6. The next theorem shows that this is indeed a center of mass operator.

Theorem 4: The operator in Algorithm 6 is a center of mass operator in the space of permutations endowed with swap distance.

Proof: From the definition of center of mass operator CM, we have to prove that the offspring permutation P returned by CM minimizes the average swap distance from P to the parent permutations $P_1, P_2, ..., P_n$, i.e., P minimizes $\frac{\sum_{i=1...n} sd(P_i,P)}{n}$. Since n is constant in P, this is equivalent to finding the permutation P such that the sum of the distances from P to all the parents is minimized.

The search operator proposed minimizes the sum of these distances for the following reasoning. (i) The operator sorts by swaps all parents toward the same permutation P, which is then returned as output. The sum of the distances from the output to the parents is then the total number of swaps needed to sort all parents to the common permutation. (ii) To sort a permutation towards another permutation using the minimum number of swaps one needs to apply all the times a swap that fixes at least a position in the permutation being sorted (put the correct element, which matches the target permutation, in the chosen position). A swap with this characteristic is called sorting swap. The permutation is sorted when all positions have been fixed. (iii) If the permutation P was known and fixed, the order in which the positions are considered to be matched to P is irrelevant and one obtains the minimum number of swaps as long as all the time sorting swaps are employed. This would sort each parent permutations to P on the minimum sorting trajectory, so the total number of swaps employed to sort all parents is the minimum. (iv) However since P is the unknown to be determined, one can chose the contents of each position of P in a way that the increment to the sum of the minimum sorting trajectory (under construction) from each parent to P is as small as possible. The minimum increment is obtained by choosing to fix the maximum occurring element $ElMax$ in the position in which it occurs the most ($PosMax$), as done by the proposed operator. This is, in fact, the choice that minimizes the number of sorting swaps to apply to all the parents to fix one element to a position (to fix $ElMax$ at $PosMax$) because it is the position (and element) that is already the most correct (no need to sort it) across all parents and positions. ■

Unlike for the Euclidean case in which the simplex is maintained non-degenerate throughout the search process, so guaranteeing that any dimension is actually being searched, this does not hold true for the cases of the Hamming space and permutation spaces. To counteract the degeneracy of the simplex, in the experiments we will use a randomized version of the CM operator which uses the frequency of the most frequent element at each position in the parents as the probability of the offspring to have that element at that position rather than fixing that element deterministically (fixing deterministically $PosMax$ to the element $ElMax$). The expected offspring of the randomized operator is the one obtained with the Algorithm 6, but the variance of the output gives a greater chance to the search of staying open in all dimensions.

Now we have operational definitions of convex combination, extension ray and center of mass for the space of permutations under swap distance. These space-specific operators can be plugged in the formal GNMA (Algorithm 2) to obtain a specific GNMA for the space of permutations.

V. Experiments

We have tested the permutation-based GNMA on randomly generated instances of the Travelling Salesman Problem (TSP), which is perhaps the most famous permutation-based optimization problem. We do not expect the GNMA to be comparable in performance with the state-of-the-art search algorithms customized to such a well-studied problem. Also, the neighborhood structure on the TSP that works best with local search heuristics is that based on the 2-opt move which reverses the order of the elements in a continuous section of the permutation. Analogously to the swap move, this move gives rise to a distance between permutations (known as reversal distance). This would be perhaps the most suitable distance to use as a base for GNMA when applied to TSP. We will test this in future work.

Local search heuristics based on the swap move are known to do reasonably well on the TSP. Also, genetic algorithms with the PMX crossover operator for permutation, which is known to be a geometric crossover under swap distance, does reasonably well on the TSP. Therefore, as a reference, we compare the GNMA on the swap space with a stochastic hill-climber based on the swap move, with a genetic algorithm with rank-based selection, PMX crossover and swap mutation, and with our recently derived Geometric Differential Evolution.

The TSP instances used in our experiments are randomly generated with length 50. The distance between each pair of cities lies between 0 and 1, and the instances are symmetric but not Euclidean. Twenty TSP instances were generated at the beginning of the experiments; every algorithm configuration is run once per instance, and the fitness averaged over all instances.

Moderately extensive tuning experiments were performed for the population-based algorithms. All algorithms (GNMA, GDE, GA and hill climber) each run lasted 100000 (hundred

thousand) function evaluations. For GNMA, GDE GA, population sizes of 10, 20, 50, 100, 1000 were tried, with the number of generations set to $100000/popsize$. For GA and GDE, their two respective key parameters were varied between 0 and 1 in increments of 0.2; for GNMA, the parameters are F and Cr. For the GA, these parameters were defined as the elite proportion (how large part of the rank-ordered population is used as the elite; the lesser fit rest of the population is replaced each generation) and mutation probability (the probability that a new offspring is created through swap mutation from the previous individual at the same position in the population rather than using PMX crossover of two randomly selected individuals in the population). We note that some extreme settings yield degenerate versions of both algorithms.

For GNMA, the following parameters were tuned in addition to the population size (tested values in parenthesis): Alpha (1.0, 1.9), Gamma (2.0), Rho (0.8, 0.9, 0.95), Sigma (0.8, 0.9, 0.95).

Alas, for the hillclimber, there is nothing to tune.

Algorithm	Fitness	Population	Parameters
Hillclimber	5.37	-	-
GA	5.13	10	elite 0.2, mut 0.6
GDE	5.35	10	F 0.0, Cr 0.2
GNMA	6.65	500	α1.0, Γ2.0, ρ0.8, Σ0.9

TABLE I

RESULTS ON TSP INSTANCES OF SIZE 50. LOWER FITNESSES ARE BETTER.

The results in table I show that while GNMA does find reasonably good solutions (a typical best fitness of the first generation is around 20), it is not competitive with the GA, GDE nor even the hillclimber.

VI. DISCUSSION

In the previous section, we have presented initial experimental results. The new algorithm does not seem to perform as well as a genetic algorithm and a stochastic local search defined on the same search space and representation. This is a rather surprising result as the Geometric Nelder-Mead Algorithm specified to binary strings under Hamming distance performed significantly better than a genetic algorithm on NK-landscapes [8], showing that, in principle, the GNMA may work well when applied to combinatorial spaces.

At first, we tested the permutation-based NMA with standard parameter settings of the classic NMA ($\alpha = 1, \gamma = 2, \rho = 0.5, \sigma = 0.5$ and population size = genome size + 1) that worked well for the GNMA on binary strings. However, for the permutation-based GNMA we found that there was a very quick loss of diversity in the population that led to a very early stop of the search and to rather poor performances. This was quite surprising as there is no sign of diversity loss in the population when both the classic NMA and the GNMA on binary strings are run with the same parameter setting.

As all specific GNMA are really the instantiation of the same search dynamics to different search spaces, we were left to wonder about the cause of this lack of consistency among the dynamics of the permutation-based GNMA with those of the classic NMA and the GNMA on binary strings. There are two possibilities to explain the quick convergence: (i) some special feature of the topography of the fitness landscape of the TSP; (ii) some special feature of the specific geometry of the permutation space under swap distance. The second possibility seems to be the reason behind the inconsistency regarding the diversity in the population, as explained in the following.

When we sample two binary strings uniformly at random, they are likely to be at half of the maximal Hamming distance for strings of that size (i.e., half of the size of the diameter of the search space). This is because the probability at each position in the two strings to have the same bit is 0.5. For non-binary strings based on alphabets with higher cardinalities than two, the Hamming distance between strings sampled uniformly at random tends to be likely to be closer to the maximal Hamming distance for strings of that size as the cardinality of the alphabet gets higher. This is because the probability at each position in the two strings to have the same symbol diminishes (rapidly) with the size of the alphabet. So, this leads to the rather counter-intuitive result that non-binary strings sampled at random are likely to be at maximal distance from each other. An analogous result holds for permutations under swap distance due to the intimate relation between Hamming distance for permutations and swap distance between permutations. The situation for points sampled uniformly at random in the Euclidean space is similar to the case of binary strings. So, under this particular aspect, the permutation space under swap distance differs substantially from both the Hamming space for binary strings and Euclidean space. This difference in the search space affects heavily the dynamics of the NMA, as explained in the following.

The geometric operations of the NMA can be classified in two types: explorative operations – reflection and expansion – which have the effect of expanding the simplex, and exploitative operations – contraction and shrinking – which have the effect of reducing the simplex. Explorative operations are based on the extension ray operation and exploitative operations are based on the convex combination operation. When an explorative operation is applied to two points in space sampled uniformly at random, we obtain two very different behaviors depending on whether the two points are at a maximal distance or not. This is because when we apply the extension ray operator to them, say with origin A and passing through B, if A and B are at a maximal distance, there are no points in the space belonging to the extension ray except for the point B, which is then the one returned. Clearly, this is a degenerate case.

This, in fact, explains why the reflection and expansion operations in the initial population in the permutations space are not explorative enough, whereas they are explorative in the case of binary strings and continuous spaces. This is essentially because the center of mass of a set of random permutations (i.e., the initial simplex) can be seen as if it were a permutation sampled at random in itself. Also the worse

permutation in the initial population is sampled at random. So, for permutations, the center of mass and the worse element of the population are likely to be at a maximal distance. Therefore, the reflection operation cannot return a point beyond the center of mass (exploration) but it can only return the center of mass (fixation/convergence). So, the operation that should have been in charge of exploring, actually it does quite the opposite. Then, the remaining operations also will push toward convergence as they are designed to do so. This does not happen in the Hamming space for binary strings as the initial center of mass and the worst individual are likely to be at half of the maximum distance of the space. So, there can be initial exploration as the extension ray operator does not return a degenerate point, and quite the opposite, there is actually plenty of points in the extension ray between two randomly chosen binary strings. This is similar to what happens in the case of the original NMA on continuous space.

To compensate for this lack of diversity we can set parameters to preserve diversity in the initial phase of the run by considering larger populations and very mild convergence setting (for shrinking and contraction). This seems to preserve diversity and produce better performance as shown in the experimental section, that however, are still inferior to those of stochastic local search and of a genetic algorithm. A more systematic study of the parameters and analysis of the behavior of the NMA is needed to understand if this is a fundamental limit of the GNMA on the permutation representation (and non-binary string representations) or not. To do so we will test the GNMA also on other permutation-based problems.

VII. Conclusions

The Geometric Nelder-Mead Algorithm is a generalization of the classical Nelder-Mead Algorithm to general metric spaces. In particular, it applies to combinatorial spaces. The permutation representation is an important representation relevant to many combinatorial optimization problems. In this paper we have demonstrated how to specify the general Geometric Nelder-Mead Algorithm to the space of permutations under swap distance. From preliminary experimental results on the TSP, the new algorithm does not seem to perform as well as a genetic algorithm and a stochastic local search defined on the same search space and representation. This is a rather surprising result as the Geometric Nelder-Mead Algorithm specified to binary strings under Hamming distance performed significantly better than a genetic algorithm on NK-landscapes [8], showing that, in principle, the GNMA may work well when applied to combinatorial spaces. The reason behind it seems to be related with the peculiar geometric properties of the permutation space that forces the GNMA towards a degenerate dynamic. This does not happen with the traditional NMA and the binary GNMA. However, further investigation is needed to elucidate if this is a fundamental limitation of the application of the GNMA to permutation spaces. More generally, this investigation will shed light on the applicability of any formal geometric algorithm – including GDE and GPSO – to any permutation-based problem.

As additional future work, we will test this new algorithm more thoroughly and on a number of combinatorial optimization problems. Also, we will derive the GNMA for permutations under other distances, such as adjacent-swap distance and reversal distance, which may be more suitable to particular classes of problems, e.g., scheduling problems. Finally, as GNMA is a close relative of GPSO and GDE, we will present the three algorithms in a common theoretical framework highlighting their commonalities and differences and we will compare them experimentally to find out which of their characteristics are better suited to which type of problems and representations.

References

[1] M. Clerc, *Discrete particle swarm optimization, illustrated by the traveling salesman problem*, New Optimization Techniques in Engineering, Springer, 2004, pp. 219–239.

[2] J. Kennedy and R. C. Eberhart, *A discrete binary version of the particle swarm algorithm*, IEEE Transactions on Systems, Man, and Cybernetics **5** (1997), 4104–4108.

[3] _____, *Swarm intelligence*, Morgan Kaufmann, 2001.

[4] Changtong Luo and Bo Yu, *Low dimensional simplex evolution a hybrid heuristic for global optimization*, Eighth International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol. 2, 2007, pp. 470–474.

[5] A. Moraglio, *Towards a geometric unification of evolutionary algorithms*, Ph.D. thesis, University of Essex, 2007.

[6] A. Moraglio, C. Di Chio, and R. Poli, *Geometric particle swarm optimization*, European Conference on Genetic Programming, 2007, pp. 125–136.

[7] A. Moraglio, C. Di Chio, J. Togelius, and R. Poli, *Geometric particle swarm optimization*, Journal of Artificial Evolution and Applications **2008** (2008), Article ID 143624.

[8] A. Moraglio and C.G. Johnson, *Geometric generalization of the nelder-mead algorithm*, Proceedings of the 10th European Conference on Evolutionary Computation in Combinatorial Optimization, 2010.

[9] A. Moraglio and S. Silva, *Geometric differential evolution on the space of genetic programs*, Proceedings of the 13th European Conference on Genetic Programming, 2010.

[10] A. Moraglio and J. Togelius, *Geometric pso for the sudoku puzzle*, Proceedings of the Genetic and Evolutionary Computation Conference, 2007, pp. 118–125.

[11] Alberto Moraglio and Julian Togelius, *Geometric differential evolution*, Proceedings of the 11th Annual conference on Genetic and evolutionary computation, 2009, pp. 1705–1712.

[12] John Ashworth Nelder and Roger A. Mead, *A simplex method for function minimization*, Computer Journal **7** (1965), 308–313.

[13] Michael O'Neill and Anthony Brabazon, *Grammatical differential evolution*, Proceedings of the 2006 International Conference on Artificial Intelligence, CSREA Press, 2006, pp. 231–236.

[14] G. C. Onwubolu and D. Davendra (eds.), *Differential evolution: A handbook for global permutation-based combinatorial optimization*, Springer, 2009.

[15] G. Pampara, A.P. Engelbrecht, and N. Franken, *Binary differential evolution*, IEEE Congress on Evolutionary Computation, 2006.

[16] K. V. Price, R. M. Storm, and J. A. Lampinen, *Differential evolution: A practical approach to global optimization*, Springer, 2005.

[17] Tetsuyuki Takahama and Setsuko Sakai, *Constrained optimization by applying the α-constrained method to the nonlinear simplex method with mutations*, IEEE Transactions on Evolutionary Computation **9(5)** (2005), 437–451.

[18] Julian Togelius, Renzo De Nardi, and Alberto Moraglio, *Geometric pso + gp = particle swarm programming*, Proceedings of the Congress on Evolutionary Comptutation (CEC), 2008.

[19] Fang Wang and Yuhui Qiu, *Empirical study of hybrid particle swarm optimizers with the simplex method operator*, Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, 2005, pp. 308–313.

[20] Thomas Weise, *Global optimization algorithms - theory and application*, on-line ebook, 2009.