Geometric Generalization of the Nelder-Mead Algorithm

Alberto Moraglio and Colin G. Johnson

School of Computing, University of Kent, Canterbury, UK
{a.moraglio, c.g.johnson}@kent.ac.uk

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). Geometric Particle Swarm Optimization (GPSO) and Geometric Differential Evolution (GDE) are recently introduced formal generalization of traditional PSO and DE that apply naturally to both continuous and combinatorial spaces. In this paper, we generalize NMA to combinatorial search spaces by naturally extending its geometric interpretation to these spaces, analogously as what was done for the traditional PSO and DE algorithms, obtaining the Geometric Nelder-Mead Algorithm (GNMA).

1 Introduction

The Nelder-Mead Algorithm published by Nelder and Mead in 1965 [9] is a numerical optimization method: despite its age, it is 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 [13]. Also, there are similarities between the search operators employed by the NMA and those of DE [12] and PSO [2] that have led a number of authors to propose hybrid approaches (see for example [15] and [3]). 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. To the authors's best knowledge, there are no generalizations of the NMA to combinatorial spaces.

Both of the searches done by PSO and DE have natural geometric interpretations and both 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 [5] and

Geometric Differential Evolution [8] are recently devised formal generalizations of PSO and DE 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 and DE algorithms 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 [6], for spaces associated with permutations [7] and for spaces associated with genetic programs [14]. GDE was specialized to the space of binary strings endowed with the Hamming distance [8]. The derived algorithms performed satisfactorily in experimental results. This suggests that the generalization methodology employed is a promising one.

In the present paper we generalize the Nelder-Mead Algorithm to combinatorial spaces extending its geometric interpretation to these spaces, analogously to what was done for the traditional PSO and DE algorithms, derive the specific GNMA for the Hamming space associated with binary strings and present experimental results on standard benchmark problems.

2 Classic Nelder-Mead Algorithm

In this section, we describe the traditional NMA [9] (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

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 ← createPop(n + 1)
while stop criterion not met do
  S ← sortPop(S, f)
  // Center of mass: determine the center of mass of the n best points
  m ← (1/n) Σ_{i=0,n−1} S[i]
  // Reflection: reflect the worst point over m
  r ← m + α(m − S[n])
  if f(S[0]) < f(r) < f(S[n]) then
    S[n] ← r
  else
    if f(r) ≤ f(S[0]) then
      // Expansion: try to search farther in this direction
      e ← r + γ(r − m)
      if f(e) < f(r) then
        S[n] ← e
      else
        S[n] ← r
      end if
    else
      b ← true
      if f(r) ≥ f(S[n − 1]) then
        // Contraction: a test point between r and m
        c ← ρr + (1 − ρ)m
        if f(c) < f(r) then
          S[n] ← c
          b ← 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] ← S[0] + σ(S[i] − S[0])
        end for
      end if
    end if
  end if
end while
return S[0]
```

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).

3 Geometric Nelder-Mead Algorithm

In this section, we derive the general Geometric Nelder-Mead Algorithm (Algorithm 2) from the classic Nelder-Mead Algorithm (Algorithm 1). The generalization is obtained using a methodology to generalize search algorithms for

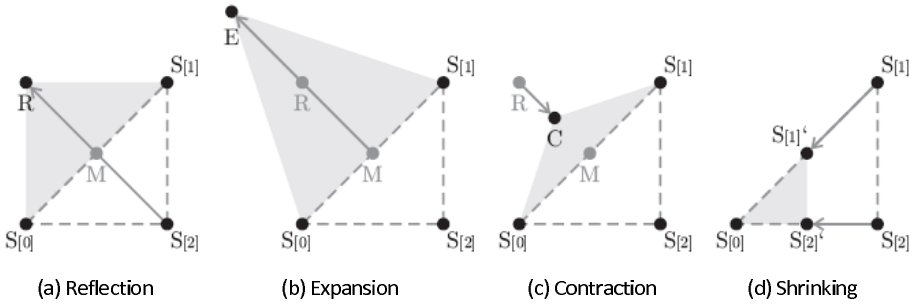


Fig. 1. One step of the NMA in R^2 (figure modified from [16])

continuous spaces to combinatorial spaces [8] based on the geometric framework introduced by Moraglio [4], 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.

As mentioned in the introduction, this methodology was used to generalize PSO and DE to any metric space obtaining GPSO and GDE and then to derive the specific search operators for a number of specific representations and distances.

Following the methodology outlined above, in the following we generalize the classic Nelder-Mead Algorithm to general metric spaces. To do this, we recast 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. Then, in Section 4, we derive the specific GNMA for the Hamming space associated with binary strings by plugging this distance in the abstract definition of the search operators.

3.1 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 sections, these are geometric elements well-defined on any metric space.

Algorithm 2 Formal 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 candidate found

S ← createPop(n + 1)
while stop criterion not met do
  S ← sortPop(S, f)
  // Center of mass: determine the center of mass of the n best points
  m ← CM(S[0], S[1], ..., S[n − 1])
  // Reflection: reflect the worst point over m
  r ← ER(S[n], m) with weights (α/(1+α), 1/(1+α))
  if f(S[0]) < f(r) < f(S[n]) then
    S[n] ← r
  else
    if f(r) ≤ f(S[0]) then
      // Expansion: try to search farther in this direction
      e ← ER(m, r) with weights (1/γ, (γ−1)/γ)
      if f(e) < f(r) then
        S[n] ← e
      else
        S[n] ← r
      end if
    else
      b ← true
      if f(r) ≥ f(S[n − 1]) then
        // Contraction: a test point between r and m
        c ← CX(r, m) with weights (ρ, 1 − ρ)
        if f(c) < f(r) then
          S[n] ← c
          b ← 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] ← CX(S[0], S[i]) with weights (1 − σ, σ)
        end for
      end if
    end if
  end if
end while
return S[0]
```

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).

3.2 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 [5]. 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 [8]. The weighted extension ray ER is defined as the inverse operation of the weighted convex combination CX, as follows. The weighted extension ray

Algorithm 3 Binary Convex Combination Operator

```
inputs: binary strings A and B and weights W_A and W_B (weights must be positive and sum up to 1)
for all position i in the strings do
    if random(0,1) ≤ W_A then
        set C(i) to A(i)
    else
        set C(i) to B(i)
    end if
end for
return string C as offspring
```

$ER((A, w_{ab}), (B, w_{bc}))$ 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 can be generalized to generic metric spaces, 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. $CM(p_1, ..., p_n) = argmin_{p \in S} \frac{\sum_{i=1...n} d(p_i, p)}{n}$.

4 Binary NMA

In this section, we present convex combination, extension ray and center of mass operators for the Hamming space on binary strings, and show formally that they meet their geometric specifications presented in the previous section. These specific operators can be plugged in the formal NMA (Algorithm 2) to obtain a specific GNMA for the Hamming space, the Binary GNMA.

4.1 Convex combination and extension ray

The convex combination operator in metric spaces was introduced in the GPSO framework [5]. When specified to Euclidean spaces, this notion of convex combination coincides with the traditional notion of convex combination of real vectors. In the Euclidean space, the output point C of a convex combination $CX((A, W_A), (B, W_B))$ is uniquely determined, however this is not the case for all metric spaces. In particular, it does not hold for Hamming spaces. When CX is formally specified to Hamming spaces on binary strings, we obtain the recombination operator outlined in Algorithm 3 [5], which is a weighted form of uniform crossover. This algorithm returns offspring C in the Hamming segment between A and B such that $hd(A, C)/hd(B, C) = W_B/W_A$ in expectation. This differs from the Euclidean case where this ratio is guaranteed.

The notion of extension ray in metric spaces was introduced in the GDE framework [8]. When specified to Euclidean spaces, this notion of extension ray coincides with the traditional notion. Analogously as for the convex combination case, in the Euclidean space, the output point C of an extension ray combination $ER((A, w_{ab}), (B, w_{bc}))$ is uniquely determined, however this is not the case for

Algorithm 4 Binary Extension Ray Recombination

inputs: binary strings A (origin) and B (through) of length n and weights W_{AB} and W_{BC} (weights must be positive and sum up to 1)
set $HD(A, B)$ as Hamming distance between A and B
set $HD(B, C)$ as $HD(A, B) \cdot w_{AB}/w_{BC}$ (compute the distance between B and C using the weights)
set p as $HD(B, C)/(n - HD(A, B))$ (this is the probability of flipping bits away from A and B beyond B)
for all position i in the strings **do**
  set $C(i) = B(i)$
  if $B(i) = A(i)$ and random(0,1) $\leq p$ **then**
    set $C(i)$ to the complement of $B(i)$
  end if
end for
return string C as offspring

Algorithm 5 Binary Center of Mass Operator

inputs: binary strings $A[1], ..., A[n]$
for all position i in the strings **do**
  $C(i) = 1/n \cdot \sum_{j=1,n} A[j](i)$
  if $C(i) = 0.5$ **then**
    $C(i) = RandomInteger(0, 1)$
  else
    $C(i) = Round(C(i))$
  end if
end for
return string C as offspring

all metric spaces. In particular, it does not hold for Hamming spaces. When ER is formally specified to Hamming spaces on binary strings, we obtain the recombination operator outlined in Algorithm 4 [8]. This algorithm implements the inverse operation of the convex combination CX reported above in that it returns offspring C such that the parent B is in the Hamming segment between A and C and that $hd(A, C)/hd(B, C) = w_{bc}/w_{ab}$ in expectation. This differs from the Euclidean case where this ratio is guaranteed.

4.2 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 and breaking ties randomly (see Algorithm 5). We prove this in the following.

Theorem 1. *The binary string p returned by the binary center of mass operator CM (Algorithm 5) applied to parents $p_1, ..., p_n$ minimizes the average Hamming distance to its parents.*

Proof. From the definition of center of mass operator, we have to prove that p minimizes $\frac{\sum_{i=1...n} hd(p_i,p)}{n}$. Since n is constant in p, this is equivalent to prove that p minimizes $\sum_{i=1...n} hd(p_i, p)$. By expanding the definition of Hamming distance and exchanging the summations, we have $\sum_{k=1..m} \sum_{i=1...n} hd(p_i^k, p^k)$

where k is the position in the string and m the number of bits in the string. The last expression is a linear combination of the (boolean) variables p^k. Minimizing the linear combination is equivalent to minimizing each term $\sum_{i=1\ldots n} hd(p_i^k, p^k)$ in p^k separately. Minimizing a term corresponds to finding the center of mass p^k of the strings p_i^k of a single bit size. Since p^k can take only two values we have only two cases: (i) when $p^k = 0$ the term $\sum_{i=1\ldots n} hd(p_i^k, p^k)$ is the number of bits in $\{p_i^k\}$ set to 1; (ii) when $p^k = 1$ the term $\sum_{i=1\ldots n} hd(p_i^k, p^k)$ is the number of bits in $\{p_i^k\}$ set to 0. So, $\sum_{i=1\ldots n} hd(p_i^k, p^k)$ is minimized by $p^k = 0$ when the number of bits in $\{p_i^k\}$ set to 1 is less than the number of bits set to 0; it is minimized by $p^k = 1$ when the number of bits in $\{p_i^k\}$ set to 0 is less than the number of bits set to 1. This is a way of describing majority voting for strings of a single bit size. This reasoning applies to any position k in the string independently, so we conclude that the majority voting returns the p that minimizes $\frac{\sum_{i=1\ldots n} hd(p_i,p)}{n}$.

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 case of the Hamming space. 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 ones and zeros at each position in the parents as the probability of generating a one or a zero in the offspring at that position. The expected offspring of the randomized operator is the one obtained with majority voting but the variance gives a greater chance to the search of staying open in all dimensions.

5 Experiments: Results and Discussion

Experiments have been carried out using the well-known NK-Landscape problem [1], which provides a tunable set of rugged, epistatic landscapes over a space of binary strings. In our experiments we use landscapes of size $n = 20, \ldots, 52$ for $k = 2, \ldots, 6$ (higher values of n have not been provided for the higher values of k due to the difficulty of calculating the optimum using exact methods). These examples are due to Pelikan [11], and can be downloaded from Pelikan's website [10].

Two algorithms have been used. The GNMA above (referred to in the diagrams as NM) with populations 10 and 100, and a genetic algorithm (referred to in the diagrams as GA) with population sizes 100, 500 and 1000. Note that it is not fair to compare algorithms directly on identical population sizes as the meaning of population size in the two algorithms is very different. The GA uses uniform crossover with probability 0.8, bitflip mutation with a probability of $1/n$, elitism and roulette-wheel selection (i.e. we have chosen standard values from the literaure). Both algorithms are run until they converge to a value; in the case of the GA this is taken to mean that the best value has the same value as in the previous 4 generations.

For each tuple of algorithm, population size n and k, the algorithm has been executed 10 times on each of 999 examples of NK-landscapes—therefore each

point in the results graphs represents 9990 runs of that particular NK pair. Three features have been measured: error on the objective function (measured against exactly-calculated optima computed using branch-and-bound [11]), number of function evaluations, and number of times the global optimum was found exactly.

Results are presented in Figure 2. The first set of results shows error measures (further comparisons, with errorbars, can be seen on the authors's website). The error is presented as absolute error in arbitary units (i.e., *not* as a percentage or proportion of the optimum). The next pair of figures shows the number of function evaluations taken to convergence (of the population, not necessarily to convergence to optimum). The final set of results shows the number of runs (out of 9990) in which the exact optimum was found. Results for $k = 3, 4, 5$ are intermediate between $k = 2$ and $k = 6$, and can be found on the authors's web site.

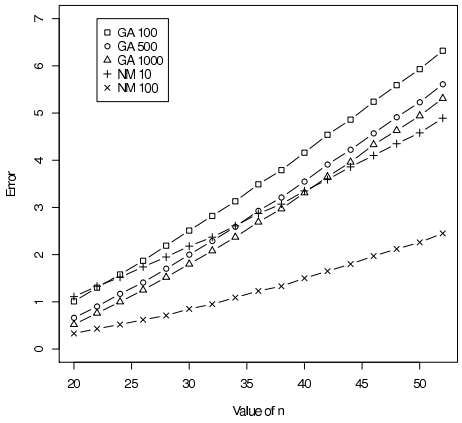
For all problems, the NM-100 algorithm is the best (or equal-best) algorithm, in particular performing considerably better than the other algorithms on the problems with larger values for n, indicating good scaling performance. Clearly the NM-100 algorithm is using the most function evaluations of any of the algorithms; however, the current version of the GNMA continues until the whole population has converged on a single individual, so a better stopping condition might provide a better measure for this. Interestingly, the NM-100 algorithm achieves a considerably larger number of exact hits precisely on the global optimal value.

The experiments above, that have compared GNMA with a simple GA, are intended to show that the GNMA is a promising proof-of-concept rather than showing its competitiveness to state-of-the-art algorithms. As an important piece of future work, we will examine the performance of GNMA more thoroughly and compare it more broadly with a number of competitive algorithms including GPSO and GDE on a larger set of benchmark problems.

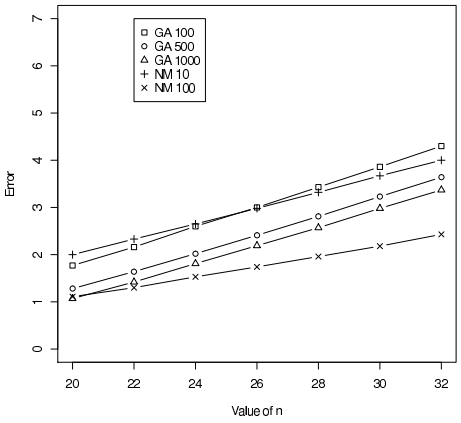
6 Conclusions

In this paper, we have generalized the Nelder-Mead Algorithm from continuous to generic combinatorial spaces by extending the geometric interpretation of the classic NMA to general metric spaces. The algorithm obtained (GNMA) can then be formally specified for specific spaces and specific representations. We have illustrated this by deriving the specific GNMA for the Hamming space associated with binary strings. We have tested the binary GNMA on a standard benchmark and compare it with a simple genetic algorithm.

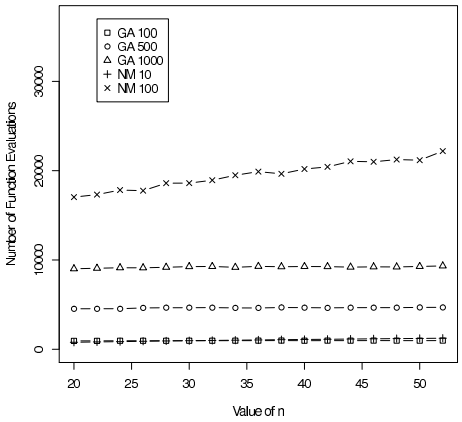
In future work, we will specify the formal GNMA for search spaces associated with permutations and test it on hard combinatorial optimization problems. The Nelder-Mead Algorithm is similar to other classical derivation-free methods for continuous optimization that make use of geometric constructions of points to determine the next candidate solution (e.g., Controlled Random Search). We will use the same technique to generalize these algorithms to general metric spaces.



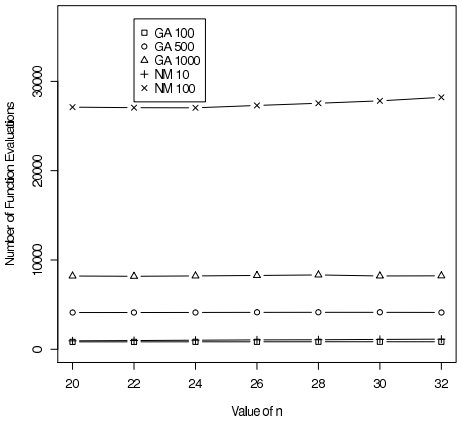
Error: $k = 2$



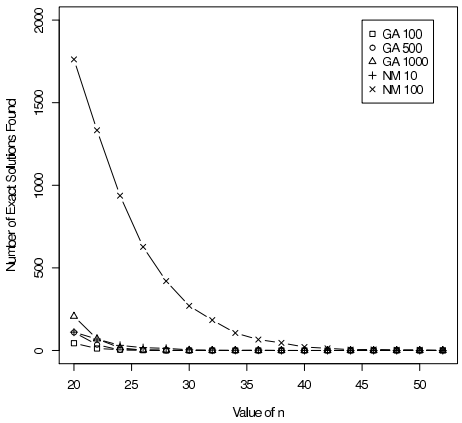
Error: $k = 6$



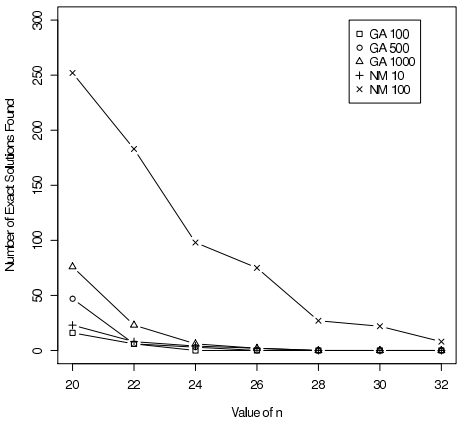
Function evaluations: $k = 2$



Function evaluations: $k = 6$



Exact optimum found: $k = 2$



Exact optimum found: $k = 6$

Fig. 2. Results for the five algorithms

References

1. S. Kauffman. *Origins of order: self-organization and selection in evolution*. Oxford University Press, 1993.
2. J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.
3. C. Luo and B. Yu. Low dimensional simplex evolution: a hybrid heuristic for global optimization. In *Eighth International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, volume 2, pages 470–474, 2007.
4. A. Moraglio. *Towards a geometric unification of evolutionary algorithms*. PhD thesis, University of Essex, 2007.
5. A. Moraglio, C. D. Chio, and R. Poli. Geometric particle swarm optimization. In *European Conference on Genetic Programming*, pages 125–136, 2007.
6. A. Moraglio, C. D. Chio, J. Togelius, and R. Poli. Geometric particle swarm optimization. *Journal of Artificial Evolution and Applications*, 2008:Article ID 143624, 2008.
7. A. Moraglio and J. Togelius. Geometric pso for the sudoku puzzle. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 118–125, 2007.
8. A. Moraglio and J. Togelius. Geometric differential evolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1705–1712, 2009.
9. J. A. Nelder and R. A. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
10. M. Pelikan. NK landscape generator and instances provided online.
11. M. Pelikan. Analysis of estimation of distribution algorithms and genetic algorithms on nk-landscapes. In *Proceedings of the 2008 Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 1033–1040. ACM Press, 2008.
12. K. V. Price, R. M. Storm, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.
13. T. Takahama and S. Sakai. Constrained optimization by applying the α-constrained method to the nonlinear simplex method with mutations. *IEEE Transactions on Evolutionary Computation*, 9(5):437–451, 2005.
14. J. Togelius, R. D. Nardi, and A. Moraglio. Geometric pso + gp = particle swarm programming. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, 2008.
15. F. Wang and Y. Qiu. Empirical study of hybrid particle swarm optimizers with the simplex method operator. In *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pages 308–313, 2005.
16. T. Weise. *Global Optimization Algorithms - Theory and Application*. on-line ebook, 2009.