Geometric Differential Evolution

Alberto Moraglio
Centre for Informatics and Systems
of the University of Coimbra (CISUC)
Polo II - University of Coimbra
Coimbra 3030-290, Portugal
moraglio@dei.uc.pt

Julian Togelius
Dalle Molle Institute
for Artificial Intelligence (IDSIA)
Galleria 2
Manno-Lugano 6928, Switzerland
julian@idsia.ch

ABSTRACT

Geometric Particle Swarm Optimization (GPSO) is a recently introduced formal generalization of traditional Particle Swarm Optimization (PSO) that applies naturally to both continuous and combinatorial spaces. Differential Evolution (DE) is similar to PSO but it uses different equations governing the motion of the particles. This paper generalizes the DE algorithm to combinatorial search spaces extending its geometric interpretation to these spaces, analogously as what was done for the traditional PSO algorithm. Using this formal algorithm, Geometric Differential Evolution (GDE), we formally derive the specific GDE for the Hamming space associated with binary strings and present experimental results on a standard benchmark of problems.

1. INTRODUCTION

Two relatively recent additions to the Evolutionary Algorithms (EAs) family are Particle Swarm Optimization [4], inspired to the flocking behavior of swarms of birds, and Differential Evolution [12], which is similar to PSO, but it uses different equations governing the motion of the particles. Despite their relatedness, DE is known to produce consistently better performance than PSO on many problems. In fact, DE is one of the most competitive EAs for continuous optimization [12].

In their initial inception, both PSO and DE were defined only for continuous problems. In both algorithms, the motion of particles is produced by linear combinations of points in space and has a natural geometric interpretation. There are only few extensions of DE to combinatorial spaces [12] [11] [1] [10]. Some of these works recast combinatorial optimization problems as continuous optimization problems and then apply the traditional DE algorithm to solve these continuous problems. Other works present DE algorithms defined directly on combinatorial spaces that, however, are only loosely related to the traditional DE in that the original geometric interpretation is lost in the transition from continuous to combinatorial spaces. Furthermore, every time a new solution representation is considered, the DE algorithm needs to be rethought and adapted to the new representation.

GPSO [7] is a recently devised formal generalization of PSO that retains a simple geometric interpretation of the dynamics of the particles in space. GPSO can be applied to any search space endowed with a distance and associated with any solution representation to derive formally a specific GPSO for the target space. Recently, specific GPSOs were derived for different types of continuous spaces and for the Hamming space associated with binary strings [8], for spaces associated with permutations [9] and for spaces associated with Genetic Programming trees [14].

The objective of the present paper is to generalize the DE algorithm to combinatorial spaces extending its geometric interpretation to these spaces, analogously to what was done for the traditional PSO algorithm, derive the specific GDE for the Hamming space associated with binary strings and present experimental results on standard benchmark problems.

2. THE GEOMETRY OF REPRESENTATIONS

In this section, we introduce the ideas behind a recent formal theory of representations [6] which forms the context for the generalization of DE presented in the following sections.

Familiar geometric shapes in the Euclidean plane such as circles, ellipses, segments, semi-lines, triangles and convex polygons can be defined using distances between points in space. For example, a circle is the locus of points from which the distance to the centre c is a given constant value, the radius r. By replacing in the definition of a shape, say a circle, the Euclidean distance with a different distance, say the Hamming distance, we obtain the definition of a circle in the Hamming space. A circle in the Hamming space looks quite different from a circle in the Euclidean plane, however they both share the same geometric definition. Analogously, if we replace the Euclidean distance with the Manhattan distance, we obtain the definition of a circle in the Manhattan space. A number of simple geometric shapes based on the Manhattan distance in the plane have been derived explicitly (see Taxicab Geometry [5]). We can in fact replace the Euclidean distance in the definition of any geometric shape with any distance meeting a minimum number of requirements (metric), obtaining the corresponding shape in a space with a different geometry. We can also raise the level of ab-

straction and replace the Euclidean distance with a generic metric, obtaining an abstract shape, such as for example an abstract circle. An abstract circle captures what is common to all circles across all possible geometries. Any property of an abstract circle is also a property of any space-specific circle.

Search algorithms can be viewed from a geometric perspective. The search space is seen as a geometric space with a notion of distance between points, and candidate solutions are points in the space. For example, search spaces associated with combinatorial optimization problems are commonly represented as graphs in which nodes corresponds to candidate solutions and edges between solutions correspond to neighbour candidate solutions. We can endow these spaces with a distance between solutions equal to the length of the shortest path between their corresponding nodes in the graph. Geometric search operators are defined using geometric shapes to delimit the region of search space where to sample offspring solutions relative to the positions of parent solutions. For example, geometric crossover is a search operator that takes two parent solutions in input corresponding to the end-points of a segment, and returns points sampled at random within the segment as offspring solutions. The specific distance associated with the search space at hand is used in the definition of segment to determine the specific geometric crossover for that space. Therefore, each search space is associated with a different space-specific geometric crossover. However, all geometric crossovers have the same abstract geometric definition.

In analytic geometry, in which points of the Cartesian plane are in one-to-one correspondence with pairs of numbers, their coordinates, the same geometric shape can be equivalently expressed geometrically as a set of points in the plane, or algebraically, by an equation whose solutions are the coordinates of its points. This is an important duality which allows us to treat geometric shapes as equations and vice versa. There is an analogous duality that holds for geometric search operators. Candidate solutions can be seen as points in space, geometric view, or equivalently, as syntactic configurations of a certain type, algebraic view. For example, a candidate solution in the Hamming space can be considered as a point in space or as a binary string corresponding to that point. The binary string can then be thought as being the coordinates of the point in the Hamming space. This allows us to think of a search operator equivalently as (i) an algorithmic procedure which manipulates the syntax of the parent solutions to obtain the syntactic configurations of the offspring solutions using well-defined representation-specific operations (algebraic view), or (ii) a geometric description which specifies what points in the space can be returned as offspring for the given parent points and with what probability (geometric view). For example, uniform crossover for binary strings [13] is a recombination operator that produces offspring binary strings by inheriting at each position in the binary string the bit of one parent string or of the other parent string with the same probability. This is an algebraic view of the uniform crossover that tells how to manipulate the parent strings to obtain the offspring string. Equivalently, the same operator can be defined geometrically as the geometric crossover based on the Hamming distance that takes offspring uniformly at random in the segment between parents.

There are two important differences between these two definitions of the same operator. The geometric definition is declarative, it defines what offspring the operator returns given their parents without explicitly telling how to actually generate the offspring from the parents. The algebraic definition, on the other hand, is operational, since it defines the search operator by telling for each combination of parents how to build the corresponding offspring. The second important difference is that the geometric description of a search operator is representation-independent and refers only indirectly to the specific solution representation via a distance defined on such representation (i.e. edit distances such as the Hamming distance which can be defined on the binary string representation as the minimum number of bit-flips to obtain one string from the other). In contrast, the algebraic definition of a search operator is representation-dependent and uses operations which are well-defined on the specific solution representation but that may not be well-defined on other representations (e.g. bit-flip on a binary string is not well-defined on a permutation).

The duality of the geometric search operators has surprising and important consequences [6]. One of them is the possibility of principled generalization of search algorithms for continuous spaces to combinatorial spaces, as sketched in the following. Given a search algorithm defined on continuous spaces, recast the definition of the search operators expressing them explicitly in terms of Euclidean distance between parents and offspring. Substitute the Euclidean distance with a generic metric, obtaining a formal search algorithm generalizing the original algorithm based on the continuous space. 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. Use this geometric and declarative description of the search operator to derive its algebraic and operational definition in terms of manipulation of the underlying representation. As mentioned in the introduction, we applied this methodology to generalize PSO to any metric space and derived the specific search operators for a number of representations. In the following sections, we use it to generalize DE. This methodology can be used to generalize to combinatorial spaces other algorithms naturally based on a notion of distance. This includes search algorithms such as Response Surface Methods, Estimation of Distribution Algorithms and Lipschitz Optimization algorithms, and also Machine Learning algorithms.

3. CLASSIC DIFFERENTIAL EVOLUTION

In this section, we describe the traditional DE [12] (see algorithm 1).

The characteristic that sets DE apart from other evolutionary algorithms is the presence of the differential mutation operator (see line 5 of algorithm 1). This operator creates a mutant vector U by perturbing a vector $X3$ picked at random from the current population with the scaled difference of other two randomly selected population vectors $F \cdot (X1 - X2)$. This operation is understood being important because it adapts the mutation direction and its step size to the level of convergence and spatial distribution of the

Algorithm 1 DE with differential mutation and discrete crossover

```
initialize population of N_p real vectors at random
while stop criterion not met do
  for all vector X(i) in the population do
    pick at random 3 distinct vectors from the current population X1, X2, X3
    create mutant vector U = X3 + F·(X1 − X2) where F is the scale factor parameter
    set V as the result of the discrete recombination of U and X(i) with probability Cr
    if f(V) ≥ f(X(i)) then
      set the i^th vector in the next population Y(i) = V
    else
      set Y(i) = X(i)
    end if
  end for
  for all vector X(i) in the population do
    set X(i) = Y(i)
  end for
end while
```

current population. The mutant vector is then recombined with the currently considered vector $X(i)$ using discrete recombination and the resulting vector V replaces the current vector in the next population if it has better fitness.

The differential mutation parameter F, known as scale factor, is a positive real normally between 0 and 1, but it can take also values greater than 1. The recombination probability parameter Cr takes values in $[0, 1]$. It is the probability, for each position in the vector $X(i)$, of the offspring V inheriting the value of the mutant vector U. When $Cr = 1$, the algorithm 1 degenerates to a DE algorithm with differential mutation only (because $V = U$). When $F = 0$, the algorithm 1 degenerates to a DE algorithm with discrete crossover only, as $U = X3$. The population size N_p normally varies from 10 to 100.

4. GEOMETRIC DIFFERENTIAL EVOLUTION

Following the methodology outlined in section 2, in this section we generalize the classic DE algorithm to general metric spaces. To do this, we recast differential mutation and discrete recombination as functions of the distance of the underlying search space, thereby obtaining their abstract geometric definitions. Then, in section 5, we derive the specific DE for the Hamming space associated with binary strings by plugging this distance in the abstract definition of the search operators.

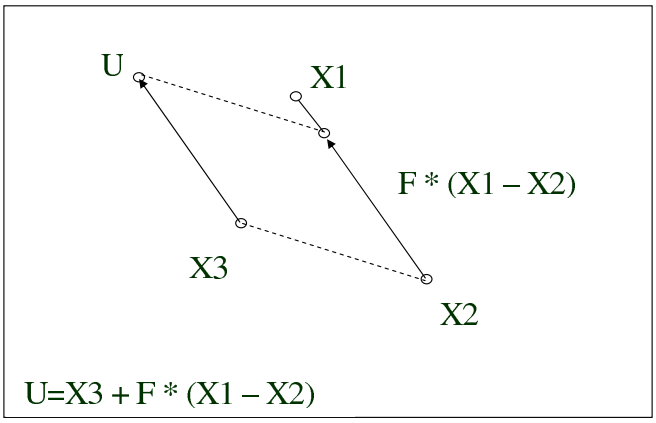

Figure 1: Construction of U using vectors.

4.1 Generalization of differential mutation

Let $X1, X2, X3$ be real vectors and $F \geq 0$ a scalar. The differential mutation operator produces a new vector U as follows:

$$U = X3 + F \cdot (X1 - X2) \tag{1}$$

The algebraic operations on real vectors in equation 1 can be represented graphically [12] as in figure 1. Real vectors are represented as points. The term $X1 - X2$ is represented as a vector originating in $X2$ and reaching $X1$. The multiplication with the scaling factor F produces a vector with the same origin and direction but with a different length. The addition of $X3$ to the scaled vector corresponds to the translation of the origin of the scaled vector from $X2$ to $X3$ keeping invariant its direction and length. The point of the (graphical) vector so obtained corresponds to the real vector U.

Unfortunately, this graphical interpretation of equation 1 in terms of operations on vectors does not help us to generalize equation 1 to general metric spaces because the notions of vector and operations on vectors are not well-defined at this level of generality. In the following, we propose a generalization based on interpreting equation 1 in terms of segments and extension rays, which are geometric elements well-defined on any metric space. To do that, we need to rewrite equation 1 in terms of only convex combinations of two vectors, which are the algebraic dual of segments. A convex combination of a set of vectors is a linear combination of these vectors provided that their weights are all positive and sum up to one.

Equation 1 can be rewritten as:

$$U + F \cdot X2 = X3 + F \cdot X1 \tag{2}$$

By dividing both sides by $1 + F$ and letting $W = \frac{1}{1+F}$ we have:

$$W \cdot U + (1 - W) \cdot X2 = W \cdot X3 + (1 - W) \cdot X1 \tag{3}$$

Both sides of equation 3 are convex combinations of two vectors. On the left-hand side, the vectors U and $X2$ have coefficients W and $1 - W$, respectively. These coefficients sum up to one and are both positive because $W \in [0, 1]$ for $F \geq 0$. Analogously, the right-hand side is a convex combination of the vectors $X3$ and $X1$ with the same coefficients.

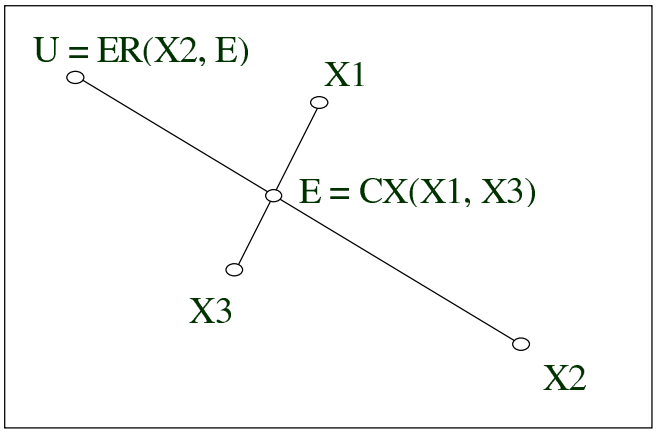

Figure 2: Construction of U using convex combination and extension ray.

There is a interesting duality between the algebraic notion of convex combination of two vectors and the geometric notion of segment in the Euclidean space. Vectors represent points in space. The points P_C corresponding to the vectors C obtained by any convex combination of two vectors A and B lay in the line segment between their corresponding points P_A and P_B. The vice versa also holds true: the vector C corresponding to a point P_C in the segment $[P_A, P_B]$ can be obtained as a convex combination of the vectors A and B. The weights W_A and W_B in the convex combination localize the point on the segment $[P_A, P_B]$: distances to P_C from P_A and P_B are inversely proportional to the corresponding weights, W_A and W_B. So, the weight of a vector can be thought geometrically as attraction force towards its corresponding point.

This duality allows for a geometric interpretation of equation 3 in terms of convex combinations (see figure 2). Let us call E the vector obtained by the convex combinations on both sides of equation 3. Geometrically the point E must be the intersection point of the segments $[U, X2]$ and $[X1, X3]$. The distances from E to the endpoints of these segments can be determined from equation 3 as they are inversely proportional to their respective weights. Since the point U is unknown (but its weight is known), it can be determined geometrically by firstly determine E as convex combination of $X1$ and $X3$; then, by projecting $X2$ beyond E (extension ray) obtaining a point U such that the proportions of the distances of $X2$ and U to the point E is inversely proportional to their weights. In the Euclidean space, the constructions of U using vectors (figure 1) and convex combinations (figure 2) are equivalent (algebraically, hence geometrically).

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. We will present their abstract definitions in section 5 before specifying these operators for the Hamming distance on binary strings.

The differential mutation operator $U = DM(X1, X2, X3)$ with scale factor F can now be defined for any metric space following the construction of U presented in figure 2 as follows:

1. Compute $W = \frac{1}{1+F}$
2. Get E as the convex combination $CX(X1, X3)$ with weights $(1 - W, W)$ (generalizing $E = (1 - W) \cdot X1 + W \cdot X3$)
3. Get U as the extension ray $ER(X2, E)$ with weights $(W, 1 - W)$ (generalizing $U = (E - (1 - W) \cdot X2)/W$)

4.2 Generalization of discrete recombination

After applying differential mutation, the DE algorithm applies discrete recombination to U and $X(i)$ generating V. Discrete recombination is a geometric crossover under Hamming distance for real vectors [6]. The Hamming distance (HD) for real vectors is defined analogously to the Hamming distance between binary strings: it is the number of sites with mismatching values across the two vectors. From its definition, we can derive that the Cr parameter of the discrete recombination is proportional to the expected number of values that V inherits from U. Therefore, $E[HD(U, V)] = Cr \cdot HD(U, X(i))$ and $E[HD(X(i), V)] = (1-Cr) \cdot HD(U, X(i))$. Consequently, Cr and $1 - Cr$ can be interpreted as the weights of U and $X(i)$, respectively, of the convex combination that returns V in the space of real vectors endowed with Hamming distance. In order to generalize the discrete recombination, by replacing hamming distance with a generic metric, we obtain the abstract convex combination operator CX introduced in the previous section. So, we have that the generalized discrete recombination of U and $X(i)$ with probability parameter Cr generating V is as follows: $V = CX(U, X(i))$ with weights $(Cr, 1 - Cr)$.

In the classic DE (algorithm 1), replacing the original differential mutation and discrete recombination operators with their generalizations, we obtain the formal Geometric Differential Evolution (see algorithm 2). When this formal algorithm is specified on the Euclidean space, the resulting Euclidean GDE does *not* coincide with the classic DE. This is because, whereas the original differential mutation operator can be expressed as a function of the Euclidean distance, the original discrete recombination operator can be expressed as a function of the Hamming distance for real vectors, not of the Euclidean distance. The Euclidean GDE coincides with an existing variant of traditional DE [12], which has the same differential mutation operator but in which the discrete recombination is replaced with blend crossover. Interestingly, blend crossover lives in the same space as differential mutation and their joint behavior has a geometric interpretation in space.

5. BINARY GDE

In this section, we present definitions of convex combination and extension ray and their weighted extensions in general metric spaces, and derive formally their specifications to the Hamming space for binary strings. These specific operators can be plugged in the formal GDE (algorithm 2) to obtain a specific GDE for the Hamming space, the Binary GDE.

Algorithm 2 Formal Geometric Differential Evolution

initialize population of N_p configurations at random
while stop criterion not met **do**
  for all configuration $X(i)$ in the population **do**
    pick at random 3 distinct configurations from the current population $X1, X2, X3$
    set $W = \frac{1}{1+F}$ where F is the scale factor parameter
    create intermediate configuration E as the convex combination $CX(X1, X3)$ with weights $(1-W, W)$
    create mutant configuration U as the extension ray $ER(X2, E)$ with weights $(W, 1-W)$
    create candidate configuration V as the convex combination $CX(U, X(i))$ with weights $(Cr, 1-Cr)$ where Cr is the recombination parameter
    if $f(V) \geq f(X(i))$ **then**
      set the i^{th} configuration in the next population $Y(i) = V$
    else
      set $Y(i) = X(i)$
    end if
  end for
  for all configuration $X(i)$ in the population **do**
    set $X(i) = Y(i)$
  end for
end while

5.1 Convex combination

The notion of convex combination in metric spaces was introduced in the GPSO framework [7]. 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 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. In the Euclidean space, C 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 specified to Hamming spaces on binary strings, we obtain the recombination operator outlined in algorithm 3 [7]. This algorithm returns offspring C such that $d(A, C)/d(B, C) = W_B/W_A$ in expectation. This differs from the Euclidean case where this ratio is guaranteed.

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) $\leq W_A$ **then**
    set $C(i)$ to $A(i)$
  else
    set $C(i)$ to $B(i)$
  end if
end for
return string C as offspring

5.2 Extension ray

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 $x, y \in S$ are called end-points of the segment. 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 extension ray in a metric space can be defined indirectly using metric segments, as follows.

DEFINITION 1. *Given points A and B, the (metric) extension ray $ER(A, B)$ is the set of points C that satisfy $C \in [A, B]$ or $B \in [A, C]$.*

In this paper, only the part of the extension ray beyond B will be of interest because the point C that we want to determine, which is, the offspring of the differential mutation operator, is never between A and B by construction.

In the following, we define the weighted extension ray for general metric spaces consistently with the geometric interpretation presented in section 4.1. The weighted extension ray ER is defined as the inverse operation of the weighted convex combination CX, as follows.

DEFINITION 2. *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.*

Notice that from this definition follows that the weights w_{ab} and w_{bc} in ER are positive real numbers between 0 and 1 and sum up to 1 because they must respect this condition in CX.

The set of points returned by the weighted extension ray ER can be characterized in terms of distances to the input points of ER, as follows.

LEMMA 1. *The points C returned by the weighted extension ray $ER((A, w_{ab}), (B, w_{bc}))$ are those points which are at a distance $d(A, B) \cdot w_{ab}/w_{bc}$ from B and at a distance $d(A, B)/w_{bc}$ from A.*

PROOF. From the definition of weighted extension ray we have that $B = CX((A, w_{ab}), (C, w_{bc})))$. Hence, $d(A, C) = d(A, B) + d(B, C)$ and the distances $d(A, B)$ and $d(B, C)$ are inversely proportional to the weights w_{ab} and w_{bc}. Consequently, $d(A, C) = d(A, B)/w_{bc}$ and substituting it in $d(B, C) = d(A, C) - d(A, B)$ we get $d(B, C) = d(A, B) \cdot w_{ab}/w_{bc}$, since $w_{ab} + w_{bc} + 1$. □

This characterization may be useful to construct procedures to implement the weighted extension ray for specific spaces. We used it, together with representation-specific properties of the extension ray in the Hamming space on binary strings, in the derivation of the Binary Extension Ray Recombination (algorithm 4).

In order to gain an intuitive understanding of how an extension ray looks like in the Hamming space, let us consider

an example of extension ray originating in $A = 110011$ and passing through $B = 111001$.

The relation $C \in [A, B]$ is satisfied by those C that match the schema $S1 = 11 * 0 * 1$. This is the set of the possible offspring of A and B that can be obtained by recombining them using the uniform crossover.

The relation $B \in [A, C]$ is satisfied by all those C that match $S2 = * * 1 * 0*$. This is the set of all those C that when recombined with A using the uniform crossover can produce B as offspring.

The following theorem characterizes the extension ray in the Hamming space in terms of schemata.

THEOREM 2. *Let A and B be fixed binary strings in the Hamming space:*

1. *the relation $C \in [A, B]$ is satisfied by those strings C that match the schema obtained by keeping the common bits in A and B and inserting $*$ where the bits of A and B do not match.*
2. *the relation $B \in [A, C]$ is satisfied by all those strings C that match the schema obtained by inserting $*$ where the bits are common in A and B and inserting the bits coming from B where the bits of A and B do not match.*

PROOF. *Proof of statement 1*: the schema so defined corresponds to the set of the possible offspring of A and B that can be obtained by recombining them using the uniform crossover. This crossover operator corresponds to the uniform geometric crossover under Hamming distance which returns offspring in the segment between parents.

Proof of statement 2: all C matching the schema S defined in this statement recombined with A can produce B as offspring. This is because at each position (in A, B and C) when in the schema S there is $*$ the bit in B at that position can be inherited from A. When in the schema there is a bit (0 or 1) the bit in B at that position can be inherited from C. Furthermore, only the strings C matching S can produce B when C is recombined with A. □

Using the characterization of the weighted extension ray in terms of distances (lemma 1) and the characterization of the extension ray in the Hamming space in terms of schemata (theorem 2), we were able to derive the weighted extension ray recombination for this space (see algorithm 4). Theorem 3 proves that this recombination operator conforms to the definition of weighted extension ray for the Hamming space.

THEOREM 3. *Given parents A and B, the recombination in algorithm 4 returns an offspring C such that $E[HD(B,C)]/HD(A,B) = W_{AB}/W_{BC}$, where $E[HD(B,C)]$ is the expected Hamming distance between B and the offspring C. Therefore, in expectation, this recombination operator conforms to the geometric definition of weighted extension ray under Hamming distance.*

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) · 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) ≤ p then
        set C(i) to the complement of B(i)
    end if
end for
return string C as offspring
```

PROOF. This can be shown as follows. The number of bits in which A and B differ are $HD(A, B)$. The number of bits in which A and B do not differ is $n - HD(A, B)$. For the bits in which A and B differ, the string C equals B. For each bit in which A and B do not differ, C does not equal B with probability p. So, the expected distance between B and C is $E[HD(B, C)] = (n - HD(A, B)) \cdot p$. By substituting $p = HD(B, C)/(n - HD(A, B))$, we have $E[HD(B, C)] = HD(B, C) = HD(A, B) \cdot W_{AB}/W_{BC}$. So, $E[HD(B, C)]/HD(A, B) = W_{AB}/W_{BC}$. □

6. EXPERIMENTS

We implemented the GDE algorithm for binary spaces within a Java framework,[1] and investigated its performance on some benchmark problems. The proposed algorithm was compared with three other algorithms:

- cGA: A canonical Genetic Algorithm, with roulette wheel fitness-proportionate selection, uniform crossover and bitflip mutation.
- tGA: a Genetic Algorithm with truncation selection, with a selection threshold of $popsize/2$.
- ES: A $\mu+\lambda$ Evolution Strategy, with $\mu = \lambda = popsize/2$ and bitflip mutation.

For the first benchmark suite, we also compared it with:

- BPSO: Discrete Binary PSO of Kennedy and Eberhart, using the results presented in [3].

For the ES and GAs, the bitflip mutation works as follows: each bit in the chromosome is considered, and with probability p this bit is flipped. In the experiments involving these algorithms, this parameter was systematically varied between 0.0 and 0.5 in increments of 0.01. For the experiments involving GDE, the key parameters F and Cr were systematically varied between 0.0 and 1.0 in increments of 0.1.

[1]Source code is available upon request from the second author.

In all experiments, the length of any single run was set to 4000 function evaluations, in order to be directly comparable with the results of Kennedy and Eberhart. For GDE, GA and ES the population size was varied systematically: sizes of 10, 20, 40, 80 and 160 were tried, with the numbers of generations limited appropriately: 400, 200, 100, 50 and 25.

6.1 Spears-DeJong functions

We used three of the same benchmark problems that Kennedy and Eberhart tested their binary PSO on. These are William Spears' binary versions of DeJong's functions $f1$, $f2$ and $f3$.[2] (We did not use $f4$ and $f5$ due to unresolved differences between different versions of the code, which might be due to differing numerical precision in different systems; further, Kennedy and Eberhart do not report precise results for $f4$.)

Each configuration (parameters and population size) was tested twenty times, and the average best score of each run was recorded, as well as how many of the runs that reached the global optimum. The results are summarized in table 1. The parameters were optimized separately for each combination of benchmark function and algorithm, and only the results of the best configuration are reported here. The best parameter settings found are reported in tables 2 and 3.

The GDE algorithm appears to work best with small population sizes, 10 or 20 individuals (and thus more generations). On $f1$, there is a clear preference for high values (e.g. 0.9) of both F and Cr, whereas on $f2$ the algorithm seems to work best with values of around 0.3 for both parameters.

In comparison, both GAs always work best with large populations and relatively high mutation rates (>0.1). The ES seems to be relatively insensitive to population size, as long as the mutation rate is in the region 0.05–0.1.

The compressed fitness structure of $f1$ and $f2$, with many local optima with values differing from the global optimum only in the third decimal, is apparently a bad match with fitness-proportional selection; the landscape of $f3$ has similar characteristics but to a lesser degree. Therefore, the results of the canonical GA are the worst on all problems.

Both the evolution strategy and the GA with truncation selection are strictly better on all benchmarks than the canonical GA; binary PSO is better than ES in that it reaches optimum more often on $f1$ and $f2$; and GDE is the best algorithm overall, as it is as good as BPSO on $f2$ and $f3$ but reaches the global optimum almost twice as often on $f1$.

6.2 NK Landscapes

In order to more systematically test the behaviour of GDE on landscapes with varying amount of epistasis, we performed additional experiments using NK fitness landscapes, as proposed by Kauffman [2]. NK landscapes have two paramters: N, the number of dimensions, was fixed to 100 in our experiments; K, the number of dependencies on other loci per locus was varied between 0 and 4. The parameters of the algorithms (mutation rate, F and Cr) were varied in the same way as with the Spears-DeJong experiments above. All evolutionary runs lasted for 10000 function evaluations, which were allocated either as population size 100 and 100 generations or as population size 10 and 1000 generations.

[2]The original c source code of these functions can be found at http://www.cs.uwyo.edu/∼wspears/functs/dejong.c

Algorithm	$f1$ (78.6)		$f2$ (3905.93)		$f3$ (55.0)	
BPSO	-	10	-	4	-	20
GDE	78.5999	19	3905.9296	4	55.0	20
cGA	78.2152	0	3905.8052	0	52.1	1
tGA	78.5993	4	3905.9266	2	55.0	20
ES	78.5998	7	3905.9291	2	55.0	20

Table 1: Results on the Spears-DeJong benchmark suite. The maxima of the functions are reported next to their names. For each combination of algorithm and problem, the results of the best parameterization of that combination are reported. The first number is the best fitness of the last generation, averaged over 20 runs. The second number is the number of those runs that reached the global optimum.

Function	pop/gen	F	Cr
$f1$	10/400	0.9	0.8
$f2$	20/200	0.3	0.3
$f3$	*	*	*

Table 2: Best parameter settings found for GDE on the Spears-DeJong benchmarks. The asterisks denote that many combinations are optimal.

The results in table 4 show that GDE is a very competitive algorithm overall. For population size 100, GDE is the best of the four algorithms for K of 1, 2 and 3, and a close second for K of 0 and 4. For population size 10, GDE is the best algorithm for all K except $K = 0$. The results further show that the ES and the GA with truncation selection performs significantly better than the canonical GA for all K.

Table 5 shows the best parameter settings for GDE for different K. Apparently, for low K larger population sizes are preferred, and for higher K smaller populations do better. Interestingly, for all K the best configuration is very low F and medium to high Cr. Table 6 presents the best parameter settings found for ES and GA. A very clear trend is that ES works best with small populations and bots GAs with larger populations; ES also generally prefers lower mutation rate than the GAs.

7. CONCLUSIONS

In this paper, we have generalized DE from continuous to generic combinatorial spaces by extending the geometric interpretation of the classic DE to general metric spaces. The algorithm obtained (GDE) can then be formally specified for specific spaces and specific representations. We have illustrated this by deriving the specific GDE for the Hamming space associated with binary strings. The binary GDE compared on a standard benchmark against a set of classic evolutionary algorithms performed best in the comparison.

In future work, we will extensively test the binary GDE on a larger set of benchmark functions. Also, we will specify the

cGA	pop/gen	mutation
$f1$	160/25	0.12
$f2$	160/25	0.16
$f3$	80/50	0.39
tGA	pop/gen	mutation
$f1$	80/50	0.29
$f2$	80/50	0.1
$f3$	80/50	0.45
ES	pop/gen	mutation
$f1$	10/400	0.13
$f2$	160/25	0.1
$f3$	*	*

Table 3: Best parameter settings found for GA (with truncation and roulette-wheel selection) and ES on the Spears-DeJong benchmarks. The asterisks denote that many combinations are optimal.

10	$K=0$	$K=1$	$K=2$	$K=3$	$K=4$	$K=5$
GDE	0.623	0.730	0.732	0.758	0.751	0.741
cGA	0.521	0.509	0.515	0.536	0.519	0.517
tGA	0.597	0.621	0.613	0.621	0.641	0.641
ES	0.667	0.721	0.746	0.740	0.736	0.727
100	$K=0$	$K=1$	$K=2$	$K=3$	$K=4$	$K=5$
GDE	0.665	0.750	0.738	0.756	0.736	0.719
cGA	0.552	0.594	0.613	0.610	0.600	0.610
tGA	0.664	0.707	0.713	0.736	0.737	0.730
ES	0.677	0.696	0.710	0.717	0.717	0.720

Table 4: Results on the NK landscape benchmark. Average maximum fitness at the last generation for each algorithm using K values between 0 and 5, using population sizes of both 10 and 100. 50 runs were performed for each configuration.

formal GDE for search spaces associated with permutations and test it on hard combinatorial optimization problems. Differential Evolution 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. Nelder and Mead method and Controlled Random Search method). We will use the same technique to generalize these algorithms to general metric spaces.

Acknowledgements

We would like to thank William Spears for his generous assistance in adapting and debugging the code for the De Jong benchmark functions to Java. Additionally, we would like to thank one of the anonymous reviewers for her/his very detailed and helpful review.

8. REFERENCES

[1] T. Gong and A. L. Tuson, *Differential evolution for binary encoding*, Soft Computing in Industrial Applications, Springer, 2007, pp. 251–262.

[2] Stuart Kauffman, *Origins of order: Self-organization and selection in evolution*, Oxford University Press, 1993.

[3] 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.

[4] ———, *Swarm intelligence*, Morgan Kaufmann, 2001.

[5] Eugene F. Krause, *Taxicab geometry: An adventure in non-euclidean geometry*, Courier Dover Publications, 1986.

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

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

[8] 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.

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

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

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

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

[13] G. Sywerda, *Uniform crossover in genetic algorithms*, Proceedings of the third international conference on Genetic algorithms, 1989.

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

K	pop/gen	F	Cr
0	100/100	0.0	0.8
1	100/100	0.0	0.7
2	100/100	0.0	0.5
3	10/1000	0.1	0.9
4	10/1000	0.1	0.8
5	10/1000	0.1	0.8

Table 5: Best parameter settings found for GDE on the NK landscape benchmark.

K	cGA	tGA	ES
0	0.01	0.35	0.01
1	0.01	0.43	0.03
2	0.28	0.47	0.03
3	0.16	0.19	0.04
4	0.39	0.36	0.02
5	0.20	0.30	0.02

Table 6: Best mutation settings found for GA and ES on the NK landscape benchmark. The GAs always performed best with population size 100, and the ES with population size 10.