Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 143624, 14 pages
doi:10.1155/2008/143624

Research Article

Geometric Particle Swarm Optimization

Alberto Moraglio,[1] Cecilia Di Chio,[2] Julian Togelius,[3] and Riccardo Poli[2]

[1] *Centre for Informatics and Systems of the University of Coimbra, Polo II - University of Coimbra, 3030-290 Coimbra, Portugal*
[2] *Department of Computing and Electronic Systems, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK*
[3] *Dalle Molle Institute for Artificial Intelligence (IDSIA), Galleria 2, 6928 Manno-Lugano, Switzerland*

Correspondence should be addressed to Alberto Moraglio, moraglio@dei.uc.pt

Received 21 July 2007; Accepted 4 December 2007

Recommended by T. Blackwell

Using a geometric framework for the interpretation of crossover of recent introduction, we show an intimate connection between particle swarm optimisation (PSO) and evolutionary algorithms. This connection enables us to generalise PSO to virtually any solution representation in a natural and straightforward way. The new geometric PSO (GPSO) applies naturally to both continuous and combinatorial spaces. We demonstrate this for the cases of Euclidean, Manhattan, and Hamming spaces and report extensive experimental results. We also demonstrate the applicability of GPSO to more challenging combinatorial spaces. The Sudoku puzzle is a perfect candidate to test new algorithmic ideas because it is entertaining and instructive as well as being a nontrivial constrained combinatorial problem. We apply GPSO to solve the Sudoku puzzle.

1. INTRODUCTION

Particle swarm optimization (PSO) is a relatively recently devised population-based stochastic global optimization algorithm [1]. PSO has many similarities with evolutionary algorithms, and has also proven to have robust performance over a variety of difficult optimization problems. However, the original formulation of PSO requires the search space to be continuous and the individuals to be represented as vectors of real numbers.

There is a number of extensions of PSO to combinatorial spaces with various degrees of success [2, 3]. (Notice that applications of traditional PSO to combinatorial optimization problems cast as continuous optimization problems are not extensions of the PSO algorithm.) However, every time a new solution representation is considered, the PSO algorithm needs to be rethought and adapted to the new representation. In this article, we extend PSO to richer spaces by making use of a rigorous mathematical generalization of the notion (and motion) of particles to a general class of spaces. This approach has the advantage that a PSO can be derived in a principled way for any search space belonging to the given class.

In particular, we show *formally* how a general form of PSO (without the inertia term) can be obtained by using theoretical tools developed for evolutionary algorithms with geometric crossover and geometric mutation. These are representation-independent operators that generalize many pre-existing search operators for the major representations, such as binary strings [4], real vectors [4], permutations [5], syntactic trees [6], and sequences [7]. (The inertia weight was not part of the original proposal of PSO, it was later introduced by Shi and Eberhart [8].)

Firstly, we formally derive geometric PSOs (GPSOs) for Euclidean, Manhattan, and Hamming spaces and discuss how to derive GPSOs for virtually any representation in a similar way. Then, we test the GPSO theory experimentally: we implement the specific GPSO for Euclidean, Manhattan, and Hamming spaces and report extensive experimental results showing that GPSOs perform very well.

Finally, we also demonstrate that GPSO can be specialized easily to nontrivial combinatorial spaces. In previous work [9], we have used the geometric framework to design an evolutionary algorithm to solve the Sudoku puzzle and obtained very good experimental results. Here, we apply GPSO to solve the Sudoku puzzle.

In Section 2, we introduce the geometric framework and introduce the notion of multiparental geometric crossover. In Section 3, we recast PSO in geometric terms and generalize it to generic metric spaces. In Section 4, we apply these

notions to the Euclidean, Manhattan, and Hamming spaces. In Section 5, we discuss how to specialize the general PSO automatically to virtually any solution representation using geometric crossover. Then, in Section 6, we report experimental results with the GPSOs for Euclidean, Manhattan, and Hamming spaces, and we compare them with a traditional PSO. In Section 7, we apply GPSO to Sudoku, and we describe the results in Section 8. Finally, in Section 9, we present conclusions and future work.

2. GEOMETRIC FRAMEWORK

Geometric operators are defined in geometric terms using the notions of line segment and ball. These notions and the corresponding genetic operators are well defined once a notion of distance in the search space is defined. Defining search operators as functions of the search space is opposite to the standard way [10] in which the search space is seen as a function of the search operators employed.

2.1. Geometric preliminaries

In the following, we give necessary preliminary geometric definitions and extend those introduced in [4]. For more details on these definitions, see [11].

The terms *distance* and *metric* denote any real valued function that conforms to the axioms of identity, symmetry, and triangular inequality. A simple connected graph is naturally associated to a metric space via its *path metric*: the distance between two nodes in the graph is the length of a shortest path between the nodes. Distances arising from graphs via their path metric are called *graphic distances*. Similarly, an edge-weighted graph with strictly positive weights is naturally associated to a metric space via a *weighted path metric*.

In a metric space (S, d), a *closed ball* is a set of the form $B(x; r) = \{y \in S \mid d(x, y) \leq r\}$, where $x \in S$ and r is a positive real number called the radius of the ball. A *line segment* is a set of the form $[x; y] = \{z \in S \mid d(x, z) + d(z, y) = d(x, y)\}$, where $x, y \in S$ are called extremes of the segment. Metric ball and metric segment generalize the familiar notions of ball and segment in the Euclidean space to any metric space through distance redefinition. In general, there may be more than one shortest path (*geodesic*) connecting the extremes of a metric segment: the metric segment is the union of all geodesics.

We assign a structure to the solution set by endowing it with a notion of distance d. $M = (S, d)$ is, therefore, a solution *space* and $L = (M, g)$, where g is the fitness function, is the corresponding *fitness landscape*.

2.2. Geometric crossover

Definition 1 (geometric crossover). A binary operator is a geometric crossover under the metric d if all offsprings are in the segment between its parents.

The definition is *representation-independent* and, therefore, crossover is well defined for any representation. Being based on the notion of metric segment, *crossover is a function only of the metric d* associated with the search space.

The class of geometric crossover operators is very broad. For vectors of reals, various types of blend or line crossovers, box recombinations, and discrete recombinations are geometric crossovers [4]. For binary and multary strings, all homologous crossovers are geometric [4, 12]. For permutations, PMX, Cycle crossover, merge crossover, and others are geometric crossovers [5]. We describe this in more detail in Section 2.3 since we will use the permutation representation in this paper. For syntactic trees, the family of homologous crossovers are geometric [6]. Recombinations for several more complex representations are also geometric [4, 5, 7, 13].

2.3. Geometric crossover for permutations

In previous work, we have studied various crossovers for permutations, revealing that PMX [14], a well-known crossover for permutations, is geometric under swap distance. Also, we found that Cycle crossover [14], another traditional crossover for permutations, is geometric under swap distance and under Hamming distance (geometricity under Hamming distance for permutations implies geometricity under swap distance, but not *vice versa*). Finally, we showed that geometric crossovers for permutations based on edit moves are naturally associated with sorting algorithms: picking offspring on a minimum path between two parents corresponds to picking partially sorted permutations on the minimal sorting trajectory between the parents.

2.4. Geometric crossover landscape

Geometric operators are defined as functions of the distance associated with the search space. However, the search space does not come with the problem itself. The problem consists of a fitness function to optimize and a solution set, but not a neighbourhood relationship. The act of putting a structure over the solution set is part of the search algorithm design and it is a designer's choice. A fitness landscape is the fitness function plus a structure over the solution space. So for each problem, there is one fitness function but as many fitness landscapes as the number of possible different structures over the solution set. In principle, the designer could choose the structure to assign to the solution set completely independently from the problem at hand. However, because the search operators are defined over such a structure, doing so would make them decoupled from the problem at hand, hence turning the search into something very close to random search.

In order to avoid this, one can exploit problem knowledge in the search. This can be achieved by carefully designing the connectivity structure of the fitness landscape. For example, one can study the objective function of the problem and select a neighbourhood structure that couples the distance between solutions and their fitness values. Once this is done, the problem knowledge can be exploited by search operators to perform better than random search, even if the

search operators are problem independent (as in the case of geometric crossover and mutation).

Under which conditions is a landscape well searchable by geometric operators? As a rule of thumb, geometric mutation and geometric crossover work well on landscapes where the closer pairs of solutions are, the more correlated their fitness values. Of course this is no surprise: the importance of landscape smoothness has been advocated in many different contexts and has been confirmed in uncountable empirical studies with many neighborhood search metaheuristics [15, 16]. To summarize, consider the following.

(i) Rule of thumb 1: If we have a good distance for the problem at hand, then we have good geometric mutation and good geometric crossover.
(ii) Rule of thumb 2: A good distance for the problem at hand is a distance that makes the landscape "smooth."

2.5. Product geometric crossover

In recent work [12], we have introduced the notion of product geometric crossover.

Theorem 1. *Cartesian product of geometric crossover is geometric under the sum of distances.*

This theorem is very useful because it allows one to build new geometric crossovers by combining crossovers that are known to be geometric. In particular, this applies to crossovers for mixed representations. The elementary geometric crossovers do not need to be independent, to form a valid product geometric crossover.

2.6. Multiparental geometric crossover

To extend geometric crossover to the case of multiple parents, we need the following definitions [17].

Definition 2. A family $\mathcal{X}$ of subsets of a set X is called *convexity on* X if

(C1) the empty set $\varnothing$ and the universal set X are in $\mathcal{X}$,
(C2) $\mathcal{D} \subseteq \mathcal{X}$ is nonempty, then $\bigcap \mathcal{D} \in \mathcal{X}$, and
(C3) $\mathcal{D} \subseteq \mathcal{X}$ is nonempty and totally ordered by inclusion, then $\bigcup \mathcal{D} \in \mathcal{X}$.

The pair $(X, \mathcal{X})$ is called *convex structure*. The members of $\mathcal{X}$ are called *convex sets*. By the axiom (C1), a subset A of X of the convex structure is included in at least one convex set, namely, X.

From axiom (C2), A is included in a smallest convex set, the *convex hull* of A:

$$\mathrm{co}(A) = \bigcap \{C \mid A \subseteq C \in \mathcal{X}\}. \tag{1}$$

The convex hull of a finite set is called a *polytope*.

The axiom (C3) requires *domain finiteness* of the convex hull operator: a set C is convex if it includes $\mathrm{co}(F)$ for each finite subset F of C.

The convex hull operator applied to a set of cardinality two is called *segment operator*. Given a metric space $M = (X, d)$, the segment between a and b is the set $[a, b]_d = \{z \in X \mid d(x, z) + d(z, y) = d(x, y)\}$. The abstract *geodetic convexity* $\mathcal{C}$ on X induced by M is obtained as follows: a subset C of X is geodetically convex, provided $[x, y]_d \subseteq C$ for all x, y in C. If co denotes the convex hull operator of $\mathcal{C}$, then for all $a, b \in X : [a, b]_d \subseteq \mathrm{co}\{a, b\}$. The two operators need not to be equal: there are metric spaces in which metric segments are not all convex.

We can now provide the following extension.

Definition 3 (multiparental geometric crossover). In a multiparental geometric crossover, given n parents $p_1, p_2, \ldots, p_n$, their offspring are contained in the metric convex hull of the parents $\mathrm{co}(\{p_1, p_2, \ldots, p_n\})$ for some metric d.

Theorem 2 (decomposable three-parent recombination). *Every multiparental recombination $RX(p_1, p_2, p_3)$ that can be decomposed as a sequence of 2-parental geometric crossovers under the same metric GX and GX′, so that $RX(p_1, p_2, p_3) = GX(GX'(p_1, p_2), p_3)$, is a three-parental geometric crossover.*

Proof. Let P be the set of parents and $\mathrm{co}(P)$ their metric convex hull. By definition of metric convex hull, for any two points $a, b \in \mathrm{co}(P)$, their offspring are in the convex hull $[a, b] \subseteq \mathrm{co}(P)$. Since $P \subseteq \mathrm{co}(P)$, any two parents $p_1, p_2 \in P$ have offspring $o_{12} \in \mathrm{co}(P)$. Then, any other parent $p_3 \in P$, when recombined with o_{12}, produces offspring o_{123} in the convex hull $\mathrm{co}(P)$. So the three-parental recombination equivalent to the sequence of geometric crossover $GX'(p_1, p_2) \to o_{12}$ and $GX(o_{12}, p_3) \to o_{123}$ is a multiparental geometric crossover. □

3. GEOMETRIC PSO

3.1. Canonical PSO algorithm and geometric crossover

Consider the canonical PSO in Algorithm 1. It is well known [18] that one can write the equation of motion of the particle without making explicit use of its velocity.

Let x be the position of a particle and v its velocity. Let $\hat{x}$ be the current best position of the particle and let $\hat{g}$ be the global best. Let v' and v'' be the velocity of the particle and $x' = x + v$ and $x'' = x' + v'$ its position at the next two time ticks. The equation of velocity update is the linear combination: $v' = \omega v + \phi_1(\hat{x} - x') + \phi_2(\hat{g} - x')$, where ω, ϕ_1, and ϕ_2 are scalar coefficients. To eliminate velocities, we substitute the identities $v = x' - x$ and $v' = x'' - x'$ in the equation of velocity update and rearrange it to obtain an equation that expresses x'' as function of x and x' as follows:

$$x'' = (1 + \omega - \phi_1 - \phi_2)x' - \omega x + \phi_1\hat{x} + \phi_2\hat{g}. \tag{2}$$

If we set $\omega = 0$ (i.e., the particle has no inertia), x'' becomes independent on its position x two time ticks earlier. If we call $w_1 = 1 - \phi_1 - \phi_2$, $w_2 = \phi_1$, and $w_3 = \phi_2$, the equation of motion becomes

$$x'' = w_1x' + w_2\hat{x} + w_3\hat{g}. \tag{3}$$

In these conditions, the main feature that allows the motion of particles is the ability to perform linear combinations

(1) *for all* particle *i do*
(2) initialize position x_i and velocity v_i
(3) *end for*
(4) *while* stop criteria not met *do*
(5) *for all* particle *i do*
(6) set personal best $\hat{x}_i$ as best position found so far by the particle
(7) set global best $\hat{g}$ as best position found so far by the whole swarm
(8) *end for*
(9) *for all* particle *i do*
(10) update velocity using equation

$$v_i(t+1) = \kappa(\omega v_i(t) + \phi_1 U(0,1)(\hat{g}(t) - x_i(t)) + \phi_2 U(0,1)(\hat{x}_i(t) - x_i(t))),$$

where, typically, either ($\kappa = 0.729$, $\omega = 1.0$) or ($\kappa = 1.0$, $\omega < 1$)
(11) update position using equation

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

(12) *end for*
(13) *end while*

Algorithm 1: Standard PSO algorithm.

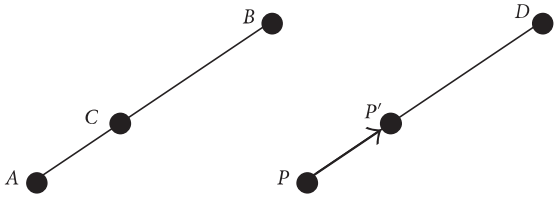

Figure 1: Geometric crossover and particle motion.

of points in the search space. As we will see in the next section, we can achieve this same ability by using multiple (geometric) crossover operations. This makes it possible to obtain a generalization of PSO to generic search spaces.

In the following, we illustrate the parallel between an evolutionary algorithm with geometric crossover and the motion of a particle in PSO (see Figure 1). Geometric crossover picks offspring C on a line segment between parents A and B. Geometric crossover can be interpreted as a motion of a particle: consider a particle P that moves in the direction of a point D reaching, in the next time step, position P'. If one equates parent A with the particle P and parent B with the direction point D, the offspring C is, therefore, the particle at the next time step P'. The distance between parent A and offspring C is the magnitude of the velocity of the particle P. Notice that the particle moves from P to P': this means that the particle P is replaced by the particle P' in the next time step. In other words, the new position of the particle replaces the previous position. Coming back to the evolutionary algorithm with geometric crossover, this means that the offspring C replaces its parent A in the new population. The fact that at a given time all particles move is equivalent to say that each particle is selected for "mating." Mating is a weighted multirecombination involving the memory of the particle and the best in the current population.

In the standard PSO, weights represent the propensity of a particle towards memory, sociality, and cognition. In the GPSO, they represent the attractions towards the particle's previous position, the particle's best position, and the swarm's best position.

Naturally, particle motion based on geometric crossover leads to a form of search that cannot extend beyond the convex hull of the initial population. Mutation can be used to allow nonconvex search. We explain these ideas in detail in the following sections.

3.2. Geometric interpretation of linear combinations

Definition 4. A *convex combination* of vectors $v_1, \ldots, v_n$ is a linear combination $a_1v_1 + a_2v_2 + a_3v_3 + \cdots + a_nv_n$, where all coefficients $a_1, \ldots, a_n$ are nonnegative and add up to 1.

It is called "convex combination" because when vectors represent points in space, the set of all convex combinations constitutes the convex hull.

A special case is $n = 2$, where a point formed by the convex combination will lie on a straight line between two points. For three points, their convex hull is the triangle with the points as vertices.

Theorem 3. *In a PSO with no inertia ($\omega = 0$) and where acceleration coefficients are such that $\phi_1 + \phi_2 \leq 1$, the next position x' of a particle is within the convex hull formed by its current position x, its local best $\hat{x}$, and the swarm best $\hat{g}$.*

Proof. As we have seen in Section 3.1, when $\omega = 0$, a particle's update equation becomes the linear combination in (3). Notice that this is an affine combination since the coefficients of x', $\hat{x}$, and $\hat{g}$ add up to 1. Interestingly, this means that the new position of the particle is coplanar with x', $\hat{x}$, and $\hat{g}$. If we restrict w_2 and w_3 to be positive and their sum to be less than 1, (3) becomes a convex combination. *Geometrically, this means that the new position of the particle is in the convex hull formed by (or more informally, is between) its previous position, its local best, and the swarm best.* □

(1) *for all* particle i *do*
(2) initialize position x_i at random in the search space
(3) *end for*
(4) *while* stop criteria not met *do*
(5) *for all* particle i *do*
(6) set personal best $\hat{x}_i$ as best position found so far by the particle
(7) set global best $\hat{g}$ as best position found so far by the whole swarm
(8) *end for*
(9) *for all* particle i *do*
(10) update position using a randomized convex combination

$$x_i = CX((x_i, w_1), (\hat{g}, w_2), (\hat{x}_i, w_3))$$

(11) mutate x_i
(12) *end for*
(13) *end while*

Algorithm 2: Geometric PSO algorithm.

In the next section, we generalize this simplified form of PSO from real vectors to generic metric spaces. As mentioned above, mutation will be required to extend the search beyond the convex hull.

3.3. Convex combinations in metric spaces

Linear combinations are well defined for vector spaces: algebraic structures endowed with scalar product and vectorial sum. A metric space is a set endowed with a notion of distance. The set underlying a metric space does not normally come with well-defined notions of scalar product and sum among its elements. Therefore, a linear combination of its elements is not defined. How can we then define a convex combination in a metric space? Vectors in a vector space can easily be understood as points in a metric space. However, the interpretation of scalars is not as straightforward: what do the scalar weights in a convex combination mean in a metric space?

As seen in Section 3.2, a convex combination is an algebraic description of a convex hull. However, even if the notion of convex combination is not defined for metric spaces, convexity in metric spaces is still well defined through the notion of metric convex set that is a straightforward generalization of traditional convex set. Since convexity is well defined for metric spaces, we still have hope to generalize the scalar weights of a convex combination trying to make sense of them in terms of distance.

The weight of a point in a convex combination can be seen as a measure of relative linear "attraction" toward its corresponding point, versus attractions toward the other points of the combination. The closer a weight to 1, the stronger the attraction to the corresponding point. The point resulting from a convex combination can be seen as the equilibrium point of all the attraction forces. The distance between the equilibrium point and a point of the convex combination is, therefore, a decreasing function of the level of attraction (weight) of the point: the stronger the attraction, the smaller its distance to the equilibrium point. This observation can be used to reinterpret the weights of a convex combination in a metric space as follows: $y = w_1x_1 + w_2x_2 + w_3x_3$ with w_1, w_2, and w_3 greater than zero and $w_1 + w_2 + w_3 = 1$ is generalized to mean that y is a point such that $d(x_1, y) = f(w_1)$, $d(x_2, y) = f(w_2)$, and $d(x_3, y) = f(w_3)$, where f is a decreasing function.

This definition is formal and valid for all metric spaces, but it is nonconstructive. In contrast, a convex combination not only defines a convex hull, but also tells how to reach all its points. So how can we actually pick a point in the convex hull respecting the above distance requirements? Geometric crossover will help us with this, as we show in the next section.

To summarize, the requirements for a convex combination in a metric space are as follows.

(1) *Convex weights*: the weights respect the form of a convex combination: $w_1, w_2, w_3 > 0$ and $w_1 + w_2 + w_3 = 1$.
(2) *Convexity*: the convex combination operator combines x_1, x_2, and x_3 and returns a point in theirmetric convex hull (or simply triangle) under the metric of the space considered.
(3) *Coherence between weights and distances*: the distances to the equilibrium point are decreasing functions of their weights.
(4) *Symmetry*: the same value assigned to w_1, w_2, or w_3 has the same effect (e.g., in a equilateral triangle, if the coefficients have all the same value, the distances to the equilibrium point are the same).

3.4. Geometric PSO algorithm

The generic GPSO algorithm is illustrated in Algorithm 2. This differs from the standard PSO (Algorithm 1), in that:

(i) there is no velocity,
(ii) the equation of position update is the convex combination,
(iii) there is mutation, and
(iv) the parameters w_1, w_2, and w_3 are nonnegative and add up to one.

The specific PSOs for the Euclidean, Manhattan, and Hamming spaces use the randomized convex combination operators described in Section 4 and space-specific mutations. The randomization introduced by the randomized convex combination and by the mutation are of different types. The former allows us to pick points at random exclusively within the convex hull. The latter, as mentioned in Section 3.1, allows us to pick points outside the convex hull.

4. GEOMETRIC PSO FOR SPECIFIC SPACES

4.1. Euclidean space

The GPSO for the Euclidean space is *not* an extension of the traditional PSO. We include it to show how the general notions introduced in the previous section materialize in a familiar context. The convex combination operator for the Euclidean space is the traditional convex combination that produces points in the traditional convex hull.

In Section 3.3, we have mentioned how to interpret the weights in a convex combination in terms of distances. In the following, we show analytically how the weights of a convex combination affect the relative distances to the equilibrium point. In particular, we show that the relative distances are decreasing functions of the corresponding weights.

Theorem 4. *In a convex combination, the distances to the equilibrium point are decreasing functions of the corresponding weights.*

Proof. Let a, b, and c be three points in $\mathbb{R}^n$ and let $x = w_a a + w_b b + w_c c$ be a convex combination. Let us now decrease w_a to $w'_a = w_a - \Delta$ such that w'_a, w'_b, and w'_c still form a convex combination and that the relative proportions of w_b and w_c remain unchanged: $w'_b/w'_c = w_b/w_c$. This requires w'_b and w'_c to be $w'_b = w_b(1+\Delta/(w_b+w_c))$ and $w'_c = w_c(1+\Delta/(w_b+w_c))$. The equilibrium point for the new convex combination is

$$\begin{aligned} x' = {} & (w_a - \Delta)a + w_b(1 + \Delta/(w_b + w_c))b \\ & + w_c(1 + \Delta/(w_b + w_c))c. \end{aligned} \tag{4}$$

The distance between a and x is

$$|a - x| = \left| w_b(a - b) + w_c(a - c) \right|, \tag{5}$$

and the distance between a and the new equilibrium point is

$$\begin{aligned} \left| a - x' \right| &= \left| w_b(1 + \Delta/(w_b + w_c))(a - b) \right. \\ &\quad \left. + w_c(1 + \Delta/(w_b + w_c))(a - c) \right| \\ &= (1 + \Delta/(w_b + w_c))|a - x|. \end{aligned} \tag{6}$$

So when w_a decreases ($\Delta > 0$) and w_b and w_c maintain the same relative proportions, the distance between the point a and the equilibrium point x increases ($|a - x'| > |a - x|$). Hence, the distance between a and the equilibrium point is a decreasing function of w_a. For symmetry, this applies to the distances between b and c and the equilibrium point: they are decreasing functions of their corresponding weights w_b and w_c, respectively. □

The traditional convex combination in the Euclidean space respects the four requirements for a convex combination presented in Section 3.3.

4.2. Manhattan space

In the following, we first define a multiparental recombination for the Manhattan space and then prove that it respects the four requirements for being a convex combination presented in Section 3.3.

Definition 5 (box recombination family). Given two parents a and b in $\mathbb{R}^n$, a box recombination operator returns offspring o such that $o_i \in [\min(a_i, b_i), \max(a_i, b_i)]$ for $i = 1, \ldots, n$.

Theorem 5 (geometricity of box recombination). *Any box recombination is a geometric crossover under Manhattan distance.*

Proof. Theorem 5 is an immediate consequence of the product geometric crossover (Theorem 1). □

Definition 6 (three-parent box recombination family). Given three parents a, b, and c in $\mathbb{R}^n$, a box recombination operator returns offspring o such that $o_i \in [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$ for $i = 1, \ldots, n$.

Theorem 6 (geometricity of a three-parent box recombination). *Any three-parent box recombination is a geometric crossover under Manhattan distance.*

Proof. We prove it by showing that any multiparent box recombination $BX(a, b, c)$ can be decomposed as a sequence of two simple box recombinations. Since the simple box recombination is geometric (Theorem 5), this theorem is a simple corollary of the multiparental geometric decomposition theorem (Theorem 2).

We will show that $o' = BX(a, b)$ followed by $BX(o', c)$ can reach any offspring $o = BX(a, b, c)$. For each i, we have $o_i \in [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$. Notice that $[\min(a_i, b_i), \max(a_i, b_i)] \cup [\min(a_i, c_i), \max(a_i, c_i)] = [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$. We have two cases: (i) $o_i \in [\min(a_i, b_i), \max(a_i, b_i)]$ in which case o_i is reachable by the sequence $BX(a, b)_i \to o_i, BX(o, c)_i \to o_i$; (ii) $o_i \notin [\min(a_i, b_i), \max(a_i, b_i)]$, then it must be in $[\min(a_i, c_i), \max(a_i, c_i)]$ in which case o_i is reachable by the sequence $BX(a, b)_i \to a_i, BX(a, c)_i \to o_i$. □

Definition 7 (weighted multiparent box recombination). Given three parents a, b, and c in $\mathbb{R}^n$ and weights w_a, w_b, and w_c, a weighted box recombination operator returns offspring o such that $o_i = w_{a_i} a_i + w_{b_i} b_i + w_{c_i} c_i$ for $i = 1, \ldots, n$, where w_{a_i}, w_{b_i}, and w_{c_i} are a convex combination of randomly perturbed weights with expected values w_a, w_b, and w_c.

The difference between box recombination and linear recombination (Euclidean space) is that in the latter, the weights w_a, w_b, and w_c are randomly perturbed only once and the same weights are used for all the dimensions, whereas

the former one has a different randomly perturbed version of the weights for each dimension.

The weighted multiparent box recombination belongs to the family of multiparent box recombination because $o_i = w_{a_i}a_i + w_{b_i}b_i + w_{c_i}c_i \in [\min(a_i, b_i, c_i), \max(a_i, b_i, c_i)]$ for $i = 1, \ldots, n$, hence, it is geometric.

Theorem 7 (coherence between weights and distances). *In weighted multiparent box recombination, the distances of the parents to the expected offspring are decreasing functions of the corresponding weights.*

Proof. The proof of Theorem 7 is a simple variation of that of Theorem 4. □

In summary, in this section, we have introduced the weighted multiparent box recombination and shown that it is a convex combination operator satisfying the four requirements of a metric convex combination for the Manhattan space: convex weights (Definition 6), convexity (geometricity, Theorem 6), coherence (Theorem 7), and symmetry (self evident).

4.3. Hamming space

In this section, we first define a multiparental recombination for binary strings, that is, a straightforward generalization of mask-based crossover with two parents and then prove that it respects the four requirements for being a convex combination in the Hamming space presented in Section 3.3.

Definition 8 (three-parent mask-based crossover family). Given three parents a, b, and c in $\{0,1\}^n$, generate randomly a crossover mask of length n with symbols from the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Build the offspring o filling each position with the bit from the parent appearing in the crossover mask at the corresponding position.

The weights w_a, w_b, and w_c of the convex combination indicate, for each position in the crossover mask, the probability of having the symbols a, b, or c.

Theorem 8 (geometricity of three-parent mask-based crossover). *Any three-parent mask-based crossover is a geometric crossover under Hamming distance.*

Proof. We prove it by showing that any three-parent mask-based crossover can be decomposed as a sequence of two simple mask-based crossovers. Since the simple mask-based crossover is geometric, this theorem is a simple corollary of the multiparental geometric decomposition theorem (Theorem 2).

Let m_{abc} be the mask to recombine a, b, and c, producing the offspring o. Let m_{ab} be the mask obtained by substituting all occurrences of c in m_{abc} with b, and let m_{bc} be the mask obtained by substituting all occurrences of a in m_{abc} with b. First, recombine a and b using m_{ab} obtaining b'. Then, recombine b' and c using m_{bc} where the b's in the mask stand for alleles in b'. The offspring produced by the second crossover is o, so the sequence of the two simple crossovers is equivalent to the three-parent crossover. This is because the first crossover passes to the offspring all genes it needs to take from a according to m_{abc}, and the rest of the genes are all from b; the second crossover corrects those genes that should have been taken from parent c according to m_{abc}, but were taken from b instead. □

Theorem 9 (coherence between weights and distances). *In the weighted three-parent mask-based crossover, the distances of the parents to the expected offspring are decreasing functions of the corresponding weights.*

Proof. We want to know the expected distance from parent p_1, p_2, and p_3 to their expected offspring o as a function of the weights w_1, w_2, and w_3. To do so, we first determine, for each position in the offspring, the probability of it being the same as p_1. Then from that, we can easily compute the expected distance between p_1 and o. We have that

$$\begin{aligned}\mathrm{pr}\{o = p_1\} = \mathrm{pr}\{p_1 \longrightarrow o\} + \mathrm{pr}\{p_2 \longrightarrow o\}\cdot\mathrm{pr}\{p_1 \mid p_2\} \\ + \mathrm{pr}\{p_3 \longrightarrow o\}\cdot\mathrm{pr}\{p_1 \mid p_3\},\end{aligned} \tag{7}$$

where $\mathrm{pr}\{o = p_1\}$ is the probability of a bit of o at a certain position to be the same as the bit of p_1 at the same position; $\mathrm{pr}\{p_1 \to o\}$, $\mathrm{pr}\{p_2 \to o\}$, and $\mathrm{pr}\{p_3 \to o\}$ are the probabilities that a bit in o is taken from parents p_1, p_2, and p_3, respectively (these coincide with the weights of the convex combination w_1, w_2, and w_3); $\mathrm{pr}\{p_1 \mid p_2\}$ and $\mathrm{pr}\{p_1 \mid p_3\}$ are the probabilities that a bit taken from p_2 or p_3 coincides with the one in p_1 at the same location. These last two probabilities equal the number of common bits in p_1 and p_2 (and p_1 and p_3) over the length of the strings n. So $\mathrm{pr}\{p_1 \mid p_2\} = 1 - H(p_1, p_2)/n$ and $\mathrm{pr}\{p_1 \mid p_3\} = 1 - H(p_1, p_3)/n$, where $H(\cdot, \cdot)$ is the Hamming distance. So (7) becomes

$$\begin{aligned}\mathrm{pr}\{o = p_1\} = w_1 + w_2(1 - H(p_1, p_2)/n) \\ + w_3(1 - H(p_1, p_3)/n).\end{aligned} \tag{8}$$

Hence, the expected distance between the parent p_1 and the offspring o is $E(H(p_1, o)) = n\cdot(1 - \mathrm{pr}\{o = p_1\}) = w_2H(p_1, p_2) + w_3H(p_1, p_3)$.

Notice that this is a decreasing function of w_1 because increasing w_1 forces w_2 or w_3 to decrease since the sum of the weights is constant, hence, $E(H(p_1, o))$ decreases. Analogously, $E(H(p_2, o))$ and $E(H(p_3, o))$ are decreasing functions of their weights w_2 and w_3, respectively. □

In summary, in this section, we have introduced the weighted multiparent mask-based crossover and shown that it is a convex combination operator satisfying the four requirements of a metric convex combination for the Hamming space: convex weights (Definition 8), convexity (geometricity, Theorem 8), coherence (Theorem 9), and symmetry (self evident).

5. GEOMETRIC PSO FOR OTHER REPRESENTATIONS

Before looking into how we can extend GPSO to other solution representations, we will discuss the relation between

the three-parental geometric crossover and the symmetry requirement for a convex combination.

For each of the spaces considered in Section 4, we have first considered (or defined) a three-parental recombination and then proven that it is a three-parental geometric crossover by showing that it can actually be decomposed into two sequential applications of a geometric crossover for the specific space.

However, we could have skipped the *explicit definition* of a three-parental recombination altogether. In fact, to obtain the three-parental recombination, we could have used two sequential applications of a known two-parental geometric crossover for the specific space. This composition is indeed a three-parental recombination (it combines three parents) and it is decomposable by construction. Hence, it is a three-parental geometric crossover. This, indeed, would have been simpler than the route we took.

The reason we preferred to define explicitly a three-parental recombination is that the requirement of symmetry of the convex combination is true by construction: if the roles of any two parents are swapped by exchanging in the three-parental recombination both positions and the respective recombination weights, the resulting recombination operator is equivalent to the original operator.

The symmetry requirement becomes harder to enforce and prove for a three-parental geometric crossover obtained by two sequential applications of a two-parental geometric crossover. We illustrate this in the following.

Let us consider three parents a, b, and c with positive weights w_a, w_b, and w_c which add up to one. If we have a symmetric three-parental weighted geometric crossover ΔGX, the symmetry of the recombination is guaranteed by the symmetry of the operator. So $\Delta GX((a,w_a),(b,w_b),(c,w_c))$ is equivalent to $\Delta GX((b,w_b),(a,w_a),(c,w_c))$. Hence, the requirement of symmetry on the weights of the convex combination holds. If we consider a three-parental recombination defined by using twice a two-parental genetic crossover GX, we have

$$\begin{aligned}&\Delta GX((a,w_a),(b,w_b),(c,w_c))\\&\quad = GX((GX((a,w'_a),(b,w'_b)),w_{ab}),(c,w'_c))\end{aligned} \tag{9}$$

with the constraint that w'_a and w'_b are positive and add up to one, and w_{ab} and w'_c are positive and add up to one. Notice the inherent asymmetry in this expression: the weights w'_a and w'_b are not directly comparable with w'_c because they are relative weights between a and b. Moreover, there is the extra weight w_{ab}. This asymmetry makes the requirement of symmetry problematic to meet: given the desired w_a, w_b, and w_c, what values of w'_a, w'_b, w_{ab}, and w'_c should we choose to obtain an equivalent symmetric three-parental weighted recombination expressed as a sequence of two two-parental geometric crossovers?

For the Euclidean space, it is easy to answer this question using simple algebra as follows:

$$\begin{aligned}\Delta GX &= w_a \cdot a + w_b \cdot b + w_c \cdot c\\&= (w_a + w_b)\left(\frac{w_a}{w_a + w_b}\cdot a + \frac{w_b}{w_a + w_b}\cdot b\right) + w_c \cdot c.\end{aligned} \tag{10}$$

Since the convex combination of two points in the Euclidean space is $GX((x,w_x),(y,w_y)) = w_x \cdot x + w_y \cdot y$, and $w_x, w_y > 0$ and $w_x + w_y = 1$, then

$$\begin{aligned}&\Delta GX((a,w_a),(b,w_b),(c,w_c))\\&= GX\left[\left(GX\left(\left(a,\frac{w_a}{w_a+w_b}\right),\left(b,\frac{w_b}{w_a+w_b}\right)\right),w_a+w_b\right),(c,w_c)\right].\end{aligned} \tag{11}$$

However, the question may be less straightforward to answer for other spaces, although, we could use the equation above as a rule-of-thumb to map the weights of ΔGX and the weights in the sequential GX decomposition to obtain a nearly symmetric convex combination.

Where does this discussion leave us in relation to the extension of GPSO to other representations? We have seen that there are two alternative ways to produce a convex combination for a new representation: (i) explicitly define a symmetric three-parental recombination for the new representation and then prove its geometricity by showing that it is decomposable into a sequence of two two-parental geometric crossovers (*explicit definition*), or (ii) use twice the simple geometric crossover to produce a symmetric or nearly symmetric three-parental recombination (*implicit definition*). The second option is also very interesting because *it allows us to extended automatically GPSO to all representations we have geometric crossovers for* (such as permutations, GP trees, and variable-length sequences, to mention but a few), *and virtually to any other complex solution representation.*

6. EXPERIMENTAL RESULTS FOR EUCLIDEAN, MANHATTAN, AND HAMMING SPACES

We have run two groups of experiments: one for the continuous version of the GPSO (`EuclideanPSO`, or EPSO for short, and `ManhattanPSO`, or MPSO), and one for the binary version (`HammingPSO`, or HPSO).

For the Euclidean and Manhattan versions, we have compared the performances with those of a continuous PSO (`BasePSO`, or BPSO) with constriction and inertia, whose parameters are as in Table 1. We have run the experiments for dimensions 2, 10, and 30 on the following five-benchmark functions: *F1C Sphere* [19], *F2C Rosenbrock* [19], *F3C Ackley* [20], *F4C Griewangk* [21], and *F5C Rastrigin* [22]. The Hamming version has been tested on the De Jong's test suite [19]: *F1 Sphere (30)*, *F2 Rosenbrock (24)*, *F3 Step (50)*, *F4 Quartic (240)*, and *F5 Shekel (34)*, where the numbers in brackets are the dimensions of the problems in bits. All functions in the test bed have global optimum 0 and are to be maximized.

Since there is no equivalent GPSO with $\phi_1 = \phi_2 = 2.05$ ($\phi_1 + \phi_2 > 1$, which does not respect the conditions in Theorem 3), we have decided to set w_1, w_2, and w_3 proportional to ω, ϕ_1, and ϕ_2, respectively, and summing up to one (see Table 2).

For the binary version, the parameters of population size, number of iterations, and w_1, w_2, and w_3 have been tuned on the sphere function and are as in Table 3. From the parameters tuning, it appears that there is a preference for values of w_1 close to zero. This means that there is a bias towards

Table 1: Parameters for BPSO.

Population size	20, 50 particles
Stop condition	200 iterations
$V_{\max} = X_{\max}$	Max_value-min_value
κ	0.729
ω	1.0
$\phi_1 = \phi_2$	2.05

Table 2: Parameters for EPSO and MPSO.

Population size	20, 50 particles
Stop condition	200 iterations
$V_{\max} = X_{\max}$	Max_value-min_value
Mutation	uniform in $[-0.5,0.5]$
w_1	$\omega/(\omega + \phi_1 + \phi_2) = 1.0/5.10$
w_2	$\phi_1/(\omega + \phi_1 + \phi_2) = 2.05/5.10$
w_3	$\phi_2/(\omega + \phi_1 + \phi_2) = 2.05/5.10$

Table 3: Selected parameters for HPSO.

Population size	100 particles
Iterations	400
Bitwise mutation rate	$1/N$
$w_1 = 0$	$w_2 = w_3 = 1/2$
$w_1 = 1/6$	$w_2 = w_3 = 5/12$

the swarm and particle bests, and less attraction towards the current particle position.

For each set up, we performed 20 independent runs. Table 4 shows the best and the mean fitness value (i.e., the fitness value at the position where the population converges) found by the swarm when exploring continuous spaces. This table summarizes the results for the three algorithms presented, over the five test functions, for the two choices of population size, giving an immediate comparison of the performances. Overall the GPSOs (EPSO and MPSO), compare very favourably with BPSO, outperforming it in many cases. This is particularly interesting since it suggests that the inertia term (not present in GPSO) is not necessary for good performance.

In two dimensions, the results for all the functions (for PSOs both with 20 and 50 particles) are nearly identical, with BPSO and the two GPSOs both performing equally well. In higher dimensions, it is interesting to see how dimensionality does not seem to affect the quality of the results of GPSOs (while there is a fairly obvious decline in the performance of BPSO when dimension increases).

Also, EPSO's and MPSO's results are virtually identical. Let us recall from Section 4.2 the difference between Euclidean and Manhattan spaces, that is, "the difference between box recombination and linear recombination (Euclidean space) is that in the latter, the weights are randomly perturbed only once and the same weights are used for all the dimensions, whereas the former one has a different randomly perturbed version of the weights for each dimension." The results obtained show, therefore, that at least in this context, randomly perturbing the weights once for all dimensions, or perturbing them for each dimension, does not seem to affect the end result.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 2: Example of Sudoku puzzle.

Table 5 shows the mean of the best fitness value and the best fitness value over the whole population for the binary version of the algorithm. The algorithm compares well with results reported in the literature, with HPSO obtaining near optimal results on all functions. Interestingly, the algorithm works at its best when w_1, the weight for x_i (the particle position), is zero. This corresponds to a degenerated PSO that makes decisions without considering the current position of the particle.

7. GEOMETRIC PSO FOR SUDOKU

In this section, we will put into practice the ideas discussed in Section 5 and propose a GPSO to solve the Sudoku puzzle. In Section 7.1, we introduce the Sudoku puzzle. In Section 7.2, we present a geometric crossover for Sudoku. In Section 7.3, we present a three-parental crossover for Sudoku and show that it is a convex combination.

7.1. Description of Sudoku

Sudoku is a logic-based placement puzzle. The aim of the puzzle is to enter a digit from 1 through 9 in each cell of a 9×9 grid made up of 3×3 subgrids (called "regions"), starting with various digits given in some cells (the "givens"). Each row, column, and region must contain only one instance of each digit. In Figure 2, we show an example of Sudoku puzzle. Sudoku puzzles with a unique solution are called proper Sudoku, and the majority of published grids are of this type.

Published puzzles often are ranked in terms of difficulty. Perhaps surprisingly, the number of "givens" has little or no bearing on a puzzle's difficulty. This is based on the relevance and the positioning of the numbers rather than the quantity of the numbers.

The 9×9 Sudoku puzzle of any difficulty can be solved very quickly by a computer. The simplest way is to use some brute force trial-and-error search employing back tracking.

Table 4: Test results for continuous versions: best and mean fitness values found by the swarm over 20 runs at last iteration (iteration 200).

		BPSO			EPSO			MPSO		
	Dim.	2	10	30	2	10	30	2	10	30
Popsize = 20	F1C best	−5.35e-14	−1.04	−59.45	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−6.54e-09	−20.75	−168.19	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	F2C best	0.00	−36.18	−1912.05	−0.71	−8.98	−28.97	−0.66	−8.96	−28.97
	mean	−97.91	−979.56	−8847.44	−1.0	−9.0	−29.0	−1.0	−9.0	−29.0
	F3C best	−3.06e-05	−8.05	−18.09	0.0	0.0	0.0	0.0	0.0	0.0
	mean	−0.00	−14.86	−20.49	0.0	0.0	0.0	0.0	0.0	0.0
	F4C best	−0.31	−1.10	−6.67	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	mean	−1.52	−2.98	−17.04	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	F5C best	−0.33	−58.78	−305.11	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−10.41	−160.98	−504.62	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
Popsize = 50	F1C best	−3.67e-13	−0.60	−53.93	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−1.11e-08	−19.09	−176.07	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	F2C best	0.00	−19.46	−1639.46	−0.57	−8.96	−28.96	−0.53	−8.95	−29.0
	mean	−56.04	−791.88	−9425.92	−1.0	−9.0	−29.0	−1.0	−9.0	−29.0
	F3C best	−1.81e-06	−6.78	−17.62	0.0	0.0	0.0	0.0	0.0	0.0
	mean	−0.00	−15.55	−20.43	0.0	0.0	0.0	0.0	0.0	0.0
	F4C best	−0.30	−1.05	−6.14	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	mean	−1.63	−2.79	−17.79	−0.29	−1.0	−1.0	−0.29	−1.0	−1.0
	F5C best	−0.10	−53.67	−302.29	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0
	mean	−3.56	−159.76	−503.48	−0.0	−0.0	−0.0	−0.0	−0.0	−0.0

Table 5: Test results for HPSO with selected parameters for De Jong's test suite.

	F1	F2	F3	F4	F5
$\overline{\omega} = 0.0$ best	−0.000 15	−0.000 34	−0.0	3.451 70	−1.131 83
mean	−5.515 40	−54.144 53	−2.594	−5.382 33	−142.678 53
$\overline{\omega} = \frac{1}{6}$ best	−0.000 125	−0.000 297	−0.0	3.273 980	−1.111 220
mean	−5.375 902	−85.170 099	−2.949	−6.919 343	−167.283 327

Constraint programming is a more efficient method that propagates the constraints successively to narrow down the solution space until a solution is found or until alternate values cannot otherwise be excluded, in which case, backtracking is applied. A highly efficient way of solving such constraint problems is the Dancing Links Algorithm by Donald Knuth [23].

The general problem of solving Sudoku puzzles on $n^2 \times n^2$ boards of $n \times n$ blocks is known to be NP complete [24]. This means that, unless P = NP, the exact solution methods that solve very quickly the 9×9 boards take exponential time in the board size in the worst case. However, it is unknown whether the general Sudoku problem restricted to puzzles with unique solutions remains NP complete or becomes polynomial.

Solving Sudoku puzzles can be expressed as a graph coloring problem. The aim of the puzzle in its standard form is to construct a proper 9 coloring of a particular graph, given a partial 9 coloring.

A valid Sudoku solution grid is also a Latin square. Sudoku imposes the additional regional constraint. Latin square completion is known to be NP complete. A further relaxation of the problem allowing repetitions on columns (or rows) makes it polynomially solvable.

Admittedly, evolutionary algorithms and meta-heuristics in general are not the best techniques to solve Sudoku because they do not exploit systematically the problem constraints to narrow down the search. However, Sudoku is an interesting study case because it is a relatively simple problem but not trivial since is NP complete, and the different types of constraints make Sudoku an interesting playground for search operator design.

7.2. Geometric crossover for Sudoku

Sudoku is a constraint satisfaction problem with four types of constraints:

(1) fixed elements,
(2) rows are permutations,
(3) columns are permutations,
(4) and boxes are permutations.

It can be cast as an optimization problem by choosing some of the constraints as hard constraints that all solutions have to respect, and the remaining constraints as soft constraints that can be only partially fulfilled, and the level of fulfilment is the fitness of the solution. We consider a space with the following characteristics:

(i) *hard constraints*: fixed positions and permutations on rows,
(ii) *soft constraints*: permutations on columns and boxes,
(iii) *distance*: sum of swap distances between paired rows (row-swap distance),
(iv) *feasible geometric mutation*: swap two nonfixed elements in a row, and
(v) *feasible geometric crossover*: row-wise PMX and row-wise cycle crossover.

The chosen mutation preserves both fixed positions and permutations on rows (hard constraints) because swapping elements within a row, which is a permutation, returns a permutation. The mutation is 1-geometric under row-swap distance. Row-wise PMX and row-wise cycle crossover recombine parent grids applying, respectively, PMX and cycle crossover to each pair of corresponding rows. In case of PMX, the crossover points can be selected to be the same for all rows, or be random for each row. In terms of offspring that can be generated, the second version of row-wise PMX includes all the offspring of the first version.

Simple PMX and simple cycle crossover applied to parent permutations always return permutations. They also preserve fixed positions. This is because both are geometric under swap distance and, in order to generate offspring on a minimal sorting path between parents using swaps (sorting one parent into the order of the other parent), they have to avoid swaps that change common elements in both parents (elements that are already sorted). Therefore, also row-wise PMX and row-wise cycle crossover preserve both hard constraints.

Using the product geometric crossover Theorem 1, it is immediate to see that both row-wise PMX and row-wise cycle crossover are geometric under row-swap distance since simple PMX and simple cycle crossover are geometric under swap distance. Since simple cycle crossover is also geometric under Hamming distance (restricted to permutations), row-wise cycle crossover is also geometric under Hamming distance.

To restrict the search to the space of grids with fixed positions and permutations on rows, the initial population must be seeded with feasible random solutions taken from this space. Generating such solutions can be done still very efficiently.

The fitness function (to maximize) is the sum of the number of unique elements in each row plus the sum of the number of unique elements in each column plus the sum of the number of unique elements in each box. So for a 9×9 grid, we have a maximum fitness of $9 \cdot 9 + 9 \cdot 9 + 9 \cdot 9 = 243$ for a completely correct Sudoku grid, and a minimum fitness little more than $9 \cdot 1 + 9 \cdot 1 + 9 \cdot 1 = 27$ because for each row, column, and square, there is at least one unique element type.

```
Mask: 1 2 2 3 1 3 2
  p1: 1 2 3 4 5 6 7
  p2: 3 5 1 4 2 7 6
  p3: 3 2 1 4 5 7 6

   o: 1 5 3 4 2 7 6
```

FIGURE 3: Example of multiparental sorting crossover.

It is possible to show that the fitness landscapes associated with this space is smooth, making the search operators proposed a good choice for Sudoku.

7.3. Convex combination for Sudoku

In this section, we first define a multiparental recombination for permutations and then prove that it respects the four requirements for being a convex combination presented in Section 3.3.

Let us consider the example in Figure 3 to illustrate how the *multiparental sorting crossover* works.

The mask is generated at random and is a vector of the same length of the parents. The number of 1's, 2's, and 3's in the mask is proportional to the recombination weights w_1, w_2, and w_3 of the parents. Every entry of the mask indicates to which parent the other two parents need to be equal to for that specific position. In a parent, the content of a position is changed by swapping it with the content of another position in the parent. The recombination proceeds as follows. The mask is scanned from the left to the right. In position 1, the mask has a 1. This means that at position 1, parent p_2 and parent p_3 have to become equal to parent p_1. This is done by swapping the elements 1 and 3 in parent p_2 and the elements 1 and 3 in parent p_3. The recombination now continues on the updated parents: parent p_1 is left unchanged and the current parent p_2 and parent p_3 are the original parents p_2 and p_3 after the swap. At position 2, the mask has 2. This means that at position 2, the current parent p_1 and current parent p_3 have to become equal to the current parent p_2. So at position 2, parent p_1 and parent p_3 have to get 5. To achieve this, in parent p_1, we need to swap elements 2 and 5, and in parent p_3, we need to swap elements 2 and 5. The recombination continues on the updated parents for position 3 and so on, up to the last position in the mask. At this point, the three parents are now equal because at each position, one element of the permutation has been fixed in that position and it is automatically not involved in any further swap. Therefore, after all positions have been considered, all elements are fixed. The permutation to which the three parents converged is the offspring permutation. This recombination sorts by swaps the three parents towards each other according to the contents of the crossover mask, and the offspring is the result of this multiple sorting. This recombination can be easily generalized to any number of parents.

Theorem 10 (geometricity of three-parental sorting crossover). *Three-parental sorting crossover is a geometric crossover under swap distance.*

Proof. A three-parental sorting crossover with recombination mask m_{123} is equivalent to a sequence of two two-parental sorting crossovers: the first is between parents p_1 and p_2 with recombination mask m_{12} obtained by substituting all 3's with 2's in m_{123}. The offspring p_{12} so obtained is recombined with p_3 with recombination mask m_{23} obtained by substituting all 1's with 2's in m_{123}. So for Theorem 2, the three-parental sorting crossover is geometric. □

Theorem 11 (coherence between weights and distances). *In a weighted multiparent sorting crossover, the swap distances of the parents to the expected offspring are decreasing functions of the corresponding weights.*

Proof. The weights associated to the parents are proportional to their frequencies in the recombination mask. The more occurrences of a parent in the recombination mask, the smaller the swap distance between this parent and the offspring. This is because the mask tells the parent to copy at each position. So the higher the weight of a parent, the smaller its distance to the offspring. □

The weighted multiparental sorting crossover is a convex combination operator satisfying the four requirements of a metric convex combination for the swap space: convex weights sum to 1 by definition, convexity (geometricity, Theorem 10), coherence (Theorem 11), and symmetry (self evident).

The solution representation for Sudoku is a vector of permutations. For the product geometric crossover theorem, the compound crossover over the vector of permutations that applies a geometric crossover to each permutation in the vector is a geometric crossover. Theorem 12 extends to the case of a multiparent geometric crossover.

Theorem 12 (product geometric crossover for convex combinations). *A convex combination operator, applied to each entry of a vector, results in a convex combination operator for the entire vector.*

Proof. The product geometric crossover theorem (Theorem 1) is true because the segment of a product space is the Cartesian product of the segments of its projections. A segment is the convex hull of two points (parents). More generally, it holds that the convex hull (of any number of points) of a product space is the Cartesian product of the convex hulls of its projections [17]. The product geometric crossover then naturally generalizes to the multiparent case. □

8. EXPERIMENTAL RESULTS FOR SUDOKU

In order to test the efficacy of the GPSO algorithm on the Sudoku problem, we ran several experiments in order to thoroughly explore the parameter space and variations of the algorithm. The algorithm in itself is a straightforward implementation of the GPSO algorithm given in Section 3.4 with the search operators for Sudoku presented in Section 7.3.

The parameters we varied were swarm sociality (w_2) and memory (w_3), each of which were in turn set to 0, 0.2, 0.4, 0.6, 0.8, and 1.0. Since the attraction to each particle's position is defined as $w_1 = 1 - w_2 - w_3$, the space of this parameter was implicitly explored. Likewise, mutation probability was set to either 0, 0.3, 0.7, or 1.0. The swarm size was set to be either 20, 100, or 500 particles, but the number of updates was set so that each run of the algorithm resulted in exactly 100 000 fitness evaluations (thus performing 5000, 1000, or 200 updates). Further, each combination was tried with ring topology, von Neumann topology (or lattice topology), and global topology, which are the most common topologies.

As explained in Section 5, there are two alternative ways of producing a convex combination: either using a convex combination operator or simply applying twice a two-parental weighted recombination with appropriate weights to obtain the convex combination. Both ways to produce convex combination operators, explicit and implicit, were tried on preliminary runs and turned out to produce indistinguishable results. In the end, we used the convex combination operator.

Table 6: Average of bests of 50 runs with population size 100, lattice topology, and mutation 0.0, varying sociality (vertical) and memory (horizontal).

	Memory					
Sociality	0.0	0.2	0.4	0.6	0.8	1.0
1.0	208	—	—	—	—	—
0.8	227	229	—	—	—	—
0.6	230	233	235	—	—	—
0.4	231	236	237	240	—	—
0.2	232	239	241	**242**	**242**	—
0.0	207	207	207	207	207	207

Table 7: Average of bests of 50 runs with population size 100, lattice topology, and mutation 0.3, varying sociality (vertical) and memory (horizontal).

	Memory					
Sociality	0.0	0.2	0.4	0.6	0.8	1.0
1.0	238	—	—	—	—	—
0.8	238	237	—	—	—	—
0.6	239	239	240	—	—	—
0.4	240	240	241	241	—	—
0.2	240	241	**242**	**242**	**242**	—
0.0	213	231	232	233	233	233

8.1. Effects of varying coefficients

The best population size is 100. The other two sizes we studied (20 and 500) were considerably worse. The best topology is the lattice (von Neumann) topology. The other two topologies we studied were worse (see Table 9).

From Tables 6–8, we can see that mutation rates of 0.3 and 0.7 perform better than no mutation at all. We can also see that parameter settings with w_1 (i.e., the attraction of

Table 8: Average of bests of 50 runs with population size 100, lattice topology and mutation 0.7, varying sociality (vertical) and memory (horizontal).

	Memory					
Sociality	0.0	0.2	0.4	0.6	0.8	1.0
1.0	232	—	—	—	—	—
0.8	232	240	—	—	—	—
0.6	228	241	241	—	—	—
0.4	224	**242**	**242**	**242**	—	—
0.2	219	234	**242**	**242**	**242**	—
0.0	215	226	233	233	236	236

Table 9: Success rate of various methods.

Method	Success
GA	50/50
Hillclimber	35/50
GPSO-global	7/50
GPSO-ring	20/50
GPSO-von Neumann	36/50

the particle's previous position) set to more than 0.4 generally perform badly. The best configurations generally have w_2 (i.e., sociality) set to 0.2 or 0.4, w_3 (i.e., memory) set to 0.4 or 0.6, and w_1 set to 0 or 0.2. This gives us some indication of the importance of the various types of recombinations in GPSO as applied at least to this particular problem. Surprisingly, the algorithm works at its best when the weight of the particle position (w_1) is zero or nearly zero. In the case of w_1 set to 0, GPSO, in fact, degenerates to a type of evolutionary algorithm with deterministic uniform selection, mating with the population best with local replacement between parents and offspring.

8.2. PSO versus EA

Table 9 compares the success rate of the best configurations of various methods we have tried. Success is here defined as the number of runs (out of 50) where the global optimum (243) is reached. All the methods were allotted the same number of function evaluations per run.

From the table, we can see that the von Neumann topology clearly outperforms the other topologies we tested, and that a GPSO with this topology can achieve a respectable success rate on this tricky noncontinuous problem. However, the best genetic algorithm still significantly outperforms the best GPSO we have found so far. (Notice that this is true only when considering GPSO as an optimizer. The approximation behavior of the GPSO is very good: with the right parameter setting, the GPSO reaches on average a fitness of 242 out of 243 (see Tables 6–8).) We believe this to be at least partly the effect of the even more extensive tuning of parameters and operators undertaken in our GA experiments.

9. CONCLUSIONS AND FUTURE WORK

We have extended the geometric framework with the notion of multiparent geometric crossover, that is, a natural generalization of two-parental geometric crossover: offspring are in the convex hull of the parents. Then, using the geometric framework, we have shown an intimate relation between a simplified form of PSO (without the inertia term) and evolutionary algorithms. This has enabled us to generalize, in a natural, rigorous, and automatic way, PSO for any type of search space for which a geometric crossover is known.

We specialized the general PSO to Euclidean, Manhattan, and Hamming spaces, obtaining three instances of the general PSO for these spaces, EPSO, MPSO, and HPSO, respectively. We have performed extensive experiments with these new GPSOs. In particular, we applied EPSO, MPSO, and HPSO to standard sets of benchmark functions and obtained a few surprising results. Firstly, the GPSOs have performed really well, beating the canonical PSO with standard parameters most of the time. Secondly, they have done so right out of the box. That is, unlike the early versions of PSO which required considerable effort before a good general set of parameters could be found, with GPSO, we have done very limited preliminary testing and parameter tuning, and yet the new PSOs have worked well. This suggests that they may be quite robust optimisers. This will need to be verified in future research. Thirdly, HPSO works at its best with only weak attraction toward the current position of the particle. With this configuration, GPSO almost degenerates to a type of genetic algorithm.

An important feature of the GPSO algorithm is that it allows one to automatically define PSOs for all spaces for which a geometric crossover is known. Since geometric crossovers are defined for all of the most frequently used representations and many variations and combinations of those, our geometric framework makes it possible to derive PSOs for all such representations. GPSO is rigorous generalization of the classical PSO to general metric spaces. In particular, it applies to combinatorial spaces.

We have demonstrated how simple it is to specify the general GPSO algorithm to the space of Sudoku grids (vectors of permutations), using both an explicit and an implicit definitions of convex combination. We have tested the new GPSO on Sudoku and have found that (i) the communication topology makes a huge difference and that the lattice topology is by far the best; (ii) as for HPSO, the GPSO on Sudoku works better with weak attraction toward the current position of the particle; (iii) the GSPO on Sudoku finds easily near-optimal solutions but it does not always find the optimum. Admittedly, GPSO is not the best algorithm for the Sudoku puzzle where the aim is to obtain the correct solution all the times, not a nearly correct one. This suggests that GPSO would be much more profitably applied to combinatorial problems for which one would be happy to find near-optimal solutions quickly.

In summary, we presented a PSO algorithm that has been quite successfully applied to a nontrivial combinatorial space. This shows that GPSO is indeed a natural and promising generalization of classical PSO. In future work, we

will consider GPSO for even more challenging combinatorial spaces such as the space of genetic programs. Also, since the inertia term is a very important feature of classical PSO, we want to generalize it and test the GPSO with the inertia term on combinatorial spaces.

ACKNOWLEDGMENT

The second and fourth authors would like to thank EPSRC, Grant no. GR/T11234/01 "Extended Particle Swarms," for the financial support.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[2] M. Clerc, "Discrete particle swarm optimization, Illustrated by the traveling salesman problem," in *New Optimization Techniques in Engineering*, Springer, New York, NY, USA, 2004.

[3] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (ICSMC '97)*, vol. 5, pp. 4104–4108, Orlando, Fla, USA, October 1997.

[4] A. Moraglio and R. Poli, "Topological interpretation of crossover," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '04)*, vol. 3102, pp. 1377–1388, Seattle, Wash, USA, June 2004.

[5] A. Moraglio and R. Poli, "Topological crossover for the permutation representation," in *Proceedings of the Workshops on Genetic and Evolutionary Computation (GECCO '05)*, pp. 332–338, Washington, DC, USA, June 2005.

[6] A. Moraglio and R. Poli, "Geometric landscape of homologous crossover for syntactic trees," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 427–434, Edinburgh, UK, September 2005.

[7] A. Moraglio, R. Poli, and R. Seehuus, "Geometric crossover for biological sequences," in *Proceedings of the 9th European Conference on Genetic Programming*, pp. 121–132, Budapest, Hungary, April 2006.

[8] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[9] A. Moraglio, J. Togelius, and S. Lucas, "Product geometric crossover for the Sudoku puzzle," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 470–476, Vancouver, BC, Canada, July 2006.

[10] T. Jones, *Evolutionary algorithms, fitness landscapes and search*, Ph.D. thesis, University of New Mexico, Albuquerque, NM, USA, 1995.

[11] M. Deza and M. Laurent, *Geometry of Cuts and Metrics*, Springer, New York, NY, USA, 1991.

[12] A. Moraglio and R. Poli, "Product geometric crossover," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06)*, pp. 1018–1027, Reykjavik, Iceland, September 2006.

[13] A. Moraglio and R. Poli, "Geometric crossover for sets, multisets and partitions," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06)*, pp. 1038–1047, Reykjavik, Iceland, September 2006.

[14] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[15] M. Clerc, "When nearer is better," Hal Open Archive, 2007.

[16] P. M. Pardalos and M. G. C. Resende, *Handbook of Applied Optimization*, Oxford University Press, Oxford, UK, 2002.

[17] M. L. J. van de Vel, *Theory of Convex Structures*, North-Holland, Amsterdam, The Netherlands, 1993.

[18] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[19] K. De Jong, *An analysis of the behaviour of a class of genetic adaptive systems*, Ph.D. thesis, University of Michigan, Ann Arbor, Mich, USA, 1975.

[20] T. Bäck, *Evolutinary Algorithms in Theory and in Practice*, Oxford University Press, Oxford, UK, 1996.

[21] A. Törn and A. Zilinskas, *Global Optimization*, vol. 350 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 1989.

[22] H. Müehlenbein, M. Schomisch, and J. Born, "The Parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, pp. 619–632, 1991.

[23] D. E. Knuth, "Dancing links," Preprint P159, Stanford University, 2000.

[24] T. Yato and T. Seta, "Complexity and completeness of finding another solution and its application to puzzles," Preprint, University of Tokyo, 2005.