Geometric Particle Swarm Optimization for the Sudoku Puzzle

Alberto Moraglio
Department of Computer Science
University of Essex, UK
amoragn@essex.ac.uk

Julian Togelius
Department of Computer Science
University of Essex, UK
jtogel@essex.ac.uk

ABSTRACT

Geometric particle swarm optimization (GPSO) is a recently introduced generalization of traditional particle swarm optimization (PSO) that applies to all combinatorial spaces. The aim of this paper is to demonstrate the applicability of GPSO to non-trivial combinatorial spaces. The Sudoku puzzle is a perfect candidate to test new algorithmic ideas because it is entertaining and instructive as well as a non-trivial constrained combinatorial problem. We apply GPSO to solve the sudoku puzzle.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory

Keywords

Particle Swarm Optimisation, Metric Space, Geometric Crossover, Sudoku

1. INTRODUCTION

Particle swarm optimisation [8] has traditionally been applied to continuous search spaces. Although a version of PSO for binary search spaces has been defined [7], all attempts to extend PSO to richer spaces, such as, for example, combinatorial spaces, have had no real success [3].

There are two ways of extending PSO to richer spaces. The first one is rethinking and adapting the PSO for each new solution representation. The second is making use of a rigorous mathematical generalisation to a general class of spaces of the notion (and motion) of particles. This second approach has the advantage that a PSO can be derived in a principled way for any search space belonging to the given class. In recent work [16] we have pursued this approach. We have shown *formally* how a general form of PSO (without the momentum term) can be obtained by using theoretical tools developed for a different form of search algorithms, namely evolutionary algorithms using geometric crossover and geometric mutation. These are representation-independent operators that generalise many pre-existing search operators for the major representations, such as binary strings [12], real vectors [12], permutations [14], syntactic trees [13] and sequences [15]. We have demonstrated how to derive the specific PSO for the cases of Euclidean, Manhattan and Hamming spaces and reported good experimental results.

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 1: Example of Sudoku puzzle.

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 9x9 grid made up of 3x3 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 fig 1 we show an example 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. It is based on the relevance and the positioning of the numbers rather than the quantity of the numbers.

The 9x9 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. 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 val-

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

ues 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 [9].

The general problem of solving Sudoku puzzles on $n^2 \times n^2$ boards of n x n blocks is known to be NP-complete [19]. This means that, unless P=NP, the exact solution methods that solve very quickly the 9x9 boards take exponential time in the board size in the worst case. However, it is unknown weather 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 technique 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.

In previous work [10] we have used the geometric framework to design an evolutionary algorithm to solve the Sudoku puzzle and obtained very good experimental results.

The aim of this paper is to demonstrate that GPSO can be specified easily to non-trivial combinatorial spaces. We demonstrate this by applying GPSO to solve the Sudoku puzzle. We report extensive experimental results.

In section 2, we introduce the geometric framework. In section 3, we introduce the general geometric particle swarm optimization algorithm. In section 4, we apply GPSO to Sudoku. In section 5, we report extensive experimental results. In section 6, 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 [6] 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 [12]. For more details on these definitions see [4].

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 $B(x; r) = \{y \in S | d(x, y) \leq r\}$ 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 $[x; y] = \{z \in S | d(x, z) + d(z, y) = d(x, y)\}$ where $x, y \in S$ are called extremes of the segment. Metric ball and metric segment generalise 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 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 $L = (M, g)$ is the corresponding *fitness landscape*.

2.2 Geometric crossover

Definition: A binary operator is a geometric crossover under the metric d if all offspring 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 only function of the metric d* associated with the search space.

This class of operators is really broad. For vectors of reals, various types of blend or line crossovers, box recombinations, and discrete recombinations are geometric crossovers [12]. For binary and multary strings, all homologous crossovers are geometric [12, 11]. For permutations, PMX, Cycle crossover, merge crossover and others are geometric crossovers [14]. For syntactic trees, the family of homologous crossovers are geometric [13]. Recombinations for several more complex representations are also geometric [15, 12, 14, 17].

2.3 Geometric crossover for permutations

In previous work we have studied various crossovers for permutations, revealing that PMX [5], a well-known crossover for permutations, is geometric under swap distance. Also, we found that Cycle crossover [5], 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. 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 prin-

ciple, 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 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, 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 meta-heuristics [18].

Rule-of-thumb 1: if we have a good distance for the problem at hand than we have good geometric mutation and good geometric crossover

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 [11] 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 interesting 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 compounding geometric crossovers do not need to be independent, for the cartesian crossover to be geometric.

2.6 Multi-parental geometric crossover

To extend geometric crossover to the case of multiple parents we need the following definitions.

A family $\mathcal{X}$ of subsets of a set X is called *convexity on* X if: (C1) the empty set $\emptyset$ and the universal set X are in $\mathcal{X}$, (C2) if $\mathcal{D} \subseteq \mathcal{X}$ is non-empty, then $\bigcap \mathcal{D} \in \mathcal{X}$, and (C3) if $\mathcal{D} \subseteq \mathcal{X}$ is non-empty 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: $co(A) = \bigcap\{C | A \subseteq C \in \mathcal{X}\}$. 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 iff it includes $co(F)$ for each finite subset F of C. The convex hull operator applied to set of cardinality two is called *segment operator*. Given a metric space $M = (X, d)$ the segment between a and b is the set $[a, b]_d = \{z \in X | d(x, z) + d(z, y) = d(x, y)\}$. The abstract *geodetic convexity* $\mathcal{C}$ on X induced by M is obtained as follow: 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 $\forall a, b \in X : [a, b]_d \subseteq co\{a, b\}$. 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 [16]:

DEFINITION 1. *(Multi-parental geometric crossover) In a multi-parental geometric crossover, given n parents $p_1, p_2, \ldots, p_n$ their offspring are contained in the metric convex hull of the parents $\mathcal{C}(\{p_1, p_2, \ldots, p_n\})$ for some metric d*

THEOREM 2. *(Decomposable three-parent recombination) Every multi-parental 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*

3. GEOMETRIC PSO

3.1 Basic, Canonical PSO Algorithm and Geometric Crossover

Consider the canonical PSO in Algorithm 1. The main feature that allows the motion of particles is the ability to perform linear combinations of points in the search space. To obtain a generalisation of PSO to generic search spaces, we can achieve this same ability by using multiple (geometric) crossover operations.

Algorithm 1 Standard PSO algorithm

1: **for all** particle i **do**
2: initialise position $x_i \in U[\mathbf{a}, \mathbf{b}]$ and velocity $v_i = \mathbf{0}$
3: **end for**
4: **while** not converged (optimum of current objective function is not found) **do**
5: **for all** particle i **do**
6: set personal best $\hat{x}_i$ as best position found so far from the particle (best of current and previous positions)
7: set global best $\hat{g}$ as best position found so far from the whole swarm (best of personal bests)
8: **end for**
9: **for all** particle i **do**
10: update velocity using equation

$$v_i(t+1) = \omega v_i(t) + \phi_1 R_1(\hat{g}(t) - x_i(t)) + \phi_2 R_2(\hat{x}_i(t) - x_i(t)) \quad (1)$$

11: update position using equation

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

12: **end for**
13: **end while**

In the following we illustrate the parallel between geometric crossover and the motion of a particle (see Figure 2). Geometric crossover picks offspring C on a line segment between parents A, B. Geometric crossover can be interpreted as a motion of a particle: consider a particle P that moves in the direction to a point D being in the next time step in position P'. If now 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

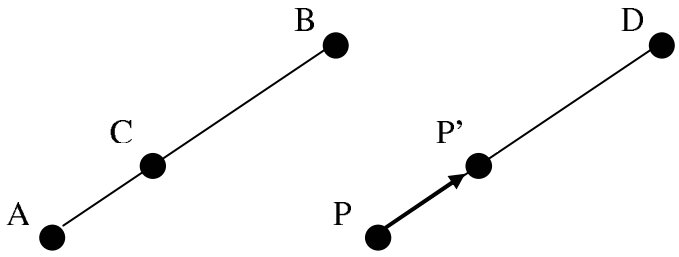

Figure 2: Geometric crossover and particle motion.

distance between parent A and offspring C is the intensity 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 geometric crossover, this means that the offspring C replaces its parent A in the new population. Since at a given time all particles move, each particle is selected for mating. Mating is a weighted multi-recombination involving the memory of the particle and the best in the current population. Weights are the propensity of a particle towards memory, sociality, stability.

We explain these ideas in detail in the following sections.

3.2 Geometric interpretation of linear combinations

In the following we first give some necessary preliminary notions on linear combinations and their geometric interpretation. Then, we use these to show that when the momentum term in the velocity update equation of PSO (equation(1)) is zero we can characterize the dynamic of PSO as a simple linear combination of the positions of the particle, particle best and swarm best without making explicit use of the particle velocity. This opens the way to the generalization of PSO to generic metric spaces presented in the next sections.

If $v_1, ..., v_n$ are vectors and $a_1, ..., a_n$ are scalars, then the *linear combination* of those vectors with those scalars as coefficients is : $a_1v_1 + a_2v_2 + a_3v_3 + \cdots + a_nv_n$. A linear combination on n linearly independent vectors spans completely a n-dimensional or lower dimensional space but not a higher dimensional one. So, the linear combination of three linearly independent points spans all a 3-dimensional space but not a 4-dimensional one.

An *affine combination* of vectors $x_1, ..., x_n$ is a linear combination $\sum_{i=1}^{n} \alpha_i \cdot x_i = \alpha_1x_1 + \alpha_2x_2 + \cdots + \alpha_nx_n$ in which the sum of the coefficients is 1, thus: $\sum_{i=1}^{n} \alpha_i = 1$. When a vector represents a point in space, the affine combination of 2 points spans completely the line passing through them; the affine combination of 3 points spans completely the plane (2D line) passing through them; increasing number of points spans completely higher dimensional "lines".

A *convex combination* is a linear combination of vectors where all coefficients are non-negative and sum up to 1. It is called "convex combination", since, when a vector represent a point in space, all possible convex combinations (given the base vectors) will be within the convex hull of the given points. In fact, the set of all convex combinations constitutes the convex hull.

A special case is with only two points, where the value of the new point (formed by the convex combination) will lie on a straight line between the two points. For three points, their convex hull is the triangle with the points as vertices.

THEOREM 3. *In a PSO with no momentum ($\omega = 0$) and where learning rates are such that $\phi_1 + \phi_2 < 1$, the future position of each particle x' is within the triangle formed by its current position x, its local best $\hat{x}$ and the swarm best $\hat{g}$. Furthermore, x' can be expressed without involving the particle's velocity as $x' = (1 - w_2 - w_3)x + w_2\hat{x} + w_3\hat{g}$.*

In the next section, we generalize this simplified form of PSO from real vectors to generic metric spaces. 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. So 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 be easily understood as points in a metric space. However, for scalars their interpretation 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 the weight to one, the stronger the attraction to its corresponding point. The resulting point of the convex combination can be seen as a weighted spatial average and it is 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 $d(x_1, y) \sim 1/w_1$, $d(x_2, y) \sim 1/w_2$ and $d(x_3, y) \sim 1/w_3$.

This definition is formal and valid for all metric spaces but it is non-constructive. In contrast a convex combination, not only defines a convex hull, but it tells also 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 in the next section.

The requirements for a convex combination in a metric space are:

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, $\hat{x}$ and $\hat{g}$ and returns a point in their metric 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 weight (so in a equilateral triangle, if the coefficients have all the same value, the distance to the equilibrium point are the same)

3.4 Geometric PSO algorithm

The generic Geometric PSO algorithm is illustrated in Algorithm 2. This differs from the standard PSO (Algorithm 1) in that: there is no velocity, the equation of position update is the convex combination, there is mutation and the parameters ω, ϕ_1, and ϕ_2 are positive and sum up to one.

Algorithm 2 Geometric PSO algorithm

for all particle i **do**
  initialise position x_i at random in the search space
end for
while stop criteria not met **do**
  for all particle i **do**
    set personal best $\hat{x}_i$ as best position found so far by the particle
    set global best $\hat{g}$ as best position found so far by the whole swarm
  end for
  for all particle i **do**
    update position using a randomized convex combination

$$x_i = CX((x_i, \omega), (\hat{g}, \phi_1), (\hat{x}_i, \phi_2)) \quad (3)$$

    mutate x_i
  end for
end while

In previous work we have specified this algorithm to Euclidean, Manhattan and Hamming spaces. In the following we show how to specify it to general representations in a straightforward way.

3.5 Geometric PSO for general representations

Before introducing how to extend geometric PSO for other solution representations, we will discuss the relation between 3-parental geometric crossover and the symmetry requirement for a convex combination.

We could consider, or define, a three-parental recombination and then prove that it is a three-parental geometric crossover by showing that it can be actually decomposed into two sequential applications of a geometric crossover for the specific space.

Alternatively, we could skip altogether the *explicit definition* of a three-parental recombination. In fact to obtain the three-parental recombination we could use 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.

The benefit of defining 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 exchanging in the three-parental recombination both positions and 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 weights w_a, w_b and w_c positive and adding 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: $\Delta GX((a, w_a), (b, w_b), (c, w_c)) =$
$GX((GX((a, w'_a), (b, w'_b)), w_{ab}), (c, w'_c))$ with the constraint that w'_a and w'_b positive and adding up to one and w_{ab} and w'_c positive and adding up to one. It is immediate to notice that there is not inherent symmetry in this expression: the weights w'_a and w'_b are not directly comparable with w'_c because are relative weights between a and b. Moreover there is the extra weight w_{ab}. This makes problematic the requirement of symmetry: given the desired w_a, w_b and w_c, what values of w'_a, w'_b, w_{ab} and w'_c do we have to choose to obtain an equivalent symmetric 3-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: $\Delta GX = w_a \cdot a + w_b \cdot b + w_c \cdot c = (w_a + w_b)(\frac{w_a}{w_a+w_b} \cdot a + \frac{w_b}{w_a+w_b} \cdot b) + w_c \cdot c$. 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 $\Delta GX((a, w_a), (b, w_b), (c, w_c)) = GX((GX((a, \frac{w_a}{w_a+w_b}), (b, \frac{w_b}{w_a+w_b})), w_a + w_b), (c, w_c))$. Although this question may be less straightforward to answer for other spaces, 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.

Where does this discussion leave us about the extension of geometric PSO 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 anew for the new representation and then prove its geometricity by showing that it is decomposable into a sequence of two two-parental geometric crossovers (ii) use twice the simple geometric crossover to produce a symmetric or nearly symmetric three-parental recombination. The second option is indeed very interesting because *it allows us to extended automatically to geometric PSO all representations we have geometric crossovers for, such as permutations, GP trees, variable-length sequences, to mention few, and virtually any other complex solution representation.*

4. GEOMETRIC PSO FOR SUDOKU

In this section we will put into practice the ideas discussed in section 3.5 and propose a geometric PSO to solve the Sudoku puzzle. In section 4.1 we present a geometric crossover for Sudoku. In section 4.2 we present a three parental crossover for Sudoku and show that is a convex combination.

4.1 Geometric crossover for Sudoku

Sudoku is a constraint satisfaction problem with 4 types of constraints:

1. Fixed elements
2. Rows are permutations
3. Columns are permutations
4. 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 fulfillment is the fitness of the solution. We consider a space with the following characteristics:

- *Hard constraints*: fixed positions and permutations on rows
- *Soft constraints*: permutations on columns and boxes
- *Distance*: sum of swap distances between paired rows (row-swap distance)
- *Feasible geometric mutation*: swap two non-fixed elements in a row
- *Feasible geometric crossover*: row-wise PMX and row-wise cycle crossover

This mutation preserves both fixed positions and permutations on rows because swapping elements within a row that 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 return always 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, it is immediate 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.

Fitness function (to maximize): sum of number of unique elements in each row, plus, sum of number of unique elements in each column, plus, sum of 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.

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.

4.2 Convex combination for Sudoku

In the following we first define a multi-parental 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 following example to illustrate how the *multi-parental sorting crossover* works.

```
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
```

The mask is generated at random and is a vector of the same length of the parents. The number of 1s, 2s and 3s 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 1. This means that at position 1 parent 2 and parent 3 have to become equal to parent 1. This is done by swapping the element 1 and 3 in parent 2 and the element 1 and 3 in parent 3. The recombination now continues on the updated parents: parent 1 is left unchanged and current parent 2 and parent 3 are the original parent 2 and 3 after the swap. At position 2 the mask has 2. This means that at position 2 current parent 1 and current parent 3 have to become equal to current parent 2. So at position 2, parent 1 and parent 3 have to get 5. To achieve this, in parent 1 we need to swap elements 2 and 5 and in parent 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. So after all positions have been considered, all elements are fixed. The permutation to which the three parents converged is the offspring permutation. So, this recombination sorts by swaps the three parents towards each others 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 4. *(Geometricity of three-parental sorting crossover) Three-parental sorting crossover is geometric crossover under swap distance*

Proof sketch: A three-parental sorting crossover with recombination mask m_{123} is equivalent to a sequence of two two-parental sorting crossovers: the first between parent 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 5. *(Coherence between weights and distances) In weighted multi-parent sorting crossover, the swap distances of the parents to the expected offspring are decreasing functions of the corresponding weights.*

Proof sketch: 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 multi-parental 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 4), coherence (theorem 5) and symmetry is 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. This theorem extends to the case of a multi-parent geometric crossover.

THEOREM 6. *(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 sketch: 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 in general, 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 [20]. The product geometric crossover then naturally generalizes to the multi-parent case.

As explained in section 3.5 there are two alternative ways of producing a convex combination: either using a convex combination operator or simply apply twice a 2-parental weighted recombination with appropriate weights to obtain the convex combination.

Table 1: Average of bests of 50 runs with population size 100, lattice topology and mutation 0.0 varying sociality (vertical) and memory (horizontal).

Soc/Mem	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 2: Average of bests of 50 runs with population size 100, lattice topology and mutation 0.3 varying sociality (vertical) and memory (horizontal).

Soc/Mem	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

5. EXPERIMENTAL RESULTS

In order to test the efficacy of the geometric PSO 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 geometric PSO algorithm given in section 3.4 with the search operators for Sudoku presented in section 4.2.

The parameters we varied were swarm sociality and memory, each of which were in turn set to 0, 0.2, 0.4, 0.6, 0.8 and 1.0. As the inertia is defined as (1 - sociality - memory) 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 100000 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. Both ways to produce convex combination operators, explicit and implicit, were tried and turned out to produce indistinguishable results.

5.1 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 topology. The other two topologies we studied were worse.

From tables 1, 2 and 3, 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 inertia set to more than 0.4 generally perform badly. The best configurations

Table 3: Average of bests of 50 runs with population size 100, lattice topology and mutation 0.7 varying sociality (vertical) and memory (horizontal).

Soc/Mem	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 4: Success rate of various methods.

Method	Success
GA	50/50
Hillclimber	35/50
PSO-Global	7/50
PSO-ring	20/50
PSO-von Neumann	36/50

generally have sociality set to 0.2 or 0.4, memory set to 0.4 or 0.6, and inertia 0 or to 0.2. This gives us some indication of the importance of the various types of recombinations in PSO as applied at least to this particular problem. Note that the only type of recombination that uniquely differentiates PSO from evolutionary algorithms is the inertia, and that the best results for this problem were found with very low or no inertia. In the case of inertia set to 0, PSO in fact degenerates to a type of genetic algorithm with local selection between parents and offspring.

5.2 PSO vs EA

Table 4 compares the success rate of the best configurations of various methods we have tried in this and our previous paper. Success is here defined as in how many runs (out of 50) 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 PSO with this topology can achieve a respectable success rate on this tricky non-continuous problem. However, the best genetic algorithm still significantly outperforms the best PSO we have found so far. We believe this at least partly to be the effect of the even more extensive tuning of parameters and operators undertaken in our GA experiments.

6. CONCLUSIONS AND FUTURE WORK

Geometric PSO 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 geometric particle swarm optimization algorithm to the space of Sudoku grids (vectors of permutations). This is the first time that a PSO algorithm has been successfully applied to a non-trivial combinatorial space. This shows that geometric PSO is indeed a natural and promising generalization of classical PSO. In future work we will consider geometric PSO for the space of genetic programs.

7. REFERENCES

[1] T. Bäck and D. B. Fogel and T. Michalewicz. Evolutionary Computation 1: Basic Algorithms and Operators. *Institute of Physics Publishing, 2000.*

[2] D. Bratton and J. Kennedy. *Defining a Standard for Particle Swarm Optimization*, Proceedings of EuroGP, 2006.

[3] M. Clerc. Discrete Particle Swarm Optimization, Illustrated by the Traveling Salesman Problem. *New Optimization Techniques in Engineering, Springer, 2004.*

[4] M. Deza and M. Laurent. *Geometry of cuts and metrics*. Springer, 1991.

[5] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[6] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, 1995.

[7] J. Kennedy and R.C. Eberhart. *A Discrete Binary Version of the Particle Swarm Algorithm*, IEEE, 1997.

[8] J. Kennedy and R.C. Eberhart. *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001.

[9] D. E. Knuth. Dancing links. *Preprint P159, Stanford University*, 2000.

[10] A. Moraglio and J. Togelius and S. Lucas. Product Geometric Crossover for the Sudoku Puzzle. *Proceedings of IEEE Congress on Evolutionary Computation, 2006.*

[11] A. Moraglio and R. Poli. Product geometric crossover. *Proceedings of Parallel Problem Solving from Nature Conference, 2006.*

[12] A. Moraglio and R. Poli. Topological interpretation of crossover. *Proceedings of Genetic and Evolutionary Computation Conference, 2004.*

[13] A. Moraglio and R. Poli. Geometric landscape of homologous crossover for syntactic trees. *Proceedings of IEEE Congress on Evolutionary Computation, 2005.*

[14] A. Moraglio and R. Poli. Topological crossover for the permutation representation. *GECCO 2005 Workshop on Theory of Representations, 2005.*

[15] A. Moraglio and R. Seehuus and R. Poli. Geometric crossover for biological sequences. *Proceedings of European Conference on Genetic Programming, 2006.*

[16] A. Moraglio and C. Di Chio and R. Poli. Geometric Particle Swarm Optimization. *Proceedings of European Conference on Genetic Programming, 2007.*

[17] A. Moraglio and R. Poli. Geometric Crossover for Sets, Multisets and Partitions. *Proceedings of Parallel Problem Solving from Nature Conference, 2006.*

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

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

[20] M. L. J. Van de Vel. Theory of Convex Structures. *North-Holland, 1993.*