Genetic Algorithm Theory

Jonathan E. Rowe

University of Birmingham, UK

GECCO 2012

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

Introduction

The theory of genetic algorithms is beginning to come together into a coherent framework. However, there are still many gaps, and much that we do not know.

This tutorial will provide an introduction to the framework, and describe some of the areas of development. I will include pointers to other research directions, some of which are covered in other tutorials.

Overview of tutorial

The main topics covered in this tutorial are:

- Genetic algorithms as Markov chains.
- Finite and infinite populations.
- Selection, mutation and crossover.
- Schemata, projections and coarse-grainings.
- Generalisation to other search spaces.

The Simple Genetic Algorithm

A basic object of study is the Simple Genetic Algorithm. At any time-step (or generation) there is a population (of size N) of binary strings (of length ℓ). We use *genetic operators* (selection, crossover, mutation) to produce the next population.

We will consider generalisations to other search spaces later.

Producing the next population

To produce the next population we follow these steps N times:

1. Select two items from the population.
2. Cross them over to form an offspring.
3. Mutate the offspring.
4. Insert the result into the new population.

Markov chains

There are a finite number of possible populations of size N. The probability of producing a particular population in one generation, depends only on the population at the previous generation.

This kind of random process is called a Markov chain. It can be characterised by a *transition matrix* Q.

$Q_{\mathbf{q},\mathbf{p}}$ is the probability of going from population $\mathbf{p}$ to population $\mathbf{q}$.

Populations as distributions

We can think of a population $\mathbf{p}$ as a distribution over the search space Ω.

p_k is the proportion of copies of item k in the population. That is, it is the number of copies of k divided by N.

$$\mathbf{p} \in \Lambda = \left\{ \mathbf{x} \in \mathbb{R}^n | \sum_k x_k = 1 \text{ and } x_k \geq 0 \text{ for all } k \right\}$$

The heuristic function

If the current population is $\mathbf{p}$, then there is a certain probability of producing item k in the next generation. Call this probability $\mathcal{G}(\mathbf{p})_k$.

That is, $\mathcal{G}(\mathbf{p}) \in \Lambda$ and $\mathcal{G}$ can be thought of as a map

$$\mathcal{G} : \Lambda \longrightarrow \Lambda$$

which we call the *heuristic function*.

Random Heuristic Search

An equivalent way of characterising a single generation of the Simple Genetic Algorithm is as follows:

1. $\mathbf{p}$ is the current population
2. Calculate $\mathcal{G}(\mathbf{p})$.
3. Take N random samples from $\mathcal{G}(\mathbf{p})$.
4. The sample is the next population.

The transition matrix

If we know the heuristic function $\mathcal{G}$, we can write down the transition matrix.

$$Q_{\mathbf{q},\mathbf{p}} = N! \prod_{k \in \Omega} \frac{(\mathcal{G}(\mathbf{p})_k)^{Nq_k}}{(Nq_k)!}$$

This is called a *multinomial* distribution.

The expected next population

The next population is a random variable. If $\mathbf{p}$ is the current population, then we can calculate what the *expected* next population is.

This turns out to be simply $\mathcal{G}(\mathbf{p})$.

The variance

We can also ask for the variance of the distribution. It is:

$$\frac{1 - \mathcal{G}(\mathbf{p})^T \mathcal{G}(\mathbf{p})}{N}$$

You can see that the variance decreases as N increases. That means that for large population sizes, the next generation is likely to be close to $\mathcal{G}(\mathbf{p})$.

The infinite population model

As $N \to \infty$, we can see that the variance decreases to zero. So the behaviour of the random heuristic search becomes closer and closer to simply iterating $\mathcal{G}$.

More formally, given a number of time steps τ, and distance $\epsilon > 0$ and a probability $\delta > 0$, there exists a population size N so that the random heuristic search algorithm stays within ϵ of the iterates of $\mathcal{G}$ for τ time steps, with probability greater than $1 - \delta$.

Finite population behaviour

We know that, if N is big enough, a finite population will stay close to the trajectory of $\mathcal{G}$ in the short-term. We are also interested in:

- In what states does it spend most of its time (in the medium and long term)?
- How long does it take to reach a particular state (e.g. one containing the optimal solution to a problem)?

We will take a look at the first problem. The second problem can only be addressed for specific cases.

Metastable states

A genetic algorithm often seems to "stall" in certain states. This is particular obvious with some fitness functions (e.g. Royal Road functions).

These states are called *metastable* and correspond to sets of populations which have the property that $\|\mathcal{G}(\mathbf{p}) - \mathbf{p}\|$ (called the *force* of $\mathcal{G}$ at $\mathbf{p}$) is small.

Fixed-points and metastability

One place we find populations with small force is close to fixed-points of $\mathcal{G}$ (at which the force is, of course, zero). These fixed-points may themselves be outside, but close to Λ, since $\mathcal{G}$ remains well-defined and continuous.

It is also sometimes possible to find much larger sets of populations, which the GA appears to wander around at random. These are called *neutral networks* as they have (almost) constant fitness.

Genetic operators and $\mathcal{G}$

We will now look at the different genetic operators: selection, mutation, crossover. We will look at them singly, and in various combinations.

For each combination, we will write down the corresponding heuristic function $\mathcal{G}$. We will then consider what is known about the fixed-points of $\mathcal{G}$ and the behaviour of (finite population) algorithms.

Selection and fitness

A *selection* operator chooses items from the current population. It makes use of a *fitness function*, which is a function

$$f : \Omega \longrightarrow \mathbb{R}^+$$

We can also think of the fitness function as a vector $\mathbf{f} \in \mathbb{R}^n$, with $f_k = f(k)$.

We usually require a selection operator to assign a higher probability to elements that have higher fitness.

Proportional selection

One common method of selection is to select elements with probability proportional to their fitness. This corresponds to a heuristic function

$$\mathcal{F}(\mathbf{p}) = \frac{\text{diag}(\mathbf{f})\mathbf{p}}{\mathbf{f}^T p}$$

where $\text{diag}(\mathbf{f})$ is a diagonal matrix with the elements of $\mathbf{f}$ along the diagonal.

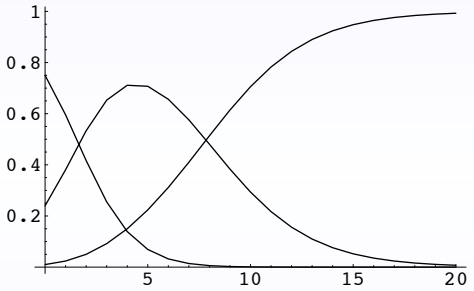
Trajectory of proportional selection

Given an initial population $\mathbf{p}(0)$, let $\mathbf{p}(t) = \mathcal{F}^t(\mathbf{p})$ be the corresponding trajectory of the infinite population model.

It can be shown that:

$$\mathbf{p}(t)_k = \frac{(f_k)^t \mathbf{p}(0)_k}{\sum_j (f_j)^t \mathbf{p}(0)_j}$$

Example trajectory — proportional selection



The proportion of each solution in the population for the trajectory, starting with $\mathbf{p}(0) = (0.01, 0.24, 0.75)$. The fitness function if $\mathbf{f} = (3, 2, 1)$.

Fixed-points and absorbing states

The fixed-points of $\mathcal{F}$ are the *homogeneous* populations. That is, populations that have copies of just one item. They are represented by vectors with a 1 in one position and zeros elsewhere.

The trajectory will converge to the homogeneous population containing copies of the fittest individual in the original population.

Homogeneous populations are also absorbing states of the Markov chain for finite populations.

Binary tournament selection

An alternative selection method is to choose two items from the population uniformly at random, and then keep the best one (as measured by the fitness function). The corresponding heuristic function is[1]

$$\mathcal{F}(\mathbf{p})_i = (p_i)^2 + 2p_i \sum_j p_j [f(j) < f(i)]$$

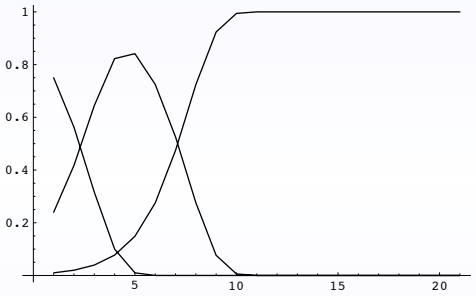
[1] $[expr]$ equals 1 if $expr$ is true, and 0 otherwise.

Behaviour of tournament selection

This function is harder to analyse, as it is quadratic. However, its fixed-points are again the homogeneous populations (as are the absorbing states of the Markov chain).

It is sometimes said that tournament selection is *stronger* than proportional selection, as it moves more quickly towards a homogeneous population.

Example trajectory — tournament selection



The proportion of each solution in the population for the trajectory, starting with $\mathbf{p}(0) = (0.01, 0.24, 0.75)$. The fitness function if $\mathbf{f} = (3, 2, 1)$.

Mutation

In general, mutating a particular item can produce any other item (with some probability). Write $U_{i,j}$ to be the probability that item j mutates to item i. Then the heuristic function for mutation is

$$\mathcal{U}(\mathbf{p}) = U\mathbf{p}$$

Example — mutation by a rate

For example, if we mutate binary strings of length ℓ bitwise, with mutation rate u, then

$$U_{i,j} = u^{d(i,j)}(1-u)^{\ell - d(i,j)}$$

where $d(i,j)$ is the Hamming distance between i and j.

Mutation commutes with exclusive-or

Standard mutation on binary strings satisfies the following curious property:

$$U_{i,j} = U_{k \oplus i, k \oplus j}$$

for all $i, j, k \in \Omega$, where $\oplus$ represents bitwise exclusive-or. We say that it *commutes* with this operator.

This means that if you mutate $k \oplus i$, you get the same result as if you mutated i by itself (to form j) and then formed $k \oplus j$.

Long-term behaviour of mutation

Any mutation operator that commutes with $\oplus$ has a unique fixed-point in Λ, which is the uniform distribution overΩ . We denote this distribution by $(1/n)\mathbf{1}$ (where $\mathbf{1}$ is the vector of all ones).

$$(U(1/n)\mathbf{1})_i = (1/n)\sum_j U_{i,j} = (1/n)\sum_j U_{j\oplus i,0} = (1/n)\sum_k U_{k,0} = 1/n$$

The corresponding Markov chain is irreducible, and the stationary distribution is uniform over all populations. That is, all populations are visited equally often.

Characterising mutation

We can characterise all mutations which commute with $\oplus$ as follows. For any probability distribution $\alpha \in \Lambda$, set

$$U_{i,j} = \alpha_{j\oplus i}$$

Then

$$U_{k\oplus i,k\oplus j} = \alpha_{k\oplus j\oplus k\oplus i} = \alpha_{j\oplus i} = U_{i,j}$$

Such a mutation can be implemented as follows: to mutate j, pick an element $k \in \Omega$ according to distribution α and form $j \oplus k$.

Proportional selection and mutation

If we first perform proportional selection, and then mutation, we get a combined heuristic function

$$\mathcal{G}(\mathbf{p}) = (\mathcal{U} \circ \mathcal{F})(\mathbf{p}) = \frac{U\text{diag}(\mathbf{f})\mathbf{p}}{\mathbf{f}^T\mathbf{p}}$$

The fixed-points of this function are easily found. They must satisfy

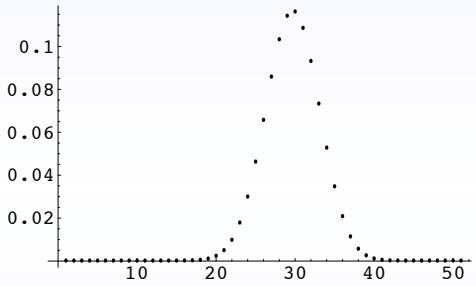
$$U\text{diag}(\mathbf{f})\mathbf{p} = (\mathbf{f}^T\mathbf{p})\mathbf{p}$$

Perron-Frobenius theorem

The fixed-points are therefore *eigenvectors* of the matrix $U\text{diag}(\mathbf{f})$. The corresponding eigenvalues are the average fitness at the fixed-point.

The Perron-Frobenius theorem tells us that exactly one of these fixed-points is in Λ, corresponding to the largest eigenvalue. The others are outside Λ but may still influence the dynamics inside.

Example — OneMax



The fixed-point population, shown as a distribution over unitation classes, for the OneMax function with $\ell = 50$, $u = 0.05$ and no crossover. The corresponding average fitness is 28.65.

Transitory behaviour

The Markov chain corresponding to a selection+mutation genetic algorithm is irreducible, and therefore has a stationary distribution. It visits all populations infinitely often, but some more frequently than others.

For particular fitness functions, it is possible to analyse some aspects of the transient behaviour, such as the expected time to reach the optimum solution (see Wegener et al.).

Metastable states

The states in which the algorithm spends a lot of its time can sometimes be found by analysing the fixed-points of $\mathcal{G}$.

One needs to take into account fixed-points inside and outside Λ as they can all have an effect. This can enable us to establish metastable states and neutral networks.

Example — fixed-points

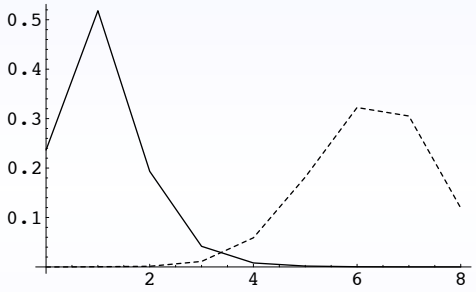
For example, consider the function of unitation

$$\mathbf{f} = (9, 8, 1, 2, 3, 4, 5, 6, 7)$$

(a *trap* function), with mutation rate 0.05.

We find there is one fixed-point in Λ, around the optimum, and another just outside, but near the false optimum.

Example — metastable states



The fixed-point population in Λ (solid line) and one just outside (dashed line). The fitness function is $\mathbf{f} = (9, 8, 1, 2, 3, 4, 5, 6, 7)$ (as a function of unitation). Mutation rate is 0.05.

Crossover

Crossover takes two individuals and produces an offspring. For each $i, j, k \in \Omega$ we need to specify the probability that crossing i and j will produce k. We denote this probability $r(i, j, k)$.

The corresponding heuristic function is

$$\mathcal{C}(\mathbf{p})_k = \sum_{i,j} p_i p_j r(i, j, k)$$

Crossover by masks

Crossing two binary strings is usually done by selecting a *binary mask* according to some probability distribution. The mask indicates which bits come from the first parent and which come from the second.

In this case we get

$$\mathcal{C}(\mathbf{p})_k = \sum_{i,j} p_i p_j \sum_b \chi_b [(b \otimes i) \oplus (\bar{b} \otimes j) = k]$$

where χ_b is the probability of picking mask b.

Crossover commutes with exclusive-or

Crossover by masks satisfies the following curious property:

$$r(a \oplus i, a \oplus j, a \oplus k) = r(i, j, k)$$

for all $i, j, k, a \in \Omega$. We say that it *commutes* with this operator.

This means that if you cross $a \oplus i$, with $a \oplus j$, you get the same result as if you crossed i with j (to form k) and then formed $a \oplus k$.

The mixing matrix

This means that crossover is determined by the matrix

$$M_{i,j} = r(i, j, 0)$$

called the *mixing matrix*. We can find all the other probabilities from this:

$$r(i, j, k) = r(k \oplus i, k \oplus j, 0) = M_{k \oplus i, k \oplus j}$$

Group representation The set of binary strings Ω forms a *group* under the operator $\oplus$.

For each element $k \in \Omega$, we can define a corresponding matrix σ_k

$$(\sigma_k)_{i,j} = [i = k \oplus j]$$

This set of matrices forms a group (under normal matrix multiplication) which is *isomorphic* to $(\Omega, \oplus)$.

The crossover heuristic

We can now re-write the crossover heuristic function, in terms of the mixing matrix, and the group of permutation matrices:

$$\mathcal{C}(\mathbf{p})_k = \mathbf{p}^T \sigma_k M \sigma_k^T \mathbf{p}$$

Characterising crossover

All crossovers which commute with $\oplus$ can be characterised as follows.

1. Start with any row stochastic matrix, A.
2. Define a new matrix $M_{i,j} = A_{i \oplus j, i}$.
3. Let M be the mixing matrix defining $\mathcal{C}$.

The twist

The mixing matrix M is called the *twist* of the matrix A. Twists play an important role in the study of crossover. For example, the derivative $d\mathcal{C}_{\mathbf{x}}$ of the heuristic function at a point $\mathbf{x}$ is the matrix

$$d\mathcal{C}_{\mathbf{x}} = 2\sum_{k} \sigma_k M^* \sigma_k^T x_k$$

where M^* is the twist of the mixing matrix.

Linkage equilibrium

Geiringer's theorem is a classic result from population genetics. It describes the fixed-points of $\mathcal{C}$ defined by masks.

The set of fixed-points are populations that are in *linkage equilibrium*. This means that if you take a string at random from such a population, the probability of finding a 1 at any position is *independent* of the contents of the rest of the string.

Geiringer's theorem

The theorem also says that if you start with some population vector $\mathbf{p}$ and repeatedly apply $\mathcal{C}$, then the limit will be one of these fixed-points.

The probability of finding a 1 in a given bit position at this fixed-point will be simply equal to the proportion of 1s in that position in the initial population.

Finite population behaviour

Geiringer's theorem was originally proved in the infinite population model. A version of the theorem has recently been proved for finite populations.

In this version, we look at the stationary distribution of the Markov chain. Select a population at random from this distribution, and pick a string at random. The probability of finding a 1 in any given position depends only on the proportion of 1s in that position in the initial population.

Selection and crossover

Suppose we have a genetic algorithm with proportional selection and crossover. Then $\mathcal{G} = \mathcal{C} \circ \mathcal{F}$.

Homogeneous populations are fixed-points. They are also absorbing states of the Markov chain. However, there can be other fixed-points.

It is conjectured that the only *asymptotically stable* fixed-points are homogeneous populations. Proving this is an open problem.

Fixed-points and local optima

Some of the homogeneous populations are asymptotically stable, and some are unstable.

Suppose we have an asymptotically stable population that contains only copies of a bitstring k. Then if we consider the fitness of k, and the fitness of all its Hamming neighbours we find that:

Asymptotically stable homogenous populations contain (Hamming) local optima.

Genepool crossover

There is another form of crossover sometimes used call *genepool* or *population* crossover. It is used, for example, in UMDA.

In each bit position, this operator shuffles the bits in the entire population. It brings the population immediately to linkage equilibrium.

It is like performing lots of normal crossovers between each selection.

Fixed-points of genepool crossover

When we use genepool crossover and proportional selection, then it can be proved that:

1. The only asymptotically stable fixed-points are homogeneous populations.
2. Moreover, they correspond exactly with the Hamming local optima.
3. This follows from the fact that average fitness increases monotonically.

Mutation and crossover

Suppose we have mutation and crossover, but no selection. The heuristic function is again a quadratic operator $\mathcal{M} = \mathcal{C} \circ \mathcal{U}$ which commutes with $\oplus$.

The mixing matrix is $U^T M U$, where U is the mutation matrix and M is the mixing matrix of crossover alone.

If mutation is by a rate, then it doesn't matter which order we do mutation and crossover — the effect is the same.

Long-term behaviour

As long as there is a positive mutation rate, $\mathcal{M}$ has a unique fixed-point $(1/n)\mathbf{1}$.

The corresponding Markov chain is irreducible. Its stationary distribution is uniform over all possible (ordered) populations of size N. This means that all strings are equally likely to appear.

We see that in both the finite and infinite population case, a vanishingly small (but non-zero) mutation rate is sufficient to destroy the fixed-points described by Geiringer's theorem.

Schemata

A *schema* is a pattern of bit-values at certain defining positions. For example:

$$1 * * 0 * * 0 *$$

is the set of all strings which have a 1 in the first position, and 0s in the fourth and seventh positions. The $*$ symbol acts as a "don't care".

Schemata are important for understanding crossover and mutation.

Schema families

A *schema family* is a collection of schemata, which all have the same defining positions (and the same "don't care" positions). For example:

$$* * * 0 * * 0 *$$
$$* * * 0 * * 1 *$$
$$* * * 1 * * 0 *$$
$$* * * 1 * * 1 *$$

The Schema Projection Theorem

Mutation and crossover done by masks can be *projected* onto a schema family. That is, one can define a corresponding crossover and mutation that act only on the defining bit positions of the family.

The effects of crossover and mutation on those bits is independent of what happens on the other bits.

Coarse-graining

This kind of effect is called a *coarse-graining*, for it enables us to study the effects of operators on aggregated chunks of the search space.

The Schema Projection theorem tells us that crossover and mutation can be exactly coarse-grained using schemata.

Selection and schemata

It would have been convenient if selection could also be projected onto schema families. Then we could have tracked the entire GA on aggregated "building blocks". Unfortunately this can't be done in a time-independent manner.

That is, the building blocks at one generation are different from those at another.

Coarse-graining at a fixed-point

If we start at a fixed-point $\mathbf{p} = \mathcal{G}(\mathbf{p})$, then nothing changes from one time-step to another. At such a point it *is* possible to coarse-grain selection (as well as crossover and mutation) using schemata.

This is because the "fitness" of a schema can no longer change from one time-step to another, and so we can aggregate in a time-independent manner.

Order-1 schemata fitness

Let $\mathcal{G}$ include proportional selection, crossover and mutation (with rate u). Specify some bit position, and some population $\mathbf{p}$.

- Let $q_0(\mathbf{p})$ be the proportion of the population with a 0 at that position.
- Let $q_1(\mathbf{p})$ be the proportion with a 1. Clearly $q_0 + q_1 = 1$.
- Let $f_0(\mathbf{p})$ be the average fitness of strings with a 0 at that position.
- Let $f_1(\mathbf{p})$ be the average fitness of strings with a 1.

Schemata fitness at a fixed-point

If $\mathbf{p}$ is a fixed-point of $\mathcal{G}$, then

$$\frac{f_0(\mathbf{p})}{f_1(\mathbf{p})} = \frac{q_1}{q_0}\left(\frac{q_0 - u}{q_1 - u}\right)$$

Notice that this result holds for any fitness function and for any kind of crossover (by masks).

Example

Let $\Omega = \{00, 01, 10, 11\}$ and $\mathbf{f} = (4.0, 4.1, 0.1, 4.11)$. Choose say, uniform crossover and a mutation rate of $u = 0.006$. We choose to focus on the second bit position.

The stable fixed-point is $\mathbf{p} = (0.032, 0.737, 0.005, 0.226)$.

Example (cont'd)

At this fixed-point we have:

$$\frac{f_0(\mathbf{p})}{f_1(\mathbf{p})} = 0.843$$

Notice that this means the schema $*1$ has higher fitness than $*0$, even though the system is in stable equilibrium.

Other coarse-grainings

We can determine exactly what coarse-grainings are possible:

- Crossover by masks can only be projected onto schema families.
- Mutation by a rate can be projected onto schema families and unitation classes.
- Proportional Selection can only be projected onto classes of equal fitness.
- Binary tournament selection can only be projected onto classes of contiguous fitness.

The Walsh Transform

The *Walsh Transform* is given by the matrix

$$W_{i,j} = \frac{(-1)^{i \cdot j}}{\sqrt{n}}$$

where $i \cdot j$ is the number of 1s that i and j have in common. Applying the transform gives us a *change of basis*.

Effects of the Walsh Transform

In the Walsh basis, crossover and mutation become considerably easier to deal with:

- The mutation matrix becomes diagonal.
- All the matrices σ_k become diagonal.
- The mixing matrix is untouched (but is generally sparse).

Applications of the Walsh Transform

Because of these simplifications, a number of results can be proved using the Walsh basis. For example:

- The operator $\mathcal{C}$ can be inverted.
- The twist is lower triangular in the Walsh basis.

Genepool crossover (with and without mutation) also becomes easier to analyse in the Walsh basis.

Selection and the Walsh basis

Proportional selection is not so nice in the Walsh basis. However, the following result is curious:

Let $S = \text{diag}(\mathbf{f})$, and let $\widehat{S} = WSW$. Then $\widehat{S}$ commutes with $\oplus$.

Selection appears to be a *dual* of mutation.

Generalising to other search spaces

Most of the existing theory of genetic algorithms has been worked out for the search space of fixed-length binary strings.

But what if we have a different search space (for example, the Travelling Salesman Problem)? How much of the theory can be generalised?

Search space groups

It turns out that quite a lot can be done in the case where the search space Ω, together with some operator $\oplus$ forms an arbitrary finite group.

For example, Ω could be the set of permutations of cities (in TSP) and $\oplus$ could be function composition.

Operators that commute with $\oplus$

Again we say that mutation commutes with $\oplus$ if

$$U_{k \oplus i, k \oplus j} = U_{i,j}$$

We also say that crossover commutes with $\oplus$ if

$$r(a \oplus i, a \oplus j, a \oplus k) = r(i, j, k)$$

Such operators can be completely characterised as before.

Mixing matrices

In this case, we can again define the crossover heuristic in terms of a mixing matrix

$$\mathcal{C}(\mathbf{p})_k = \mathbf{p}^T \sigma_k M \sigma_k^T \mathbf{p}$$

where the matrices $(\sigma_k)_{i,j} = [i = k \oplus j]$ form a group isomorphic to $(\Omega, \oplus)$.

Generalised schemata

Given a collection $\mathcal{N}$ of subgroups of Ω, then an $\mathcal{N}$-schema is a set of the form

$$a \oplus G = \{a \oplus g | g \in G\}$$

where $G \in \mathcal{N}$. That is, schemata are *cosets* of subgroups of Ω.

Example — TSP

A route in TSP is an ordering $\pi : \{1, 2, \ldots, m\} \rightarrow \{1, 2, \ldots, m\}$ of the m cities, where $\pi(i) =$ the ith city in the tour.

Choose some subset $C \subseteq \{1, 2, \ldots, m\}$ and consider permutations that have the property that $\pi(i) = i$ for all $i \in C$. These permutations form a subgroup.

Schemata in TSP

The schemata corresponding to such a subgroup are those tours which specify that certain cities must be visited at positions specified by C. For example, $*2*5*$ is the schema

$$(1,2,3,5,4), (1,2,4,5,3), (3,2,1,5,4), (3,2,4,5,1), (4,2,1,5,3), (4,2,3,5,1)$$

Respectfulness

Crossover is said to *respect* a subset of Ω if whenever we cross two elements of that subset, the offspring is guaranteed to also be in the subset.

It can be shown that if crossover respects all the subgroups of $\mathcal{N}$ then it also respects all $\mathcal{N}$-schemata.

Pure crossover

Crossover is said to be *pure* if crossing an element with itself always gives you the same element back again (that is, it is *cloning*).

It can be shown that crossover is pure if and only if it respects the subgroup $\{0\}$, where 0 is the identity element of the group.

Separative matrices

Given any $n \times n$ matrix A, we call it *separative* with respect to $\mathcal{N}$ if whenever $A_{i,j}$ ¿ 0, and i and j are both elements of some $\mathcal{N}$-schema, then the identity element 0 is also an element of that $\mathcal{N}$-schema.

Crossover respects all $\mathcal{N}$-schemata if and only if the mixing matrix is seperative.

Example — binary strings

For binary strings, this theorem means that for any crossover (that commutes with $\oplus$) to respect schemata, then if it is possible for i and j to produce the string of all zeros, the only bit-values at which they agree must also be zeros.

1 0 0 1 0 0 1 1
0 1 1 0 0 0 0 0
⇓
0 0 0 0 0 0 0 0

Example — TSP

The theorem can be used to show that a number of standard crossovers defined for TSP do not repsect the schemata that we have defined.

It is also possible to use the theorem to help design crossovers which do respect those schemata.

The Fourier Transform

Since the Walsh Transform is so helpful in unravelling the effects of crossover and mutation on binary strings, we would like to know if a generalised (Fourier) Transform exists for other groups. We have the following result:

A change of basis in which all the matrices σ_k are diagonal exists if and only if the search space group is commutative.

Commutative groups

In this case, elements of the group can be represented as strings over alphabets of possibly varying cardinality. This is because any finite commutative group can be written as a direct sum of *cyclic* groups.

For this type of search space, we can make use of crossover by masks and mutation by a rate in the standard way. Schemata correspond to the usual definition, and the Schema Projection Theorem still holds.

Group actions

It may not always be possible to define a natural group structure on a search space. In this case, it might be possible to define a *group action*.

This is a set of bijections on Ω, which form a group under function composition.

Transitivity

Much of the theory of genetic algorithms can be generalised to this situation.

The main requirement is that the group action is *transitive*, that is, given any two elements $i, j \in \Omega$, there is always a group element a such that $a(i) = j$.

Landscape graphs

It is interesting to ask how such group actions might naturally arise in the study of search problems. One possibility is to consider the properties of an associated *landscape* graph.

This would correspond, for example, to the Hamming cube in tha case of binary strings.

Neighbourhoods

Start by defining a *neighbourhood structure* on Ω. Every point in Ω has a subset of neighbours, such as one might use to define a *local search* algorithm.

The neighbourhood structure is *symmetric* if y is a neighbour of x implies x is a neighbour of y.

We make Ω into a graph by putting an edge between neighbouring elements.

Automorphisms

We now consider all permutations of Ω which preserve the graph structure. That is, $\pi : \Omega \rightarrow \Omega$ is such a permutation if whenever i and j are neighbours, so are $\pi(i)$ and $\pi(j)$.

This set of permutations forms a group action on Ω called the *automorphism* group.

Search space symmetries

The automorphism group characterises the different *symmetries* that exist in the landscape graph. The idea is that the labels we give to elements of the search space do not matter so much as the relationships between elements.

The study of landscapes and their relationship to genetic algorithms is a subject of current research.

Important research areas

- The development of systematic approximations (e.g. using statistical mechanics).
- Relating aspects of the infinite population model to time-complexity analysis for finite populations.
- Investigating the relationship of fitness landscapes to genetic operators.
- Generalisation to infinite search spaces.

Further reading

- *The Simple Genetic Algorithm* by Michael Vose.
- *Genetic Algorithms: Principles and Perspectives* by Colin Reeves and Jonathan Rowe.
- *Theoretical Aspects of Evolutionary Computing* by Leila Kallel, Bart Naudts and Alex Rogers.
- *Foundations of Genetic Algorithms*.