Lecture 14: Fitness Functions and Landscapes

Fitness Functions

- So far we have examined most of the components of a GA in quite some detail
- One component we haven't paid much attention to so far is the fitness function
- We may be able to learn something about GA behaviour by analysing fitness functions, and the *fitness landscapes* they give rise to

Fitness Functions - Analysis

- An early attempt to analyse fitness functions and hence predict GA performance was via *epistasis variance*
  - *Definition:* Epistasis is the nonlinear interaction of two or more loci on some trait such as fitness
  - Epistasis is the multiple locus equivalent of dominance, which is defined for single loci
- Epistasis will be significant in most GA problems
  - If there is no epistasis, then the problem is basically linear and is best solved by some other technique
- If we can quantify the degree of epistasis in a problem we might gain some advantage
  - Identify the most appropriate technique for the problem
    - GAs are assumed to be best suited to problems with 'intermediate' epistasis
  - Identify suitable encodings and operators
    - E.g. change in encoding might turn a linear fitness function into a non-linear one, and vice-versa

Fitness Functions - Analysis

- Analysis of the entire space of possible solutions is clearly impossible for any realistic optimisation problem we face
- Hence we analyse epistasis variance in terms of a sample population
- Mean fitness of the population P is simply

$$\bar{f} = \sum_{x \in P} \frac{f(x)}{N}$$

- Then the excess fitness value of a chromosome can be calculated as

$$\delta(x) = f(x) - \bar{f}$$

Fitness Functions - Analysis

- If allele a occur at locus i with frequency $N_i(a)$, then the average value of that allele is

$$A_i(a) = \sum_{x:x_i=a} \frac{f(x)}{N_i(a)}$$

- The excess fitness value of having allele a at locus i is thus

$$\delta_i(a) = A_i(a) - \bar{f}$$

- Summing these gives the excess genic value of chromosome x

$$\delta_G(x) = \sum_{i=0}^{\ell-1} \delta_i(a)$$

- So the epistasis value is the difference between the fitness predicted by linear combination of the alleles of the chromosome, and the actual fitness of the chromosome

$$\epsilon(x) = \delta(x) - \delta_G(x)$$

Fitness Functions - Analysis

- Under this epistasis variance analysis it is seen that

$$\sum_{x} \delta(x)^2 = \sum_{i=0}^{\ell-1} \sum_{a} \delta_i(a)^2 + \sum_{x} \epsilon(x)^2$$

- I.e.
  - Total 'variance' = Genic 'variance' + Epistasis 'variance'
- This is directly analogous to the result we get in statistics on partitioning Sum of Squares Error (SS) from Analysis of Variance (ANOVA)
  - Total SS = Main effects SS + Interaction SS
- In fact the theory of epistastis variance was previously developed in the field of statistics called *experimental design*

Fitness Functions - Analysis

- Is epistasis variance useful for predicting GA performance on a fitness function?
- There are two main problems with this approach
  - Sampling
  - Signs of the interaction effects
- *Sampling*
  - As already observed the search space will be too large to fully analyse for epistasis variance
  - So a random subset sample must be used instead
  - However this sample is likely to be far too small in comparison to the search space to guarantee meaningful results from the analysis

Fitness Functions - Analysis

- *Signs of the interaction effects*
  - ANOVA is only concerned with the absolute magnitude of the interaction effects
    - Squaring the errors gives them all the same sign
  - For epistasis variance, however, the sign of the interaction effects is highly relevant
    - If the signs of the interaction effects on a locus *match* the sign of the genic effects at that locus, interaction simply *reinforces* the selective advantage of particular alleles
    - If the signs of the interaction effects on a locus *differ* from the sign of the genic effects at that locus, interaction *interferes with* the selective advantage of particular alleles

Fitness Functions - Walsh Analysis

- We can also analyse a fitness function with a useful tool known as the *Walsh transform*
- The Walsh transform decomposes any boolean function into a superposition of boolean functions known as the *Walsh functions*
  - The Walsh transform is the boolean equivalent of the *Fourier transform*
  - Like the Fourier transform, the Walsh transform is useful in signal processing, image processing, etc.
- The Walsh transform is also useful in many area of GA analysis

Fitness Functions - Walsh Analysis

- The Walsh-transform of a function is

$$f(x) = \sum_{j=0}^{2^\ell - 1} w_j \psi_j(x)$$

- The j-th Walsh function of x is defined as

$$\psi_j(x) = \prod_{i=1}^{\ell} (1 - 2x_i)^{j_i}$$

  where j_i is the i-th bit of the binary vector representing the integer j, and similarly for x_i
- The j-th Walsh coefficient of $f(x)$ is defined as

$$w_j = \frac{1}{2^\ell} \sum_{x=0}^{2^\ell - 1} f(x) \psi_j(x).$$

Fitness Functions - Walsh Analysis

- Walsh-decomposition can only be done on small enough function, but is useful for general theory
- The Walsh coefficients crop up in many places
  - Epistasis variance
  - Schema theory
  - ...

Walsh Analysis - Epistasis Variance

- For all binary chromosomes we can write the decomposition of effects on the fitness of string $(0, 0, 0)$ as

$$f_{000} = \mu + \alpha_0 + \beta_0 + (\alpha\beta)_{00} + \gamma_0 + (\alpha\gamma)_{00} + (\beta\gamma)_{00} + (\alpha\beta\gamma)_{000}$$

- Then the fitness effects above are given by the Walsh coefficients arising from the Walsh-transform of the fitness function

$$\begin{aligned} \mu &= w_0 \\ \alpha_0 &= w_1 \\ \beta_0 &= w_2 \\ (\alpha\beta)_{00} &= w_3 \\ \gamma_0 &= w_4 \\ (\alpha\gamma)_{00} &= w_5 \\ (\beta\gamma)_{00} &= w_6 \\ (\alpha\beta\gamma)_{000} &= w_7 \end{aligned}$$

Walsh Analysis - Epistasis Variance

- The relationship between the Walsh coefficients and the variation terms can be understood as follows
  - The number of 1s in the binary version of the Walsh function's subscript gives the *number* of interaction effects represented by that coefficient
  - The position of those 1s represents *which* interaction effects are represented by that coefficient, by indexing from least significant bit to most significant bit the gene-level effect variables we have from the previous equation

$$(0)\alpha, (1)\beta, (2)\gamma, \ldots$$

    - E.g. $w_0 = w_{000} = \mu$ (no interaction effects)
    - E.g. $w_3 = w_{011} = (\alpha\beta)_{00}$

Walsh Analysis - Schema Theorem

- Recall that schemata can be thought of as periodic binary functions
- Walsh functions are also periodic binary functions
- Hence we can also get schema fitness averages, by choosing the appropriate Walsh coefficients to combine
  - E.g. population mean fitness$= w_0$
  - $f(**0) = w_0 + w_1$
  - $f(**1) = w_0 - w_1$
  - Etc...
- This can be used to construct *deceptive* fitness functions, by imposing appropriate conditions on the Walsh coefficients

Deception

- Deception is a historically important concept in the development of GA theory, and is based on the schema theory
- The idea is that a fitness function will be hard for a GA if the schema fitnesses lead the GA *away* from the global optimum
  - Such a function is described as deceptive
- Goldberg proposed the following simple deceptive fitness function

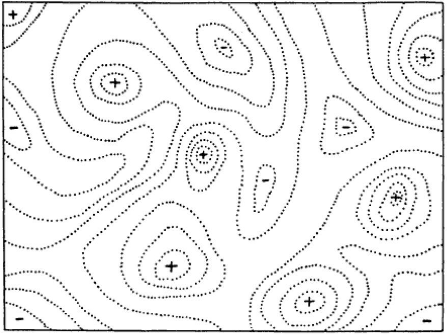
String	Fitness
000	7
001	5
010	5
011	0
100	3
101	0
110	0
111	8

- Fitness is increased by removing ones from the string, but the global optimum contains only ones

Fitness Landscapes

Fitness Landscapes - History

- Sewall Wright's idea of a *fitness landscape* for evolution through natural selection (below, from Wright (1932)) is also appealing for EC



- Populations under selection seek to occupy *fitness peaks* separated by *fitness valleys*

Fitness Landscapes - Definition

- The analysis of fitness landscapes, and the behaviour of Evolutionary Algorithms on them, is an interesting area
- To perform such analysis, we must first define what a fitness landscape is
  - *Definition:* A fitness landscape for a fitness function f is a triple $\mathcal{L} = (\mathcal{C}, f, d)$, where $d : \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}^{+} \cup \{\infty\}$ is the distance measure, such that for all s, t and u in the search space $\mathcal{C}$

$$d(s,t) \geq 0$$

$$d(s,t) = 0 \Leftrightarrow s = t$$

$$d(s,u) \leq d(s,t) + d(t,u)$$

  - If the measure is also symmetric, then we have a distance *metric* on points in our search space

Fitness Landscapes - Neighbourhoods

- Once we have a distance measure on the search space, we can define the concept of a *neighbourhood*
  - *Definition:* The neighbourhood $N(S)$ of a point s in the search space is the set of points that can be reached from s by a single application of an operator ω. d_ω to be the distance measure under the operator ω where

$$t \in N(s) \Leftrightarrow d_\omega(s,t) = 1$$

    The distance between non-neighbours is the length of the shortest path between them
- The important thing to note is that for discrete optimisation problems neighbourhoods, and fitness landscapes, must be defined in terms of an operator
  - It contrasts with optimisation of functions such as $f : \mathbb{R} \rightarrow \mathbb{R}$ where the real number line gives a well defined neighbourhood structure independent of 'operator'
  - It is important for any analysis of a fitness landscape in terms of optima, etc.

Fitness Landscapes - Isomorphism

- Fitness landscapes may be isomorphic
  - *Definition:* Two fitness landscapes are isomorphic if they are equivalent to each other under an appropriate change of representation and operator
- E.g. consider the Bit-Flip (BF) and Complementary Crossover (CX) operators
  - BF neighbourhood

$$N([00000]) = \{(10000), (01000), (00100), (00010), (00001)\}$$

  - CX neighbourhood (1X with a complementary string)

$$N([00000]) = \{(00001), (00011), (00111), (01111), (11111)\}$$

Fitness Landscapes - Isomorphism

- The fitness landscape with standard binary encoding and the CX operator, and the fitness landscape with Gray-encoding and the BF operator, are isomorphic
- E.g. consider the neighbours of (00000) in both landscapes

	BF-Neighbours	Gray-Integer
00000	00001	1
	00010	3
	00100	7
	01000	15
	10000	31
	CX-Neighbours	binary-Integer
00000	00001	1
	00011	3
	00111	7
	01111	15
	11111	31

- This recalls the isomorphism results from Radcliffe & Surry's NFL proof

Fitness Landscapes - Local Optima

- *Definition:* A point $s \in \mathcal{C}$ on a fitness landscape $\mathcal{L} = (\mathcal{C}, f, d)$ is a *local optimum* if for all $t \in N(s)$, $f(s) \geq f(t)$
  - N.B. one (or more) of the local optima in the landscape will also be the global optimum/optima of the search space
  - The number of local optima in a fitness landscape will be one important factor in performance of a search algorithm
  - Even more important are the relative *basins of attraction* of the different optima
    - *Definition:* The basin of attraction of a local optimum is the set of points in the search space from which that local optimum will be attained under some search strategy
  - N.B. while the fitness landscape depends on the operator used but not on the search strategy, the basins of attraction also depend on the search strategy used

Fitness Landscapes - Basins of Attraction

- Compare the basins of attraction under the BF operator of two different neighbourhood search strategies
  - Steepest ascent

Local optimum	01010	01100	00111	10000
Fitness	4100	3988	3803	3236
Basin size	20	3	4	5

  - First ascent

Local optimum	01010	01100	00111	10000
Fitness	4100	3988	3803	3236
Basin size	14	6	4	8

- While the local optima are unchanged, the basins of attraction under the two different search algorithms are different

Fitness Landscapes - Analysis

- It is also possible to analyse specific problems (objective functions) under specific search neighbourhoods
- *Definition:* Under objective function f, for some neighbourhood $\mathcal{N}$ the *graph Laplacian* is

$$\nabla^2 f = \left(\sum_{i=1}^{|\mathcal{N}|} \delta_i \right) / |\mathcal{N}|.$$

  where δ_i is the difference in objective value between the current search point and its i-th neighbour
- So the graph Laplacian is the mean difference in objective value between a point in the search space and its neighbours
  - This is analagous to the continuous-space Laplacian operator from physics, of interest when modelling wave propagation and other physical phenomena

Fitness Landscapes - Analysis

- Some objective functions and neighbourhoods induce a graph Laplacian for the search space $\mathcal{S}$ of the form

$$\nabla^2 f + \frac{K}{|\mathcal{S}|} f = 0$$

  for constant K

- Consider the Symmetric Travelling Salesman Problem with objective function

$$h = \sum_{i=1}^{|\mathcal{S}|} l_{i,(i+1) \bmod |\mathcal{S}|}$$

  where $l_{i,j}$ is the length of the edge between vertices i and j

  - So we want to minimise h during our search

Fitness Landscapes - Analysis

- *Theorem (Grover1):* The Travelling Salesman Problem with city pair swap neighbourhood and with 'normalised' objective function $\overline{h} = h - \langle h \rangle$ satisfies

$$\nabla^2 \overline{h} = -\frac{4}{|\mathcal{S}|} \overline{h}$$

- *I.e.* the mean difference in neighbours' values is always a constant (negative) multiple of the value of the current solution
  - This is already interesting, because it tells us that uniformly at random swapping two cities in a tour is expected to result in an improved solution when that solution is below average quality...
  - ...and a worse solution when that solution is above average quality
  - *I.e.* random search converges on average quality solutions

Fitness Landscapes - Analysis

- Graph Laplacians like those in the Symmetric TSP example can tell us further interesting things
- *Theorem (Grover2):* for objective functions whose graph Laplacian has the form $\nabla^2 f + Kf/|\mathcal{S}| = 0$, all local minima have $f \leq 0$ and all local maxima have $f \geq 0$
  - So all local minima are below average value and all local maxima are above average value
  - Greedy local search until a local optimum is hit will definitely yield an above average solution

Fitness Landscapes - Analysis

- How long will it take local search to find a local optimum?
- *Theorem (Grover3):* a greedy local search algorithm starting from an arbitrarily poor configuration will reach a local optimum in at most $O(|\mathcal{S}|L)$ iterations, where the best solution is at most 2^L better than the average over the search space
  - *N.B.* how L relates to the problem is not addressed here... if it is a constant this is good news!
- Symmetric TSP with pair-swap neighbourhood is not unique in satisfying this requirement on the Graph Laplacian
- It can also hold for other well known NP-complete problems
  - Min-cut graph partitioning
  - Graph colouring
  - Minimum weight partition
  - ...

Fitness Landscapes - Reference

- Grover, L. K. (1992) Local search and the local structure of NP-complete problems. Operations Research Letters 12, 235-243