

MESA: Monotonic Evolutionary Simulation Algorithm

By William A. Dembski

MESA models evolutionary searches that employ monotonic smooth fitness gradients. It presupposes a fitness landscape that converges gradually to a single optimum (peak or valley) and asks how quickly evolution can locate the optimum when fitness is randomly perturbed and/or when variables are coupled. The problem of local optima far removed from the global optimum and thereby undermining an evolutionary search is thus set aside. Instead, the focus is on how fitness perturbation and variable coupling impede evolutionary searches.

Usually when evolutionary algorithms are run to solve engineering problems, the fitness function is well-defined and precisely evaluated for each candidate solution. In nature, however, fitness cannot be scored precisely. Darwin exaggerated when he wrote: "Natural selection is daily and hourly scrutinising, throughout the world, every variation, even the slightest; rejecting that which is bad, preserving and adding up all that is good; silently and insensibly working, whenever and wherever opportunity offers, at the improvement of each organic being in relation to its organic and inorganic conditions of life." Nature selects what is "good enough" and typically cannot discriminate fitness gradations that are too fine. What's more, contingencies can wipe out otherwise optimal organisms. To model this, MESA allows the user to randomly perturb the fitness values assigned to organisms.

Another obstacle to evolutionary searches is the coupling of variables. Often fitness does not increase gradually as the search traverses a continuous path toward an optimum in the search space. Rather, variables may be coupled and increased fitness may occur only when all the coupled variables have the right settings (otherwise, there is no advantage). The coupling of variables in this way models Michael Behe's notion of irreducible complexity. The greater the degree of coupling, the slower evolutionary computation completes its search.

The details of MESA are as follows. Call the search space within which the evolutionary search takes place Ω . Ω consists of all bit strings of length N (enter N in "String Length"). Call these bit strings the "organisms." MESA evolves organisms within a population of size M (enter M in "Population Size"). The initial population consists of M random organisms.

Within Ω , the optimal organism is always the bit string consisting entirely of 0s and the underlying fitness function always measures fitness in terms of proximity to this optimal bit string. Toggle "Target" to make higher fitness correspond to finding a minimum (i.e., "Target at Min Valley" -- fitness counts number of remaining 1s in string) or finding a maximum (i.e., "Target at Max Peak" -- fitness counts number of 0s in string). Let f be the first of these fitness functions (f counts number of 1s) and g the second. Then $g = N - f$. Since f and g are essentially mirror images of each other, for the remainder of this overview, we focus on f .

At each generation, the population is divided in half, with only those organisms in the top 50th percentile used for reproduction. If elitism is off, each of these organisms generate two new organisms by mutations. If elitism is on, all the organisms in the top 50th percentile go into the next generation, but each of these also generate a new organism by mutations. Mutations occur pointwise in accord with the mutation rate r . If crossover is turned off, then each of these procedures yields the next generation.

If crossover is turned on, then there is yet one more step before the next generation: organisms (after being mutated) get paired randomly and then crossed over according to the crossover rate. The crossover rate gives the probability that there is a crossover at any bit location on the paired strings (e.g., for strings of length 100 and crossover points 20 and 40 randomly chosen according to the crossover rate, substrings between locations 20 and 40 of the bit strings get swapped).

Mutations are implemented by two randomization procedures. Under "Mutation Distribution" toggle "Bit by Bit" to go through each string bit by bit, randomly mutating a bit with probability given by the mutation rate r . Alternatively, toggle "Poisson Distribution" to randomly choose the number of bits to be changed according to a Poisson distribution with parameter λ set to N (length of bit string) times r (the mutation rate). The Poisson distribution approach in most instances approximates the bit by bit approach (because when N is large and r is small, the Poisson distribution with parameter $\lambda = Nr$ approximates the binomial distribution with parameters (N,r)).

Fitness perturbation can take three forms. Toggle "Fitness Perturbation" to "Uniform Distribution" and set "Uniform Fitness Perturbation Range" to some nonnegative natural number k . Then for each fitness value of f computed, the outputted fitness value f^* is $f + j$ where j is a uniformly random variate from the set $\{-k, -k+1, \dots, -1, 0, 1, \dots, k-1, k\}$. If toggled to "Binomial Distribution" and "Binomial Fitness Perturbation" is set to (n,p) , then the outputted fitness value f^* is $f + j - np$ where j is a binomial random variate with parameters n and p . If f^* is noninteger, then it is adjusted to the nearest integer. Finally, toggle "Manual Distribution" and enter a distribution by hand under "Manual Fitness Perturbation Distribution" to enter a preferred fitness perturbation probability distribution. Since the fitness function f ranges between 0 and N (the bit string length), if fitness perturbation takes the outputted fitness value f^* outside this range, f^* collapses to either 0 or N (0 if f^* dips below 0, N if f^* exceeds N).

Couplings are described in terms of degree of coupling (i.e., number of bits coupled) and number of couplings of that degree. Couplings are always assigned to the beginning of strings and ordered in the same way they are inputted on the screen. Consider the following input:

10,2
22,1
5,3

In this instance, start with 2 blocks of 10-bit couplings, which are immediately followed by 1 block of a 22-bit coupling, which is immediately followed by 3 blocks of 5-bit couplings. Coupled bits provide no advantage in fitness until all the bits in the coupled

block correspond to the target sequence (the zero bit string). Specifically, this means that unless the bits in the coupled block are all zero, the fitness of the string is evaluated as though all the bits in the block were 1s. The effect of coupling is to flatten the fitness function f (or g) on coupled blocks where the bits diverge from the zero bit string.

Hit "Start" to start the simulation. Hit "Stop" to stop it. Hit "Pause" to pause a simulation and, if desired, readjust parameters. Hit "Unpause" to resume simulation.

The reference to monotonicity in MESA (Monotonic Evolutionary Simulation Algorithm) refers to the underlying fitness function being monotonic -- that is, the underlying fitness function is like a mountain (valley) where everything converges up (down) to one lone peak (valley) that is the global maximum (minimum) with no competing local optima. This enables one to isolate the effects of coupling and fitness perturbation in evolutionary computation.

Note that there is no loss of generality locating the optimum at the zero bit string or for that matter having all couplings placed in blocks at the beginning of the bit strings. Ω is an Abelian group for which the group operation of component by component modular arithmetic constitutes an automorphism (i.e., bijective structure preserving map of Ω to itself). One can therefore take an arbitrary bit string u and map Ω to itself under the operation $x \rightarrow x + u$. If u corresponds to the ASCII encoding of METHINKS IT IS LIKE A WEASEL, then the convergence of MESA to the zero bit string is isomorphic to the convergence of Richard Dawkins's Weasel program to METHINKS IT IS LIKE A WEASEL. Indeed, since bit strings can represent just about anything (graphic images, text, protein sequence data, you name it), convergence of MESA to the zero bit string can be made to mirror the convergence of evolutionary computation with respect to monotonic smooth fitness gradients. As for placing coupled blocks at the beginning of the bit strings, this too involves no loss of generality since one can map Ω to itself bijectively by permuting all bit location indices. In this way, arbitrary couplings can be mapped onto blocked couplings at the beginning of bit strings.

MESA is the joint work of John Bracht, William Dembski, and Micah Sparacio. Briefly, WD had the idea of modifying monotonic smooth fitness functions via fitness perturbation and coupling. JB provided key biological insights and connections. MS brought the project to reality by doing the actual programming of MESA. Each of us was intimately involved in all aspects of the project, which is the outcome of weeks of intense discussion and work by the three of us. We also wish to thank Iain Strachan for key initial insights (the idea of variable coupling and its relation to irreducible complexity was his) as well as invaluable advice along the way.

MESA is a work in progress. We will continue to add functionalities to the program. We solicit your feedback for strengthening the program as well as for interpreting its results. The Brainstorms discussion forum is the place for that feedback. Eventually, we may make MESA an open source code project.