Evolution Strategies

Evolution Strategies (ESs) are in many ways very similar to Genetic Algorithms (GAs). As their name implies, ESs also simulate natural evolution. The differences between GAs and ESs arise primarily because the original applications for which the algorithms were developed are different. While GAs were designed to solve discrete or integer optimization problems, ESs were first applied to continuous parameter optimization problems associated with laboratory experiments.

ESs were introduced in the 1960s by Rechenberg [1973] working in Berlin and further developed by Schwefel [1975a]. The first numerical simulations were performed by Hartmann [1974], and the first attempts at using ESs to solve discrete optimization problems were made by Schwefel [1975b].

Like GAs, ESs differ from traditional optimization algorithms in some important respects:

- They search from one population of solutions to another, rather than from individual to individual.
- They use only objective function information, not derivatives.
- They use probabilistic, not deterministic, transition rules.

The basic structure of an ES is shown in Figure 1 and is very similar to that of a basic GA. One minor change from the standard optimization routine flow diagram is the use of the word ‘population’ rather than ‘solution’. A more major difference is that the usual operation of generating a new solution has been replaced by three separate activities — selection, recombination and mutation. It is in the implementation of these operations that the differences between ESs and GAs lie.

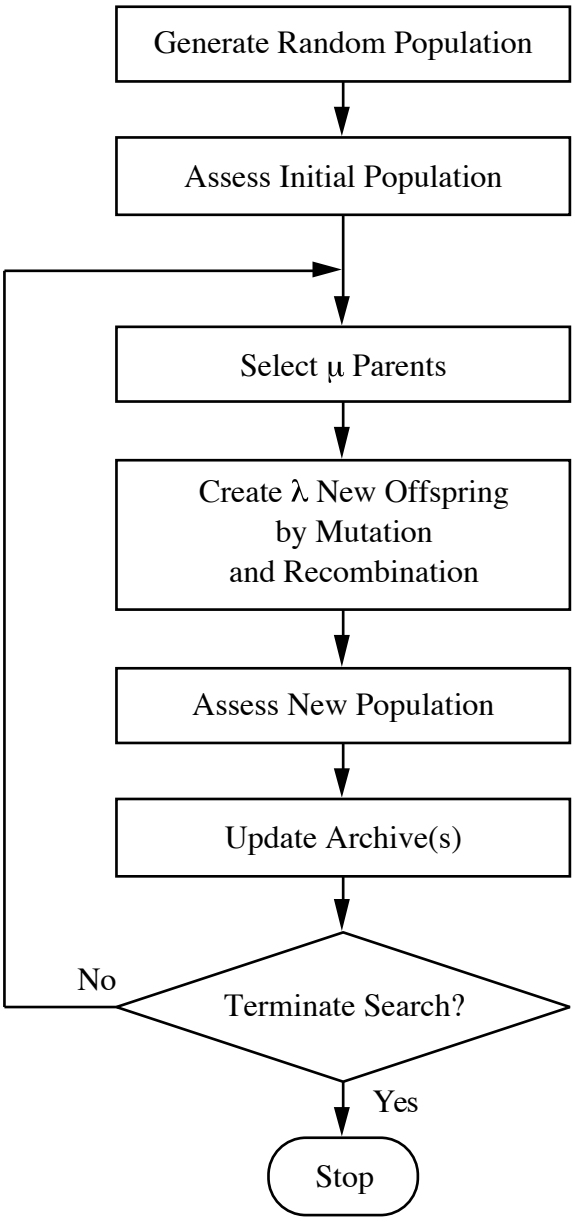
ESs, GAs and associated algorithms are now known collectively as *evolutionary algorithms* and their use as *evolutionary computation*. This field is thoroughly reviewed in much more detail than is possible here in two excellent publications by Bäck [1995] and Schwefel [1995].

Solution Representation

ESs are still primarily used to solve optimization problems with continuous control variables and for applications of this sort the natural representation of the control variables as an n-dimensional real-valued vector $\mathbf{x}$ is entirely appropriate. In addition, the representation of a solution may include (depending on the specific ES implementation being employed) up to n different variances $c_{ii} = \sigma_i^2$ and up to $n(n-1)/2$ covariances c_{ij} of the generalized n-dimensional normal distribution with zero means and a probability density function:

$$p(\mathbf{z}) = \sqrt{\frac{|\boldsymbol{A}|}{(2\pi)^n}} \exp\left(-\frac{1}{2}\mathbf{z}^T\boldsymbol{A}\mathbf{z}\right), \qquad (1)$$

Figure 1: The Basic Structure of an Evolution Strategy.



where $\boldsymbol{A}^{-1} = \{ c_{ij} \}$ is the covariance matrix and $\mathbf{z}$ the vector of random variables. To ensure that the matrix $\boldsymbol{A}^{-1}$ is positive-definite, ES algorithm implementations usually work in terms of the equivalent rotation angles rather than the covariances themselves:

$$\alpha_{ij} = \frac{1}{2} \tan^{-1} \left(\frac{2 c_{ij}}{\sigma_i^2 - \sigma_j^2} \right). \tag{2}$$

These variances, covariances and rotation angles are known as *strategy parameters*. They control the probability density function $p(\mathbf{z})$, which is used in performing mutation.

Mutation

In GA implementations mutation is usually a background operator, with crossover (recombination) being the primary search mechanism. In ES implementations mutation takes a much more central role. In its most general form the ES mutation operator works as follows:

- First, if they are used, the standard deviations and rotation angles (strategy parameters) associated with the individual solution are mutated:

$$\sigma_i' = \sigma_i \exp(\tau' \times \aleph_0 + \tau \times \aleph_i) ; \tag{3}$$

$$\alpha_{ij}' = \alpha_{ij} + \beta \times \aleph_{ij} , \tag{4}$$

  where the $\aleph$s are (different) random numbers sampled from a normally distributed one-dimensional random variable with zero mean and unity standard deviation. $\aleph_0$ is chosen once only; $\aleph_i$ once for each i; $\aleph_{ij}$ once for each ij. τ, τ' and β are algorithm control parameters for which Schwefel [1995] recommends the following values:

$$\tau = \frac{1}{\sqrt{2\sqrt{n}}} \qquad \tau' = \frac{1}{\sqrt{2n}} \qquad \beta = 0.0873 , \tag{5}$$

  n being the number of control variables.

- Then the vector of control variables is mutated:

$$\mathbf{x}' = \mathbf{x} + \mathbf{n} , \tag{6}$$

  where $\mathbf{n}$ is a vector of random numbers sampled from the n-dimensional normal distribution with zero means and the probability density function in equation (1).

This scheme allows the control variable mutations to be correlated according to the values of the rotation angles α_{ij}, while the standard deviations σ_i control the size of individual mutations. The parameter τ' controls overall changes in mutability and parameter τ allows for changes in the individual 'step sizes' σ_i.

This mutation mechanism enables the ES algorithm to evolve its own strategy parameters appropriate to the problem being tackled as the search progresses, a process termed *self-adaptation* by Schwefel [1987].

Recombination

A variety of recombination operators have been used in ESs. Some, like the GA crossover operator, combine components from two randomly selected parents, while others allow components to be taken from any of the solutions in the parent population. Recombination is applied not only to the control variables but also to the strategy parameters. Indeed, in some ES implementations different recombination operators are applied to different components of the solution representation. The most commonly used recombination operators are described in the following sections.

Discrete Recombination

In discrete recombination the offspring solution inherits its components (control variables, strategy parameters) from one or other of two randomly selected parents, the parent to contribute each component being chosen by a series of 'coin tosses'. For instance, a possible offspring of the two parents:

$$(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$(y_1, y_2, y_3, y_4, y_5, y_6)$$

might be:

$$(y_1, x_2, y_3, x_4, y_5, y_6)$$

Global Discrete Recombination

In global discrete recombination the offspring solution inherits its components from any member of the population, the parent to contribute each component being chosen by 'balanced roulette wheel selection'. Thus, if there are μ members of the population, each has a $1/\mu$ chance of being selected to contribute each component of each offspring solution. For instance, a possible offspring from the four-member population:

$$(u_1, u_2, u_3, u_4, u_5, u_6)$$

$$(v_1, v_2, v_3, v_4, v_5, v_6)$$

$$(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$(y_1, y_2, y_3, y_4, y_5, y_6)$$

might be:

$$(v_1, u_2, x_3, v_4, y_5, v_6)$$

Intermediate Recombination

In intermediate recombination the offspring solution (O) inherits components which are a weighted average of the components from two randomly selected parents (1 and 2):

$$u_{Ok} = \omega u_{1k} + (1 - \omega) u_{2k} , \qquad (7)$$

where component u_k could be a control variable component x_i, a standard deviation σ_i or a rotation angle α_{ij}, and weighting ω traditionally has a value of 0.5, although some ES implementations allow ω to vary.

Global Intermediate Recombination

Global intermediate recombination is similar to intermediate recombination in that the offspring solution inherits components which are a weighted average of the components from two randomly selected parents, but now new parents for each component of the offspring are

chosen from the population by ‘balanced roulette wheel selection’.

It has been found that, in practice, ESs tend to perform best if discrete recombination is performed on the control variables and intermediate recombination on the strategy parameters.

Selection

In ESs selection is completely deterministic. The best μ individuals from the population of λ offspring or from the combination of the μ previous parents and their λ offspring are chosen as parents for the next generation. The former scheme is known as (μ, λ)-selection, the latter as $(\mu + \lambda)$-selection.

Though $(\mu + \lambda)$-selection is *elitist* (the best solution is guaranteed to survive from generation to generation), (μ, λ)-selection is, in general, preferred, because it is better able to adapt to changing environments (a common feature of the parameter optimization problems for which ESs were originally developed). Schwefel [1987] recommends a ratio of μ:λ of 1:7.

Population Assessment

An ES does not use derivative information, it just needs to be supplied with an objective function value for each member of each population. Thus, the evaluation of the problem functions is essentially a ‘black box’ operation as far as the ES is concerned. Obviously, in the interests of overall computational efficiency, the problem function evaluations should be performed efficiently.

The guidelines for constraint handling in an ES are basically identical to those outlined for GAs. As long as there are no equality constraints, the problem is not very heavily constrained, and the feasible space is not disjoint, then infeasible solutions can simply be ‘rejected’. In an ES this means ensuring that those particular solutions are not selected as parents in the next generation.

If these conditions on the constraints are not met, then a penalty function method should be used. A suitable form for an ES is:

$$f_A(\mathbf{x}) = f(\mathbf{x}) + M^k \mathbf{w}^T \mathbf{c}_V(\mathbf{x}), \tag{8}$$

where $\mathbf{w}$ is a vector of nonnegative weighting coefficients, the vector $\mathbf{c}_V$ quantifies the magnitudes of any constraint violations, M is the number of the current generation and k is a suitable exponent. The dependence of the penalty on generation number biases the search increasingly heavily towards feasible space as it progresses.

Convergence Criteria

Two standard convergence tests are used to terminate ES searches. One is that the *absolute difference* in the objective function values of the best and worst members of the post-selection

population is less than a user-specified limit, i.e.

$$\left| f_w - f_b \right| \leq \varepsilon_c \,. \tag{9}$$

The other is that the *relative difference* in the objective function values of the best and worst members of the post-selection population is less than a user-specified limit, i.e.

$$\left| f_w - f_b \right| \leq \frac{\varepsilon_d}{\mu} \left| \sum_{i=1}^{\mu} f_i \right| . \tag{10}$$

The latter should only be used if f is guaranteed to be positive.

Thus, absolutely or relatively, the objective function values of the parents must lie close together before the algorithm is deemed to have converged.

Computational Considerations

Like GAs, ESs are particularly well-suited to implementation on parallel computers or multi-processor networks. Evaluation of the objective function and constraints can be done simultaneously for a whole population, as can the production of the new population by mutation and recombination. Thus, on a highly parallel machine/network, an ES can be expected to run nearly N times as fast for many problems, where N is the smaller of the population size and the number of processors.

Example: Rosenbrock's Function $f(\mathbf{x}) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$

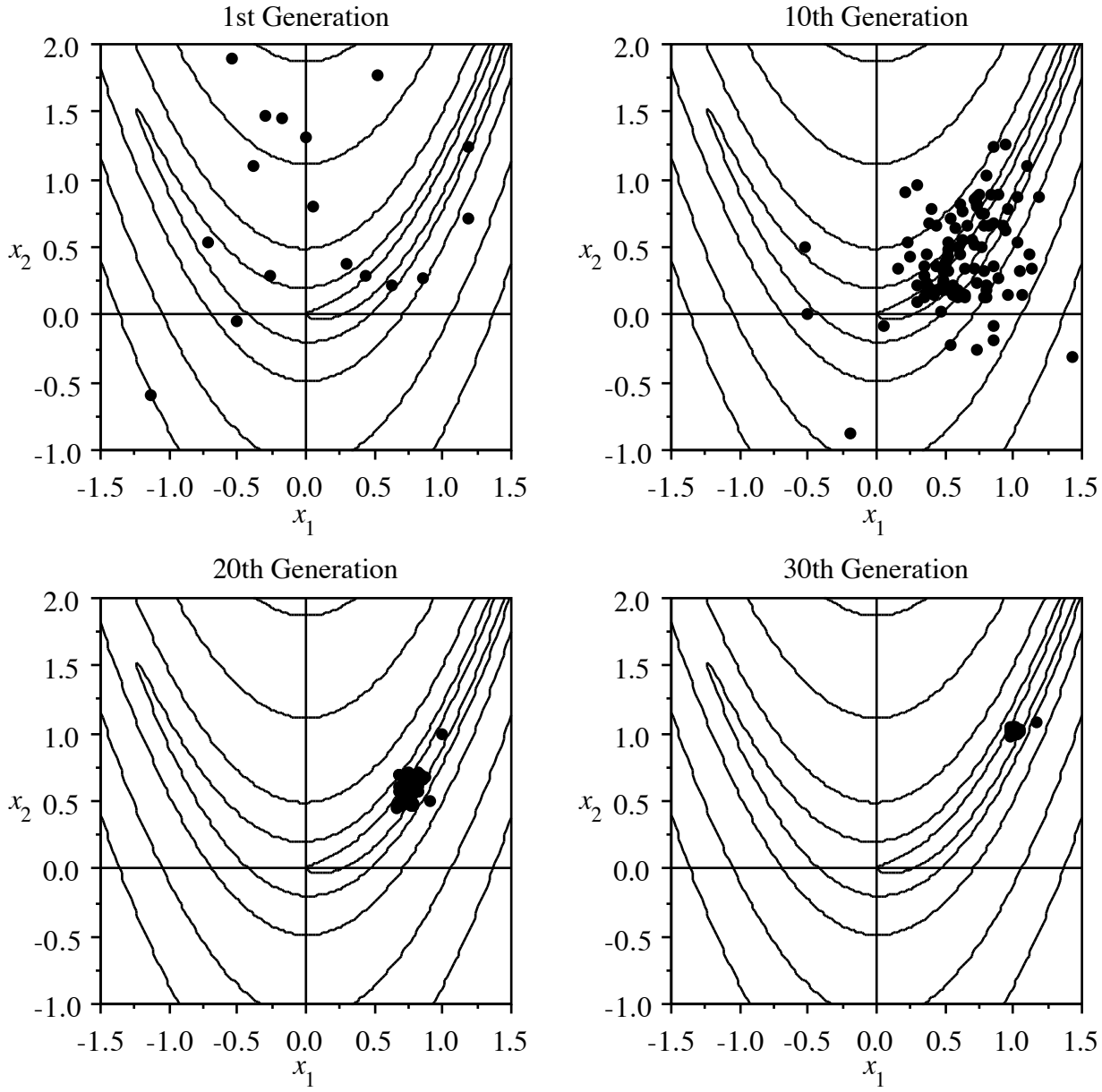
Figure 2 shows the progress of an ES on the two-dimensional Rosenbrock function. Each member of the 1st, 10th, 20th and 30th generations is shown (by a • symbol), although, in fact, most of the members of the 1st generation lie outside the bounds of the figure. The convergence of the population to the neighbourhood of the optimum at $(1, 1)$ is readily apparent. Notice how the 20th generation appears to be converging on $(0.75, 0.7)$, but in the next 10 generations the search successfully progresses along the shallow valley to the true optimum.

For this run a (20, 100)-ES was used. The two control variables were subject to discrete recombination (of pairs of parents), while the strategy parameters were subject to global intermediate recombination.

Figure 3 shows the progress in reducing the objective function for the same search. Both the objective function of the best individual within each population and the population average objective are shown (note that the scales are different). These are the two standard measures of progress in an ES run. The difference between these two measures is indicative of the degree of convergence in the population.

The code used in this example was Schwefel's KORR program. This and other ES programs is available (on diskette) with his book "Evolution and Optimum Searching" in both FORTRAN

Figure 2: Minimization of Rosenbrock's Function by an ES — Population Distributions.



and C. The book also contains fully commented listings of the FORTRAN source codes.

References

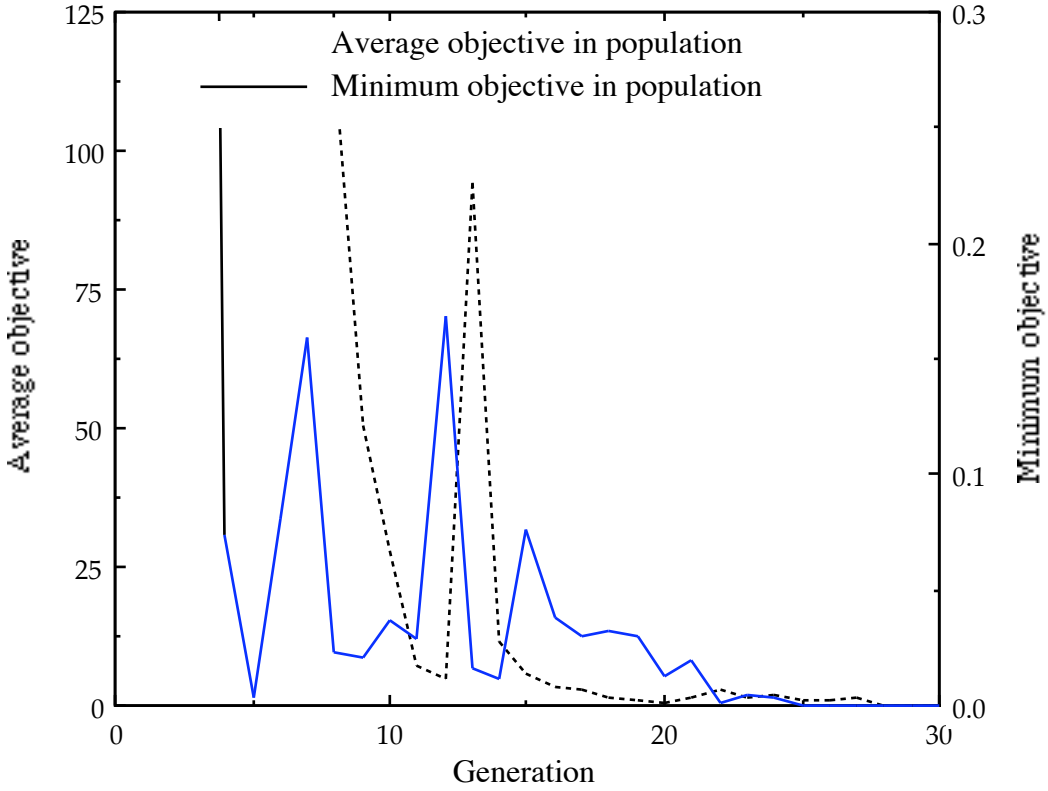
Bäck, T., (1995) *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, Oxford

Hartmann, D., (1974) *Optimierung Balkenartiger Zylindeerschalen aus Stahlbeton mit Elastischem und Plastischem Werkstoffverhalten*, PhD Thesis, University of Dortmund.

Rechenberg, I., (1973) *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart.

Schwefel, H-P., (1975a) *Evolutionsstrategie und Numerische Optimierung*, Dissertation, Technical University of Berlin.

Figure 3: Minimization of Rosenbrock's Function by an ES — Objective Reduction.



Schwefel, H-P., (1975b) "Binäre Optimierung durch Somatische Mutation", Technical Report, Technical University of Berlin and Medical University of Hannover.

Schwefel, H-P., (1987) "Collective Phenomena in Evolutionary Systems", 1025-1033 in Preprints of the 31st Annual Meeting of the International Society for General System Research, Budapest.

Schwefel, H-P., (1995) *Evolution and Optimum Searching*, Wiley Interscience, John Wiley & Sons, New York.