C2.7 Generation gap methods

Jayshree Sarma and Kenneth De Jong

George Mason University, Virginia, USA

Abstract

In this section, we will examine how generation gap methods are related to the concept of overlapping and non-overlapping populations in evolutionary algorithms (EAs). We will then review why generation gap methods were designed. We will compare and contrast the generational and steady state systems. Finally we will describe the elitist strategies used in evolutionary algorithms.

C2.7.1 Introduction

The concept of generation gap is linked to the notion of *non-overlapping* and *overlapping* populations. In a non-overlapping model parents and offspring never compete with one another, i.e., the entire parent population is always replaced by the offspring population, while in an overlapping system, parents and offspring compete for survival. The term *generation gap* refers to the amount of overlap between parents and offspring. The notion of a generation gap is closely related to selection algorithms and population management issues.

A selection algorithm in an evolutionary algorithm (EA) involves two elements: (1) a selection pool and (2) a selection distribution over that pool. A selection pool is required for reproduction selection as well as for deletion selection. The key issue in both these cases is "what does the pool contain when parents are selected and when survivors are selected?"

In the selection for reproduction phase, parents are selected to produce offspring and the selection pool consists of the current population. How the parents are selected for reproduction depends on the individual EA paradigm.

In the selection for deletion phase, a decision has to be made as to which individuals to select for deletion to make room for the new offspring. In non-overlapping systems the entire selection pool consisting of the current population is selected for deletion, i.e., the parent population (μ) is always replaced by the offspring population (λ). In overlapping models, the selection pool for deletion consists of both parents and their offspring. Selection for deletion is performed on this combined set and the actual selection procedure varies in each of the EA paradigms.

Historically, both evolutionary programming and evolution strategies had overlapping populations while the canonical genetic algorithms used non-overlapping populations.

C2.7.2 Historical perspective

In evolutionary programming (Fogel *et al* 1966), each individual produces one offspring and the best half from the parent and offspring populations are selected to form the new population. This is an overlapping system as the parents and their offspring constantly compete with each other for survival.

In evolution strategies (Schwefel 1981), the $(\mu + \lambda)$ and the (μ, λ) models correspond to the overlapping and non-overlapping populations respectively. In the $(\mu + \lambda)$ system parents and offspring compete for survival and the best μ are selected. In the (μ, λ) model the number of offspring produced is generally far greater than the parents. The offspring are then ranked according to fitness and the best μ are selected to replace the parent population.

Genetic algorithms are based on the two reproductive plans introduced and analyzed by Holland (1975). In the first plan, R_1, at each time step a single individual was selected probabilistically using payoff-proportional selection to produce a single offspring. To make room for this new offspring, one individual from the current population was selected for deletion using a uniform random distribution.

In the second plan, R_d, at each time step all individuals were deterministically selected to produce their expected number of offspring. The selected parents were kept in a temporary storage location. When the process of recombination was completed, the offspring produced replaced the entire current population. Thus in R_d, individuals were guaranteed to produce their expected number of offspring (within probabilistic roundoff).

At that time, from a theoretical point of view, the two plans were viewed as generally equivalent. However, because of practical considerations relating to the overhead of recalculating selection probabilities and severe genetic drift (allele loss) in small populations, most early researchers favored the R_d approach.

The earliest attempt at evaluating the properties of R_1 and R_d plans was a set of empirical studies (De Jong 1975) in which a parameter G, called the *generation gap*, was defined to introduce the notion of overlapping generations. Generation gap parameter controls the fraction of the population to be replaced in each generation. Thus, $G = 1$ (replacing the entire population) corresponded to R_d and $G = 1/\mu$ (replacing a single individual) represented R_1.

These early studies (De Jong 1975) suggested that any advantages that overlapping populations might have were offset by the negative effects of genetic drift (allele loss). The genetic drift was caused by the high variance in expected lifetimes and expected number of offspring, mainly because at that time, generally, modest population sizes were used ($\mu \leq 100$). These negative effects were shown to increase in severity as G was reduced. These studies also suggested the advantages of an *implicit* generation overlap. That is, using the optimal crossover rate of 0.6 and optimal mutation rate of 0.001 (identified empirically for the test suite used) meant that approximately 40% of the offspring were clones of their parents, even when $G = 1$. A later empirical study by Grefenstette (1986) confirmed the earlier results that a larger generation gap value improved performance.

However, early experience with classifier systems (e.g. Holland 1978) yielded quite the opposite behavior. In classifier systems only a subset of the population is replaced each time step. Replacing a small number of classifiers was generally more beneficial than replacing a large number or possibly all of them. Here the poor performance observed as the generation gap value increased was attributed to the fact that the population as a whole represented a single solution and thus could not tolerate large changes in its content.

In recent years, computing equipment with increased capacity is easily available and this effectively removes the reason for preferring the R_d approach. The desire to solve more complex problems using genetic algorithms have prompted researchers to develop an alternative to the generational system called the "steady state" approach in which typically parents and offspring do coexist (e.g. Syswerda 1989, Whitley and Kauth 1988).

C2.7.3 Steady state and generational evolutionary algorithms

Steady state EAs are systems in which usually only 1 or 2 offspring are produced in each generation. The selection pool for deletion can consist of the parent population only or can be possibly augmented by the offspring produced. The appropriate number of individuals are selected for deletion, based on some distribution, to make room for these new offspring. Generational systems are so named because the entire population is replaced every generation by the offspring population, i.e., the lifetime of each individual in the population is only one generation. This is the same as the *non-overlapping* population systems, while the steady state EA is an *overlapping* population system.

One can conceptually think of a steady state model in evolutionary programming and evolution strategies. For example, from a parent population of μ individuals, a single offspring can be formed by recombination and mutation which can then be inserted into the population. A recent study of the steady state evolutionary programming done by Fogel and Fogel (1995) concluded that the generational model of evolutionary programming may be more appropriate for practical optimization problems. The first example of the steady state evolutionary strategies is the $(\mu + 1)$ approach introduced by Rechenberg (1973) which had a parent population greater than 1 ($\mu > 1$). All the parents were then allowed to participate in the reproduction phase to create one offspring. The $(\mu + 1)$ model was not used as it was not feasible to self-adapt the step sizes (Bäck 1991).

An early example of the steady state model of genetic algorithms is the R_1 model defined by Holland (1975) in which the selection pool for deletion consists only of the parent population and uniform deletion strategy is used. The R_d approach is the generational genetic algorithm. Theoretically, the two systems (overlapping systems using uniform deletion and non-overlapping systems) are considered to be similar in expectation for infinite populations. However, there can be high variance in the expected lifetimes and expected number of offspring when small finite populations are used.

This variance can be highlighted by keeping everything in the two systems constant and changing only one parameter, viz., the number of offspring produced. Figures C2.7.1 and C2.7.2 illustrate the average and variance for the growth curve of the best in two systems, replacing the entire population each generation in one and producing and replacing only a single individual each generation in the second. A population size of 50 was used and the best occupied 10% of the initial population and the curves are averaged over 100 independent runs. Only payoff proportional selection, reproduction, and uniform deletion were used to drive the systems to a state of equilibrium. Notice that in the overlapping system (Figure C2.7.1) the best individuals take over the population only about 80% of the time and the growth curves exhibit much higher variance when compared to the non-overlapping population (Figure C2.7.2).

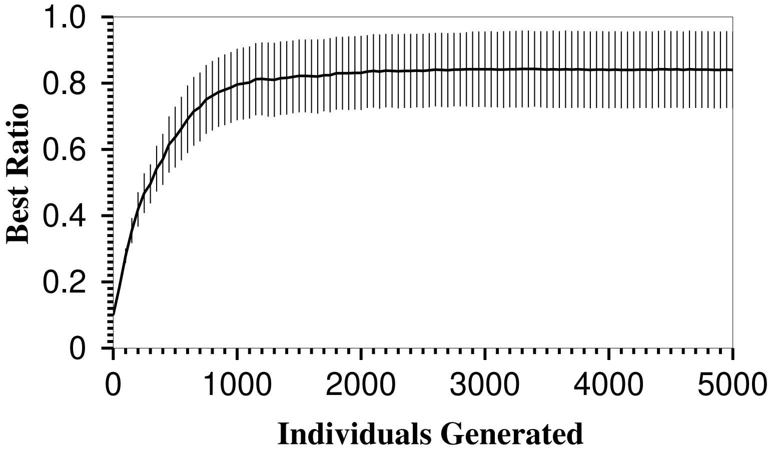

Figure C2.7.1. Mean and variance of the growth curves of the best in an overlapping system (Population Size=50, G=1/50)

This high variance for small generation gap values causes more genetic drift (allele loss). Hence, with smaller population sizes, the higher variance in a steady state system makes it easier for alleles to disappear. Increasing the population size is one way to reduce the the variance (see Figure C2.7.3) and thus offset the allele loss. In summary, the main difference between the generational and steady state systems is higher genetic drift in the latter especially when small population sizes are used with low generation gap values. (See De Jong and Sarma 1993 for more details).

So far we have assumed that there is an uniform distribution on the selection pool used for deletion. But most researchers using a steady state GA generally use a distribution other than the standard uniform distribution. Syswerda (1991) shows how the growth curves can change when different deletion strategies, like deleting the least fit, exponential ranking of the members in the selection pool, and reverse fitness, are used. Peck and Dhawan (1995) demonstrate an improvement in the ideal growth behavior of the steady state system when uniform deletion is changed to a first-in-first-out (FIFO) deletion strategy. An early model of a steady state (overlapping) system is GENITOR (Whitley and Kauth 1988, Whitley 1989) which not only uses ranking selection instead of proportional selection on the selection pool for reproduction but also uses deletion of the worst member as the deletion strategy. The GENITOR approach exhibited significant performance

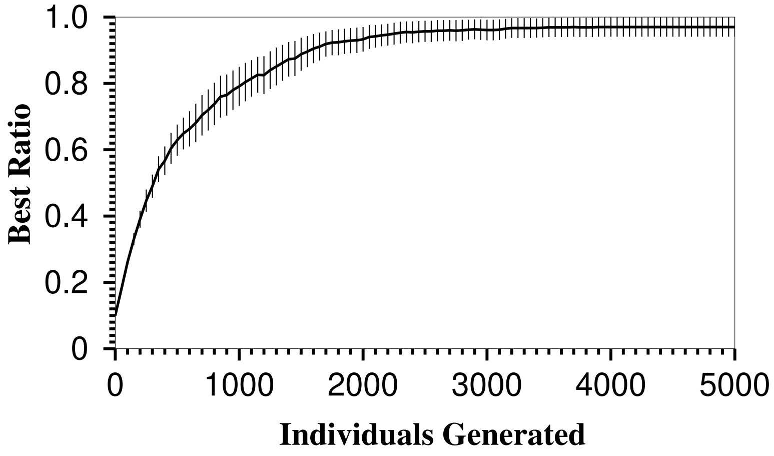

Figure C2.7.2. Mean and variance of the growth curves of the best in a non-overlapping system (Population Size=50, G=1)

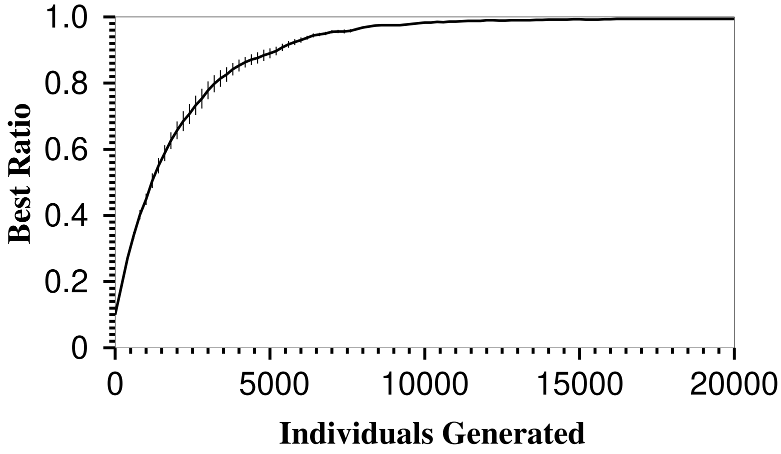

Figure C2.7.3. Mean and variance of the growth curves of the best in an overlapping system (Population Size=200, G=1/200)

improvement over the standard generational approach.

Using a deletion scheme other than an uniform deletion changes the selection pressure. The selection pressure induced by the different selection schemes can vary considerably. Both these changes can alter the exploration/exploitation balance. Two different studies have shown that improved performance in a steady state system, like GENITOR, is due to higher growth rates and changes in the exploration/exploitation balance caused by using different selection and deletion strategies and is not due to the use of an overlapping model (Goldberg and Deb 1991, De Jong and Sarma 1993).

C2.7.4 Elitist strategies

The cycle of birth and death of individuals is very much linked to the management of the population. Individuals that are born have an associated lifetime. The expected lifetime of an individual is typically one generation, but in some EA systems it can be longer. We now explore this issue in more detail.

Elitist strategies link the lifetime of individuals to their fitness. Elitist strategies are techniques to keep good solutions in the population longer than one generation. Though all individuals in a population can expect to have a lifetime of one generation, individuals with higher fitness can have a longer lifetime when elitist strategies are used.

As stated earlier, the selection pool for deletion is comprised of both the parents and the offspring populations in the overlapping system. This combined population is usually ranked according to fitness and then truncated to form the new population. This method ensures that most of the current individuals with higher fitness survive into the next generation thus extending their lifetime. In the $(\mu + \lambda)$ evolution strategies, a very strong elitist policy is in effect as the top μ are always kept. In evolutionary programming, a stochastic tournament is used to select the survivors, and hence the elitist policy is not quite as strong as in the evolution strategies case. In the (μ, λ) evolution strategies there is no elitist strategy to preserve the best parents.

Unlike evolution strategies and evolutionary programming where there is post-selection of survivors based on fitness, in generational genetic algorithms there is only pre-selection of parents for reproduction. Recombination operators are applied on these parents to produce new offspring which are then subject to mutation. Since all parents are replaced each generation by their offspring, there is no guarantee that the individuals with higher fitness will survive into the next generation. An elitist strategy in generational genetic algorithms is a way of ensuring that the lifetime of the very best individual is extended beyond one generation. Thus, unlike evolutionary programming and evolution strategies where more than just the best individual survive, in generational genetic algorithms generally only the best individual survives. Steady state genetic algorithms which use deletion schemes other than uniform random deletion have an implicit elitist policy and so automatically extend the lifetime of the higher fitness individuals in the population.

It should be noted that the elitist strategies were deemed necessary when genetic algorithms are used as function optimizers and the goal is to find a global optimal solution (De Jong 1993). Elitist strategies tend to make the search more exploitative rather than explorative and may not work for problems in which one is required to find multiple optimal solutions.

Reference

Bäck T, Hoffmeister F and Schwefel H.-P A survey of Evolutionary Strategies *Proceedings of the Fourth International Conference on Genetic Algorithms* eds. R K Belew and L B Booker (San Mateo, Calif.: Morgan Kaufmann) pp 2-9

De Jong K A 1975 An analysis of the behavior of a class of genetic adaptive systems *Ph.d Dissertation University of Michigan Ann Arbor*

De Jong K A 1993 Genetic Algorithms are NOT function optimizers *Foundations of Genetic Algorithms - 2* ed. L Darrell Whitley (San Mateo, Calif.: Morgan Kaufmann) pp 5-17

De Jong K A and Sarma J 1993 Generation gaps revisited *Foundations of Genetic Algorithms - 2* ed. L Darrell Whitley (San Mateo, Calif.: Morgan Kaufmann) pp 19-28

Fogel G B and Fogel D B Continuous Evolutionary Programming: Analysis and Experiments *Cybernetics and Systems: An International Journal* **26** 79-90

Fogel L J, Owens A J and Walsh M J *Artificial intelligence through simulated evolution* (New York : John Wiley)

Goldberg D E and Deb K 1991 A comparative analysis of selection schemes used in genetic algorithms *Foundations of Genetic Algorithms - 1* ed. G J E Rawlins (San Mateo, Calif.: Morgan Kaufmann) pp 69-93

Grefenstette J J 1986 Optimization of Control Parameters for genetic algorithms *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-16** 122-8

Holland J H and Reitman J S 1978 Cognitive systems based on adaptive algorithms *Pattern-directed inference systems* eds. D A Waterman & F Hayes-Roth (New York : Academic Press)

Holland J H 1975 *Adaptation in Natural and Artificial Systems* (Ann Arbor, Michigan : University of Michigan Press)

Peck C C and Dhawan A P 1995 Genetic algorithms as global random search methods: An alternative perspective *Evolutionary Computation* **3** 39-80

Rechenberg I *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* (Frommann-Holzboog Verlag : Stuttgart)

Schwefel H.-P 1981 *Numerical optimization of computer models* (Chichester : John Wiley)

Syswerda G 1989 Uniform crossover in genetic algorithms *Proceedings of the Third International Conference on Genetic Algorithms* ed. J David Schaffer (San Mateo, Calif.: Morgan Kaufmann) pp 2-9

Syswerda G 1991 A study of reproduction in generational and steady-state genetic algorithms *Foundations of Genetic Algorithms - 1* ed. G J E Rawlins (San Mateo, Calif.: Morgan Kaufmann) pp 94-101

Whitley D and Kauth J 1988 GENITOR: A different genetic algorithm *Tech. Report CS-88-101 Colorado State University*

Whitley D 1989 The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best *Proceedings of the Third International Conference on Genetic Algorithms* ed. J David Schaffer (San Mateo, Calif.: Morgan Kaufmann) pp 116-21

