Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. James C. Spall

ISBN: 0-471-33052-3

CHAPTER 10

EVOLUTIONARY COMPUTATION II: GENERAL METHODS AND THEORY

This chapter continues the study of evolutionary computation (EC) begun in Chapter 9. The genetic algorithms (GAs) emphasized in Chapter 9 are the most popular methods of EC. Among the issues not addressed in that chapter are the relationship of GAs to other EC methods and the theoretical justification for EC. This chapter addresses these issues.

Section 10.1 provides some background related to EC. Section 10.2 summarizes some of the EC approaches that differ slightly from the GA emphasized in Chapter 9, including evolution strategies and evolutionary programming. Section 10.3 discusses the schema theory that has developed in support of GAs and Section 10.4 uses this and other theory to comment on what makes a problem inherently difficult. Section 10.5 discusses some of the theory available for convergence of EC methods, with an emphasis on theory based on Markov chains. In light of what has been covered in EC and other methods, Section 10.6 revisits the no free lunch theorems that were introduced in Section 1.2. Section 10.7 offers some concluding remarks.

10.1 INTRODUCTION

Chapter 9 introduced evolutionary computation (EC) methods for search and optimization with an emphasis on the genetic algorithm form of such methods. EC is the umbrella term that is used to refer to most population-based search methods. EC algorithms share the principle of being computer-based approximate representations of natural evolution. These algorithms alter the population solutions over a sequence of generations according to statistical analogues of the processes of evolution.

Although there are currently many varieties of EC, there have historically been three general approaches that fall under the umbrella EC heading: GAs, evolution strategies (ES), and evolutionary programming (EP). The three approaches differ in the types of generation-to-generation alterations and the form of computer representation for the population elements. As noted by Fogel (2000, p. 85), however, "an iterative blending has occurred such that all classes of evolutionary algorithms now appear quite similar." Each area within EC has borrowed and modified ideas from the others.

Two broad approaches to theoretical analysis of EC algorithms are the schema-based analysis and Markov chains. Schema-based theory was the first serious attempt at putting EC methods on a rigorous footing. More recently—and with greater success—Markov chains have been used. In fact, the Markov-based approach has revealed significant shortcomings in the schema-based approach and has led to theory that contradicts the implications of the more-informal schema-based methods.

Associated with the theory for EC are the no free lunch theorems introduced in Subsection 1.2.2. The NFL theorems have had their greatest impact on the EC community, partly because the bulk of the NFL work has been published in the EC literature and partly because some of the most extravagant claims of "universality"—which NFL negates—have been associated with EC algorithms.

Let us now begin our discussion of the non-GA methods for EC, the theory for EC, and the NFL theorems. Consistent with the vast majority of EC literature, this chapter considers only noise-free loss (or fitness) measurements, unless noted otherwise.

10.2 OVERVIEW OF EVOLUTION STRATEGY AND EVOLUTIONARY PROGRAMMING WITH COMPARISONS TO GENETIC ALGORITHMS

As discussed in Section 10.1, there are numerous approaches to EC other than GAs, two of which are ES and EP. The aim of this section is to summarize the main distinctions between these two other population-based approaches and GAs. Recall, however, that over time the distinctions have been getting less important as more practical implementations borrow elements from more than one of the EC methods. Nonetheless, it is worth being exposed to the distinctions in order to understand the literature in the field and in order to recognize features that may be useful in practical implementations. More detailed discussion on these distinctions is given in Michalewicz (1996, pp. 159–168, 283–285) and Schwefel (1995, pp. 151–160). An additional relatively popular approach similar to ES is differential evolution, as described in Storn and Price (1997).

ES was originally designed for constrained continuous variable optimization problems, in contrast to the broader aims of GAs with their use in studying adaptive systems. (Of course, in practice, GAs have also been most often used in optimization in both discrete and continuous variable problems.) Like GAs, the ES moves a population of candidate solutions from generation-to-generation with the aim of converging to a global minimum $\boldsymbol{\theta}^*$. Although the original ES in Rechenberg (1965) worked with only a population size $N = 1$, more modern implementations have emphasized the $N > 1$ setting. The two general forms of ES in most widespread use are referred to by the notation $(N+\lambda)$-ES and (N, λ)-ES; these will be summarized in the basic steps below and discussion to follow. ES works directly with $\boldsymbol{\theta}$; there is no coding of $\boldsymbol{\theta}$ as often occurs with GAs.

Core ES Steps for Noise-Free Fitness Evaluations

Step 0 **(Initialization)** Randomly or deterministically generate an initial population of N values of $\boldsymbol{\theta} \in \Theta$ and evaluate L for each of the values.

Step 1 Generate λ offspring from the current population of N candidate $\boldsymbol{\theta}$ values such that all λ values satisfy direct or indirect constraints on $\boldsymbol{\theta}$.

Step 2 For an $(N+\lambda)$-ES, select the N best values from the combined population of N original values plus λ offspring; for an (N, λ)-ES, select the N best values from the population of $\lambda > N$ offspring only.

Step 3 Terminate the algorithm if a stopping criterion is met or the budget of fitness function evaluations is exhausted; else return to step 1.

The central operation in moving from generation to generation in ES is the creation of the λ offspring (child) $\boldsymbol{\theta}$ values in step 1 above. The offspring are generated from parent $\boldsymbol{\theta}$ values according to the formula

$$\boldsymbol{\theta}_{\text{child}} = \boldsymbol{\theta}_{\text{parent}} + N(\mathbf{0}, \boldsymbol{D}_{\text{parent}}), \tag{10.1}$$

where the "$+ N(\mathbf{0}, \boldsymbol{D}_{\text{parent}})$" part refers to the addition of a p-dimensional normal random vector with a diagonal covariance matrix $\boldsymbol{D}_{\text{parent}}$. If $\boldsymbol{\theta}_{\text{child}}$ does not satisfy the constraints in the problem (on $\boldsymbol{\theta}$ and/or on $L(\boldsymbol{\theta})$), then the solution is discarded and another $\boldsymbol{\theta}_{\text{child}}$ is generated in its place. Optionally, the $i = 1, 2, \ldots, p$ diagonal elements of the matrix $\boldsymbol{D}_{\text{parent}}$ can themselves be updated according to

$$D_{\text{child},i} = D_{\text{parent},i} \exp[N(0, \sigma^2)], \tag{10.2}$$

where $\exp[N(0, \sigma^2)]$ refers to the exponentiation of a $N(0, \sigma^2)$ random variable with user-specified variance σ^2. The matrix $\boldsymbol{D}_{\text{parent}}$ in the subsequent generation is replaced by $\boldsymbol{D}_{\text{child}}$ (which becomes $\boldsymbol{D}_{\text{parent}}$ in the next iteration).

The parent $\boldsymbol{\theta}$ in (10.1) can be chosen in any of several ways from the current population of N elements. One of the most popular methods is via an analogue to the crossover operation of a GA. Essentially, two $\boldsymbol{\theta}$ vectors are chosen (with or without replacement) from the current population of size N according to a uniform distribution. Then each of the p components of $\boldsymbol{\theta}_{\text{parent}}$ is created by randomly choosing from the corresponding component of either the first or second $\boldsymbol{\theta}$ vector. For example, when $p = 3$ and the first and second $\boldsymbol{\theta}$ vectors are $[0.1, 10.4, 2.3]^T$ and $[0.7, 46.3, 1.3]^T$, respectively, we might have $\boldsymbol{\theta}_{\text{parent}} = [0.1, 10.4, 1.3]^T$. The same type of interchange operation used to create $\boldsymbol{\theta}_{\text{parent}}$ can be used to create $\boldsymbol{D}_{\text{parent}}$. Relevant constraints must be satisfied before a $\boldsymbol{\theta}_{\text{child}}$ will be considered for survival into the next generation. The process of drawing from the population and creating offspring is continued until a full population for the next generation is created.

Steps 2 and 3 of the ES are largely self-explanatory. Note that in the (N, λ) strategy, the life of each population element is limited to one generation

since the selection of elements for the succeeding generation is from the λ offspring only. This is believed to have advantages in nonstationary systems where $\boldsymbol{\theta}^*$ may be changing in time (Michalewicz, 1996, p. 162). As with other stochastic algorithms, termination occurs using ad hoc stopping principles or when the budget of loss evaluations is expended (see Subsection 9.4.3).

The third cornerstone of EC, evolutionary programming (EP), was described in its early form by L. Fogel et al. (1966). Some more recent incarnations are described in D. Fogel (2000, Sect. 3.3), among other references. EP is strongly motivated by problems in artificial intelligence. The original goal of EP was slightly different from the $\boldsymbol{\theta}$-based search and optimization goals emphasized above.

Namely, the EP was aimed at evolving artificial intelligence by creating *finite-state machines* that are adept at prediction. (Chapter 11 also treats prediction problems via learning and the temporal difference method.) A finite-state machine can be represented as a directed graph (a representation with nodes and edges); this representation can be used to predict the next symbol in a sequence of symbols. For example, if some real system has generated output s_1, s_2, ..., s_n, then the machine can be used to predict s_{n+1}. EP works by evolving a population of finite-state machines, much as the other evolutionary methods evolve some representation of the $\boldsymbol{\theta}$ vector. Each finite-state machine in the population is typically represented in matrix form. In evaluating the fitness of a machine in the population, the predictions from the machine are compared with real outcomes according to some fitness function. As in ES above, EP first creates offspring through a mutation operator and then selects individuals for the next generation. An extensive list of references on EP, including references dealing with our main interest in minimizing $L(\boldsymbol{\theta})$ with respect to a vector $\boldsymbol{\theta}$, is given in Fogel (2000, Sect. 3.3).

Let us now contrast the three EC approaches. The main commonality is that they are population-based methods. GAs have traditionally relied on bit coding, whereas ES and EP have operated with the more natural floating-point representations (although, more recently, there has also been a drift in GA applications toward floating-point implementations). Another difference between GAs and both ES and EP is the ordering of the main operations. In GAs, the selection step precedes the crossover and mutation operations, while with ES and EP the opposite is true. For example, in GAs, candidate parents are selected and then crossover and mutation are applied. In ES, first, a $\boldsymbol{\theta}_{\text{parent}}$ is formed by a crossover operation; then the mutation step is applied, leading to $N+\lambda$ or $\lambda > N$ offspring; finally, selection is applied to bring the population size back to N.

A final contrast is the emphasis on general constrained problems in the ES and EP. These algorithms allow for a direct check on constraint violation and the exclusion of an offspring that violates the constraints. The coefficients of the algorithms may automatically be adjusted if the constraints are violated too frequently (e.g., by lowering $\boldsymbol{D}_{\text{parent}}$ in an ES). In contrast, the GA is largely used with simple hypercube constraints, although it is possible to modify the fitness function to include a penalty function as a way of handling more general

constraints. Other distinctions are discussed by Michalewicz (1996, Sects. 8.2 and 13.1) and Fogel (2000, Chap. 3).

As mentioned above, however, the distinctions among these various EC methods are washing away over time as various desirable elements of the evolutionary methods are combined into hybrid forms (e.g., real-coded GAs with adaptable algorithm parameters). All EC methods involve random variation applied to a population of candidate solutions and some method for selection that tends to favor the best solutions.

10.3 SCHEMA THEORY

This and the next two sections are devoted to some of the theory behind EC, focusing on GAs in particular. The schema theory of this section is both illuminating and controversial, illuminating because it provides some of the intuitive basis for GAs and controversial because it has led to some "leaps of faith" that later proved misguided. Although EC ideas have been around since at least the 1950s, schema theory was later developed as the first serious attempt to put EC—and GAs in particular—on a rigorous footing.

Holland (1975) pioneered this concept as applied to bit-based GAs, and a large fraction of his book is devoted to the discussion of schemas. A more recent description of the theory is given in Michalewicz (1996, Chap. 3). Essentially, a schema is a template for the chromosome that constrains certain elements to take on fixed bit values while allowing the other elements to be free. An example would be the eight-bit chromosome with template [* 1 0 * * * * 1], where the * symbol represents a *don't care* (or free) element. The chromosomes [0 1 0 1 1 0 1 1] and [1 1 0 0 1 1 0 1] are two specific *instances* of this schema. On the other hand, [1 1 1 0 1 1 0 1] is not an instance. The *order* of a schema is the number of defined elements (0 or 1). Schemas are sometimes referred to as the *building blocks* of GAs because it is believed by some that a GA constructs its solution by the manipulation of schemas. There are two main theoretical results associated with schemas. The first is sometimes referred to as the *schema theorem* and the second goes by the name of *implicit parallelism*. We summarize each of these results below. Unless noted otherwise, we restrict ourselves to the standard bit coding introduced in Subsection 9.3.2.

The essence of the schema theorem is that templates with better-than-average fitness values (where the template average is taken over all $2^{\text{no. of free elements}}$ possible chromosomes) will dominate the population as generations proceed (and conversely, the influence of templates with below-average fitness values will dwindle over generations). For consistency with prior work in this area, we suppose that the fundamental problem of minimizing $L(\boldsymbol{\theta})$ has been mapped into comparing *strictly positive* fitness values with the higher fitness value being better. Holland (1975, Corollary 6.4.1) and Michalewicz (1996, expression (3.3)) give a lower bound to the rate at which the domination will occur across generations under the basic GA operations of roulette selection, crossover, and mutation that were discussed in Section 9.4. Let $\varphi(S, k) \leq N$

denote the number of chromosomes in the population of size N that are instances of the schema template S at iteration k and *fitness*(S) represent the mean fitness of all the chromosomes that are instances of S.

The bound from the schema theorem under the basic GA of Section 9.5 with $N_e = 0$ (i.e., no elitism), crossover rate $0 < P_c \leq 1$, and mutation rate $P_m \approx 0$ is

$$\min\left\{\varphi(S,k)\frac{fitness(S)}{(mean\ fitness)_k}(1 - c_0P_c - c_1P_m), N\right\} \leq \varphi(S,k+1) \leq N, \quad (10.3)$$

where c_0 and c_1 denote constants that depend on the bit string length and characteristics of the schema configuration (e.g., the order of the schema), and the indicated ratio measures the fitness of the specified schema relative to the mean fitness of all the chromosomes in the population at iteration k, $(mean\ fitness)_k$. (The "min" operator in (10.3) ensures that $\varphi(S,k+1) = N$ if the bound would otherwise be larger than N.) As a consequence of the definitions of c_0 and c_1, the $-c_0P_c - c_1P_m$ term will be nearly zero in most circumstances when there are relatively few "don't care" symbols (*) in the schema of interest, S. Hence, (10.3) implies that a schema S that consistently has a fitness value higher (better) than the possibly increasing average of all the chromosomes (i.e., $fitness(S)/(mean\ fitness)_k > 1$) will dominate in the sense that more and more chromosomes will be instances of that schema over generations.

The GA literature often refers to (10.3) as indicating that good schemas will have an exponentially growing influence across generations. However, the use of *exponentially growing* in this context is not fully justified: (i) the time-varying $(mean\ fitness)_k$ term may (at least partly) neutralize the exponential growth that would result if $(mean\ fitness)_k$ were to remain constant (i.e., a product such as $(1+1/2)(1+1/3)(1+1/4)\ldots$ does not represent exponential growth; see Stephens and Waelbroeck, 1999, for a specific GA example) and (ii) there is a Taylor series approximation associated with the role of the mutation probability (see Holland, 1975, p. 111) that makes the bound in (10.3), strictly speaking, only an approximation. Despite these potential shortcomings, the schema theorem as expressed in (10.3) provides some insight into the underlying mechanics of the standard GA (sans elitism) described in Section 9.5.

The second widely cited schema result, implicit parallelism (or *intrinsic* parallelism in Holland, 1975), states that the number of schemas processed by the algorithm in one generation is much larger than N. This suggests that the GA has powerful capabilities to process a greater amount of information at each iteration than would be suggested by the population size alone. Further, this implicit information is available without additional storage and/or processing requirements. Information is obtained about a number of schemas much larger than the population size because a given chromosome can be an instance of many different schemas; each chromosome, therefore, reveals information about the worth of many specific schemas.

It is widely reported in the GA literature that implicit parallelism indicates that $O(N^3)$ schemas are processed at each iteration. However, the conditions for this result are more stringent than much of this literature would suggest, as discussed below. The claimed $O(N^3)$ result is also frequently cited as a basis for the optimality of coding with the standard bit (binary) sequence (via the claim that this coding maximizes the number of schemas that are implicitly processed). However, this claim has been shown to be incorrect, as summarized in Fogel (2000, pp. 75–76); this claim is also wrong in the sense of violating the no free lunch theorems of Subsection 1.2.2 and Section 10.6. Fogel (2000, p. 76) notes that bit coding has been formally shown to offer no particular advantage unless the problem is well mapped to a sequence of binary decisions.

There are also many numerical examples where bit coding performs worse than other coding schemes, primarily real-number coding. Some of these studies are summarized in Section 9.7 and in Davis (1996, Chap. 4), Michalewicz (1996, p. 54), and Mitchell (1996, Chap. 4). Davis, for example, notes that in nine real-world applications he has solved with GAs, the bit string encoding was always outperformed by other coding schemes. The examples in Michalewicz and Mitchell are of a more conceptual (versus empirical) nature, pointing to potential problems with the underlying "building block" basis of schema theory when the chromosome-based equivalent of a loss function evaluation involves a large amount of interaction among the genes in a chromosome (i.e., high epistasis, as discussed in Subsection 9.3.4).

A more careful derivation of the implicit parallelism bound on schemas is given in Bertoni and Dorigo (1993). Their results apply to the standard bit coding and to a GA with crossover but no mutation (it appears that including a small mutation probability would not substantially change the bounds reported). We now summarize these results. Let $\beta > 0$ be such that $N = 2^{\beta t}$, where t is proportional to the string length B according to $t \equiv BP^u/2$, with P^u an upper bound to the probability that one of the schemas gets destroyed by crossover (not to be confused with an upper bound to the crossover probability P_c). Bertoni and Dorigo (1993) show that there is a strong relationship between β and the implicit parallelism bound, which is tantamount to the relative size of the population and the string length having a strong influence on the degree of implicit parallelism.

The central result in Bertoni and Dorigo (1993) states that a lower bound on the expected number of disjoint schemas to be processed at each iteration is of the form $N^{f(\beta)}/\sqrt{\log_2 N}$, where $f(\beta) > 0$ is defined differently for each of three ranges of β.[1] Only when $\beta = 1$ does the resulting bound correspond to the widely cited $O(N^3)$ bound. As an illustration of the large differences from the widely cited case, if $\beta = 0.1$ (i.e., a small population size relative to the string length), then the bound is $N^{21}/\sqrt{\log_2 N}$, while if $\beta = 10.0$ (a large population size

[1]This result relies on the hypothesis of a uniformly random population. Because the GA operations of selection and reproduction will skew the population after the initial random population, the results do not strictly hold in subsequent generations.

relative to the string length) the bound is $N^{0.317}/\sqrt{\log_2 N}$. A further consequence of the results in Bertoni and Dorigo (1993) is that it is misleading to refer to any of these bounds as "big O" type bounds. When N changes, β changes, and with that, the *form* of dependence on N changes (so a dependence, say, on N^3 may change to a dependence on N^2 when the value of N increases). Hence, while the GA literature discussing specific order bounds for implicit parallelism is not entirely correct, the literature may be *qualitatively* correct in arguing that the GA tends to implicitly process a larger number of schemas than the population size.

There is considerable controversy about the implications of the above schema results on practical implementations of GAs. It is clear that the notion of a schema is a somewhat removed from the actual issues an analyst faces in solving an optimization problem. The importance of good overall templates on specific solutions must be accepted with a degree of faith. The fact that many good schemas are processed in a particular generation of a GA may or may not be relevant to reaching a good solution with the chromosomes actually processed. For example, Baum et al. (2001) discuss a problem where the schema theorem in (10.3) applies, but where a standard GA is very inefficient (some modifications to the GA can make it much more efficient in this specific problem).

Although the theory discussed in some of the earlier chapters had limitations (generally due to the asymptotic nature), it was more germane to actual algorithm performance since it dealt directly with convergence and rates of convergence of the θ-based iterates. In contrast, schema theory describes a notion that is related only indirectly to algorithm performance. Further, in convergence theory, seemingly minor approximations must be carefully handled due to their potential impact over the course of the many iterations of an asymptotic analysis. Despite these reservations, schema theory provides some of the intuitive rationale for why GAs are often effective on challenging problems (Stephens and Waelbroeck, 1999).

10.4 WHAT MAKES A PROBLEM HARD?

One of the most interesting collateral aspects of the field of EC in general and GAs in particular has been the study of what makes a problem inherently difficult. Of course, this has also long been of concern to those involved in the *general* theory and practice of search and optimization (e.g., Hromkovič, 2001).

The motivation for researchers in EC has arisen largely from observed performance. Despite their success in a wide variety of applications, GAs have also failed in a large number of problems. In fact, GAs may perform significantly poorer than even a simple random bit-flipping algorithm (e.g., Davis, 1991;

Mitchell, 1996, pp. 129–130).[2] This has prompted researchers and analysts to try to analyze the essential characteristics of problems in which GAs fail. Not surprisingly, some of the difficulties seen in earlier chapters (high dimensionality, challenging nonlinearities, noise in the loss/fitness evaluations, etc.) also hamper GA performance (see, e.g., Kargupta and Goldberg, 1997; Goldberg, 2002, pp. 76–100). Beyond some of these obvious hindrances, GA researchers have looked for problem aspects that may uniquely hamper a GA. Unfortunately, identifying what makes a problem "GA hard" is hard itself! If this were not so, we would have insight into the dual question of what makes a problem appropriate for a GA, contradicting the statements in Section 9.8 on the difficulty of knowing a priori whether a GA will be a good match to a problem.

One of the characteristics long associated with problem difficulty is *deception*, a term introduced by Goldberg (1987). This term is motivated by the schema-based interpretation of how GAs work. Namely, in the early iterations of the algorithm, lower-order schemas (those with a large proportion of * symbols) should provide useful information about subsequent higher-order schemas that are closer to the full chromosome representations. This process is sometimes referred to as the *building block hypothesis* (see, e.g., Goldberg, 1989, pp. 41–45; Stephens and Waelbroeck, 1999). Deception occurs when the lower-order schemas provide misleading information about what are the best chromosomes.

As an extreme example of deception, suppose that any schema of order less than B whose defined (non-*) bits are all 1's is the best among schemas of the same order but that in the full-order (no * bits) schema, [0 0 0 ... 0] offers the best fitness. Then, in principle, it should be difficult for the GA to find the optimal value [0 0 0 ... 0] since all lower-order schemas suggest that having many 1's is good and the schema theorem (Section 10.3) will tend to emphasize the schema with many 1's. In fact, in analyzing a test suite of problems where the GA performed poorly, Forrest and Mitchell (1993) and Goldberg et al. (1993) offered the lack of information from lower-order schemas and consequent hampering of the crossover process as a reason for a GA's poor performance. However, despite the intuition behind this schema-based explanation of difficulty through deception, there is ample evidence that deception alone is neither necessary nor sufficient to cause difficulties for a GA. Some discussion of this is given in Mitchell (1996, pp. 125–127).

Epistasis is another problem characteristic that has been cited as creating difficulties in GAs (e.g., Reeves and Wright, 1994). However, as with deception, it is not clear that epistasis will always have negative consequences (other things being equal). Heckendorn and Whitley (1999), for example, point out that the effect of epistasis is subtler than previously believed. Their analysis is based on the concept of *Walsh polynomials*, which can be used as alternative

[2]Random bit flipping refers to a class of bit-based analogues to the random search algorithms of Chapter 2. As with random search, there are many variations of random bit flipping. The most basic form is to start with a single chromosome (not a population) and randomly choose a bit (or bits) to change in value, accepting the new chromosome if the loss (fitness) is improved.

representations of fitness functions defined on the domain $\{0, 1\}^B$ (i.e., the fitness function relies on a standard bit-based, Bth-order chromosome as the argument). Naudts and Kallel (2000) show by theory and example that the traditional measures for "GA easy" or "GA hard" are easily contradicted. That is, a function that appears difficult to optimize based on measures such as deception may, in fact, be easy. The opposite is also true. They conclude that any static measure of problem difficulty will always be severely limited.

10.5 CONVERGENCE THEORY

We have yet to analyze the formal convergence of EC algorithms along the lines of the probabilistic convergence analysis of Chapters 2–8. Although the schema theory in Section 10.3 has played a role in understanding why GAs may sometimes be effective, it has not been especially useful in addressing the question of convergence. There have, however, been convergence results established by other means. This section summarizes some of these results.

One of the reasons that schema theory has not been used toward formal convergence analysis is the following negative result due to Rudolph (1994). This result applies to a *canonical GA* having strictly positive (noise-free) fitness values, bit coding, crossover, and mutation, but no elitism. The canonical GA is the basic algorithm of Holland (1975) and one for which the schema theorem and implicit parallelism (Section 10.3) apply. Let $\hat{L}_{\min,k}$ denote the lowest of the N loss values within the population at iteration k. That is, $\hat{L}_{\min,k}$ represents the loss value for the $\boldsymbol{\theta}$ in population k that has the maximum fitness value. The following negative result applies.

Theorem 10.1. A canonical GA with roulette wheel selection (Subsection 9.4.1), $0 \leq P_c \leq 1$, and $0 < P_m < 1$ does not converge in the sense that

$$\lim_{k \to \infty} P\left(\hat{L}_{\min,k} = L(\boldsymbol{\theta}^*)\right) \neq 1 .$$

Comments on proof. The proof is given in Rudolph (1994, Theorems 3 and 4). The proof is based on Markov chains (see Appendix E). The GA can be represented as a Markov chain with 2^{NB} states, each representing a possible population. ❑

One implication of Theorem 10.1 is that in the convergence sense, the canonical GA performs even more poorly than a brute-force enumeration or blind random search (algorithm A in Subsection 2.2.2), both of which are notoriously poor algorithms in all but the simplest practical problems. The essential problem with the canonical GA is that it has no way to guarantee keeping the best solution it finds; even if the algorithm finds an optimum, it is

likely to be lost through subsequent crossover and/or mutation. On the other hand, enumeration is guaranteed to converge since the search space containing the solution has a finite number of elements (i.e., the bit representation allows only a finite number of possible outcomes for each chromosome) and the best solution is retained. The search will definitely "hit" (and keep) the best solution at some finite number of iterations, although "finite" in this case must be interpreted judiciously. Achieving the required number of iterations may take millennia on the fastest available computers. Likewise, modest conditions are given in Subsection 2.2.2 for convergence (a.s.) of blind random search.

Although Theorem 10.1 is interesting, and points to one of the reasons that people have not been able to use schema theory (or any other method!) to show convergence of the canonical GA, it has limited practical implications. Almost any serious GA implementation will include elitism and possibly other enhancements (such as in Section 9.6). Of course, such enhancements complicate the theoretical analysis, but some positive results are available, as outlined later. Further, Theorem 10.1 says nothing about *finite-sample* performance of a GA, and, in fact, a simple canonical GA may in some cases produce a satisfactory finite-iteration solution (see Holland, 1975, pp. 161–164; De Jong, 1975, pp. 96–101).

The following small-scale example provides some intuitive sense of how a canonical GA may fail to converge. The same principles apply to general problems (larger N and/or B), as illustrated in Exercise 10.9.

Example 10.1—Failure of convergence for a GA. Consider a very small scale canonical GA with $N = 2$ and $B = 3$. Let $P_c > 0$ and P_m be negligibly small. The total number of possible populations is $2^{NB} = 64$ (i.e., at every iteration, the GA will produce one of these 64 populations). Among these 64 states are $2^B = 8$ states where the two population elements are identical. In these eight states, crossover between the two population elements will not change the elements. For example, if the population has the two chromosomes {[1 0 1], [1 0 1]}, the population after crossover is also guaranteed to be {[1 0 1], [1 0 1]}. By the principles of roulette selection there is a nonzero probability of reaching any one of these eight absorbing states because they all have nonzero fitness values. Because the mutation probability is negligible, the GA becomes "stuck" after reaching one of these states. (Even if the population should eventually change due to a mutation, the selection/crossover process will repeat itself and continue pushing the GA to the absorbing states.) Unless the chromosomes in the absorbing states correspond to $\boldsymbol{\theta}^*$ upon decoding, the GA does not converge to the optimum. ❑

On a more positive note, conditions for the formal convergence of EC algorithms to an optimal $\boldsymbol{\theta}^*$ are presented in a number of references. Qi and Palmeiri (1994), Hart (1997), Rudolph (1997a, 1998), Vose (1999, Chaps. 13 and 14), Fogel (2000, Chap. 4), and Kallel et al. (2001) are several of the references that present conditions for EC convergence.

Let us summarize an approach to convergence and convergence rates based on Markov chains. Consider a bit-coded GA with (as usual) a population size of N and a string length of B bits per population element (chromosome). Hence, there are 2^B possible strings for an *individual* chromosome. Then the total number of possible *unique* populations is "$N+2^B-1$ choose N":

$$N_P \equiv \binom{N+2^B-1}{N} = \frac{(N+2^B-1)!}{(2^B-1)!N!} \tag{10.4}$$

(Exercise 10.7). (Eqn. (10.4) differs from the 2^{NB} states mentioned in the context of Theorem 10.1 because of redundancy among the 2^{NB} states. For example, with $N = 2$ and $B = 3$, the two populations {[1 1 0], [0 1 0]} and {[0 1 0], [1 1 0]} are counted only once in (10.4).) One of the ways to analyze the performance of GA is to determine the probability that a particular population contains a chromosome corresponding to an optimum $\boldsymbol{\theta}^*$.

It is possible to construct an $N_P \times N_P$ Markov transition matrix $\boldsymbol{P}$, where the ijth element is the probability of transitioning from the ith population of N chromosomes to the jth population of the same size. Let $\boldsymbol{p}_k$ be an $N_P \times 1$ vector having jth component $p_k(j)$ equal to the probability that the kth generation will result in population $j, j = 1, 2, \ldots, N_P$. From basic Markov chain theory (Appendix E),

$$\boldsymbol{p}_{k+1}^T = \boldsymbol{p}_k^T \boldsymbol{P} = \boldsymbol{p}_0^T \boldsymbol{P}^{k+1},$$

where $\boldsymbol{p}_0$ is an initial probability distribution. If the chain is irreducible and ergodic, the limiting distribution of the GA (i.e., $\overline{\boldsymbol{p}}^T = \lim_{k\to\infty} \boldsymbol{p}_k^T = \lim_{k\to\infty} \boldsymbol{p}_0^T \boldsymbol{P}^k$) exists and satisfies the stationarity equation $\overline{\boldsymbol{p}}^T = \overline{\boldsymbol{p}}^T \boldsymbol{P}$. (Recall from Appendix E that irreducibility indicates that any state is accessible from any other state.) An individual element in $\boldsymbol{P}$ can be computed according to the formulas in Suzuki (1995) and Stark and Spall (2001). These elements depend in a nontrivial way on N, the crossover rate, and the mutation rate; the number of elite chromosomes is assumed to be $N_e = 1$.

Suppose that $\boldsymbol{\theta}^*$ is unique (i.e., Θ^* is the singleton $\boldsymbol{\theta}^*$). Let $J \subseteq \{1, 2, \ldots, N_P\}$ be the set of indices corresponding to the populations that contain at least one chromosome representing $\boldsymbol{\theta}^*$. So, for example, if $J = \{4, N_P - 2\}$, then each of the two populations indexed by 4 and $N_P - 2$ contains at least one chromosome that, when decoded, is equal to $\boldsymbol{\theta}^*$. Under the above-mentioned assumptions of irreducibility and ergodicity, $\sum_{i\in J} \overline{p}_i = 1$, where $\overline{p}_i$ is the ith element of $\overline{\boldsymbol{p}}$. Hence, a GA with $N_e = 1$ and a transition matrix that is irreducible and ergodic converges in probability to $\boldsymbol{\theta}^*$. This result should not be surprising. The assumptions on $\boldsymbol{P}$ imply that the GA will eventually visit every state. Because the algorithm is always saving the best population chromosome encountered (knowable here because of the noise-free loss/fitness measurements), the

algorithm will ultimately be in a state containing the best possible chromosome, a chromosome equaling θ^* upon decoding.

To establish the fact of convergence alone, it may not be necessary to compute the $\boldsymbol{P}$ matrix. Rather, it suffices to know that the chain is irreducible and ergodic. (For example, Rudolph, 1997a, p. 125, shows that the Markov chain approach yields convergence when $0 < P_m < 1$.) However, $\boldsymbol{P}$ must be explicitly computed to get the *rate* of convergence information that is available from $\boldsymbol{p}_k$. Unfortunately, this is rarely possible in practice. The dimension N_P grows very rapidly with increases in the number of bits B and/or the population size N. An estimate of N_P can be obtained by Stirling's approximation to a factorial (see Exercise 10.8). Even a modest-sized GA can result in a very large $\boldsymbol{P}$. For example, if $N = B = 6$, $\boldsymbol{P}$ is larger than a $10^8 \times 10^8$ matrix. Nevertheless, even with the very rapid growth in N_P, it is possible to use the Markov chain analysis to analyze the convergence rate in some problems, as we now illustrate.

Example 10.2—Convergence rate for a GA. Consider the loss function with scalar θ from Schwefel (1995, pp. 328–329): $L(\theta) = -|\theta \sin(\sqrt{\theta})|$, $\theta \in \Theta = [0, 15]$. The function is shown in Figure 10.1. There is a local minimum near 5.2 and a global minimum at the boundary point, $\theta^* = 15$. Using the formulas for the elements of $\boldsymbol{P}$ in Suzuki (1995) and Stark and Spall (2001), Table 10.1 gives the probability of the GA finding one of the unique populations containing an optimal chromosome from within the N_P unique populations. Given the decoding process in Subsection 9.3.2, an optimal chromosome is correct to within an

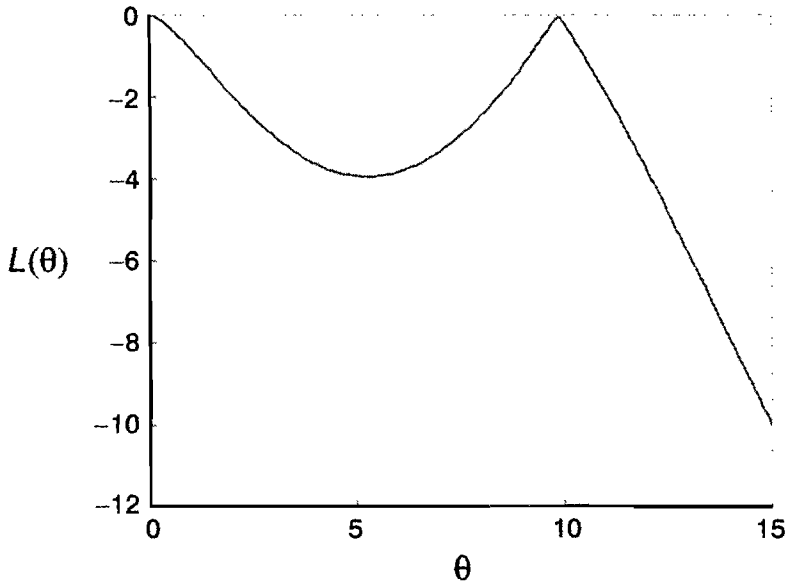

Figure 10.1. Loss function for use in Example 10.2. With $\Theta = [0, 15]$, local minimum is at $\theta \approx 5.2$; global minimum is at $\theta^* = 15$.

Table 10.1. Probability of the GA finding a population that contains an optimal chromosome.

GA coefficients	Iteration number						
	0	5	10	20	30	50	100
$P_c = 1.0$ and $P_m = 0.05$; $N = 2$ and $B = 6$	0.03	0.08	0.15	0.32	0.48	0.74	0.97
$P_c = 1.0$ and $P_m = 0.05$; $N = 4$ and $B = 4$	0.21	0.51	0.69	0.92	1.00	1.00	1.00

accuracy related to N and B (the resulting quantity d in Subsection 9.3.2 governs the accuracy). It is assumed that the initial population is drawn according to a uniform distribution over the unique populations (see Exercise 10.11). The values for N and B are very small by the standards of most practical GA implementations. This was done to keep the computations for the transition matrix $\boldsymbol{P}$ manageable (the calculations for each element of $\boldsymbol{P}$ are significant and there are many elements). In particular, $\boldsymbol{P}$ has dimension 2080×2080 when $N = 2$ and $B = 6$ and dimension 3876×3876 when $N = 4$ and $B = 4$. ❑

Let us close this section with a brief discussion of other approaches to convergence rates. Some of these approaches use methods other than Markov chains. For example, when the algorithm applies directly to $\boldsymbol{\theta}$ without a coding process and with real numbers for at least some elements of $\boldsymbol{\theta}$, there is no longer a finite number of states, making it less amenable to Markov chain analysis. Stochastic approximation methods (Section 4.3) have been used for convergence analysis of some forms of EC (e.g., Yin et al., 1996).

Some of the available results for convergence *rates* are for EC algorithms using mutation and selection only, or using crossover and selection only. Both Beyer (1995) and Rudolph (1997a) examine ES algorithms that include selection, mutation, and crossover. The function analyzed in both cases is the simple spherical loss $L(\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{\theta}$. Convergence rates based on the spherical fitness function are somewhat useful in practice if it is assumed that the sphere approximates a local basin of attraction for the loss function of real interest. A number of other convergence rate results are also available for this simple function—for example, Qi and Palmeiri (1994) for real-valued GA. Rudolph (1997b) considers convergence rates for an ES algorithm applied to a subclass of convex functions (but a subclass broader than only $L(\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{\theta}$). He establishes conditions such that $\hat{L}_{\min,k} - L(\boldsymbol{\theta}^*) = o(c^k)$ a.s. for some $1/2 < c < 1$. This implies that the best loss value in the population at each k converges to $L(\boldsymbol{\theta}^*)$ at a rate *faster* than c^k goes to zero as $k \to \infty$.

10.6 NO FREE LUNCH THEOREMS

Subsection 1.2.2 introduced the basics of the no free lunch theorems, showing that no algorithm can be universally more efficient than other algorithms. We now build on this earlier discussion by presenting some aspects of the NFL theory in a more formal manner. Although the implications of this theory extend to arbitrary search and optimization methods, this discussion is appearing in a chapter devoted to EC algorithms because most of the critical work has come from those in the EC field. A premonition of the NFL results appeared in Goldberg et al. (1993): "...the fiddling with codes, operators, and parameters that characterize so much of the GA literature seriously challenges any claims that the promised land of robustness has been achieved." The NFL theorems indicate that this "promised land" can never be achieved.

In their NFL theory, Wolpert and Macready (1997) present a formal analysis of general search algorithms for optimization, including, of course, the EC class considered here, but also encompassing simulated annealing, random search, and so on. (Some related philosophical discussion and a summary of earlier references are given in Goldberg, 2002, pp. 74–75.) Wolpert and Macready (1997) present several theorems linked to two general approaches to the question of general algorithm efficiency. One of their approaches is to compare the performance of algorithms over the set of possible optimization problems; the other is to compare performance for a particular problem over a specified collection of algorithms. Most of their results (and the ones we discuss here) focus on the former type of comparison. The essence of the NFL theorems is that the performance of any optimization algorithm, when averaged across all possible problems, is identical to the performance of any other algorithm. Of course, these results do not reflect the usual types of prior information that might be available to the algorithms and thus may not adequately reflect the performance of algorithms as they are actually applied. Nevertheless, the NFL results are an antidote to inflated claims of efficiency that have appeared in some of the EC and other literature.

Recall from Subsection 1.2.2 that the NFL results apply to discrete problems where there are $N_\theta < \infty$ possible values for $\boldsymbol{\theta}$ (i.e., N_θ points in Θ) and $N_L < \infty$ possible values for the loss (fitness) function. (We discuss NFL in the context of noise-free loss measurements; the same ideas apply when there are N_L possible noisy values y.) Note that the finiteness assumptions for the search space Θ and associated space of possible loss values are met in practice for algorithms implemented on digital computers (i.e., 32- or 64-bit representations of real numbers). Let Λ represent the set of $(N_L)^{N_\theta}$ possible mappings $L(\boldsymbol{\theta})$. Each element in Λ represents *one* set of rules that takes the N_θ possible values for $\boldsymbol{\theta}$ and maps them into the N_L possible values for the loss. Example 1.7 (Subsection 1.2.2) illustrates a simple case where Λ has eight possible mappings. The number of possible mappings is often huge ($>10^{1000}$ is common when considering digital computer implementations of continuous problems).

Suppose that an algorithm $\mathcal{A}$ is applied to a loss function L. Let $\hat{L}^{(n)}$ represent the loss value reported out of the algorithm after n unique loss evaluations (multiple evaluations that result when an algorithm revisits a point in Θ are only counted once). For example, if $\mathcal{A}$ represents a particular EC algorithm, $\hat{L}^{(n)}$ may represent the best (lowest) loss value from the most recent N population elements after the algorithm has performed $n \geq N$ unique loss evaluations. If elitism is included, then $\hat{L}^{(n)}$ is guaranteed to be the lowest loss value encountered over *all* of the n loss evaluations.

For any given problem, it is likely that the performance of one algorithm will be superior to others. A priori, of course, it is rarely possible to know which algorithm is superior. NFL theorems do not address the performance of a specific algorithm applied to a specific loss function. Rather, they compare the performance of algorithms over *all* problems, where each problem (each mapping L) is considered equally likely. In particular, assume that the prior probability of encountering any given problem is the same as the probability for any other problem (i.e., the prior distribution on the set of all mappings [loss functions] in Λ is the uniform distribution). An NFL theorem based on summing over Λ is:

Theorem 10.2 (NFL). Consider any pair of algorithms $\mathcal{A}_1, \mathcal{A}_2$ relying on a fixed number of unique loss (fitness) evaluations n. For the discrete optimization structure outlined above and any point λ in the set of N_L allowable values for the loss:

$$\sum_{L \in \Lambda} P\left(\hat{L}^{(n)} = \lambda \middle| L, \mathcal{A}_1\right) = \sum_{L \in \Lambda} P\left(\hat{L}^{(n)} = \lambda \middle| L, \mathcal{A}_2\right),$$

where the probability $P\left(\hat{L}^{(n)} = \lambda \middle| L, \mathcal{A}_i\right)$ is conditional on one of the $(N_L)^{N_\Theta}$ mappings and on the specific algorithm.

An important special case is where λ is the minimum of the possible values for the loss (this minimum is achievable in only those mappings where it lies in the range space). Then, we are considering the probabilities of $\hat{L}^{(n)}$ achieving the optimum for those mappings where λ is in the range space; in other mappings, the probability $P\left(\hat{L}^{(n)} = \lambda \middle| L, \mathcal{A}_i\right) = 0$.

According to the above NFL theorem, the average efficiency of all algorithms is the same in the sense that $\sum_{L \in \Lambda} P\left(\hat{L}^{(n)} = \lambda \middle| L, \mathcal{A}_i\right)$ is independent of $\mathcal{A}_i$. A self-evident implication of the above theorem is: If $\mathcal{A}_1$ has a faster rate of convergence than $\mathcal{A}_2$ for one set of problems, then there is a set of problems for which $\mathcal{A}_2$ has a faster rate of convergence than $\mathcal{A}_1$. A related implication is that no algorithm can have better overall efficiency than blind random search (algorithm A in Subsection 2.2.2). Simply knowing that a loss function has a particular structure does not generally make it a priori preferable to use one

algorithm over another. However, if the problem structure is used in the algorithm design or if it is known that the algorithm and problem are well matched, then it *is* possible to overcome the limits of NFL in the sense that the algorithm may outperform blind random search and other methods.

10.7 CONCLUDING REMARKS

This completes the second of two chapters devoted to evolutionary computation. EC algorithms represent an abstraction of natural evolutionary processes. While various EC approaches have historically been placed in "bins" (GA, ES, EP, etc.), the distinctions have become less important. Effective implementations of EC often borrow aspects from several of the named approaches to achieve effective practical performance.

We summarized some of the theory available for EC, focusing on the traditional schema theory and on the Markov chain-based approach. Over the last several years, schema theory has fallen somewhat into disfavor. This seems to have happened because some used the theory to make performance claims that, under more careful scrutiny, were not actually supported by the theory. That is, while the theory *itself* is largely correct (subject to some "adjustments" and caveats), many of the stated *implications* have not been correct. Nevertheless, schema theory has historical significance in EC development and, in an appropriately restricted sense, provides some intuitive justification for the good performance that is frequently observed. Theory based on Markov chains and other forms of probabilistic analysis is more directly connected to the performance of EC methods. Using such theory, it can be shown that typical EC algorithms *with elitism* converge to an optimal point.

Because EC is based on analogies to natural evolution, it was long believed by some that EC algorithms are "best" in some sense. This is now known to be false. Aside from the fact that evolution itself does not appear to be an optimized process, the no free lunch theorems formalize the intuitively sensible notion that no algorithm can be universally preferred.

What is one to conclude about the efficacy of EC algorithms for challenging optimization problems? Essentially, there is no reason to believe that GAs or other EC algorithms will be generally superior to other methods. In particular, simpler methods such as random search in Chapter 2 (or the FDSA or SPSA methods of Chapters 6 and 7) may be as (or more) numerically efficient on a particular problem. We saw evidence of this in the numerical studies of Chapter 9. In fact, the appeal of the simpler approaches goes beyond numerical efficiency as analyzed via the NFL theorems; one should also consider the human cost associated with the relative complexity of many EC implementations. In spite of NFL, Culberson (1998) and Baum et al. (2001) discuss some cases where the EC framework may be beneficial. One area is in combining the results from various local searches. However, much further research is required to identify a broad class (or classes) of problems for which

EC algorithms are especially powerful. It does not appear that such results will be available in the near future.

Nevertheless, despite the issues mentioned above, EC methods have proven very effective in many challenging practical problems. Their popularity is expected to grow in the years to come.

EXERCISES

10.1 Suppose that $B = 4$ and that a standard bit representation (Subsection 9.3.2) is used. Suppose the fitness function is the integer represented by the binary argument (so, e.g., the fitness of [0 1 0 1] is 5). Contrast the mean fitness of the schemas [1 * * *] and [0 * * *].

10.2 Suppose that $B = 7$ and consider two schemas [* 1 * * * * 0] and [* * * 1 0 * *]. Identify one chromosome that is an instance of both of the schemas. Illustrate with this chromosome why the schema with the greater number of successive * positions (the first schema here) is more likely to be destroyed by single-point crossover. Now suppose that two matings will take place and that for the first mating, one of the chromosome parents is an instance of the first schema and the other is not. Likewise for the second mating and schema. What is the probability (conditional on the supposition) of each of the two schemas surviving a single-point crossover? (This problem illustrates that schemas where the non-* positions are clustered together are more likely to survive into future generations than other schemas.)

10.3 Prove that a specific chromosome of string length B is an instance of 2^B schemas (the null representation [* * * ... *] counts as a schema; likewise, a specific chromosome is a schema with zero "don't care" symbols).

10.4 An intermediate calculation for the implicit parallelism bound of Section 10.3 is the maximization of $\binom{2t}{x}$ with respect to x (t and x are positive integers). Show that $x = t$ is the solution to this maximization problem.

10.5 With noisy loss (fitness) measurements or other randomness, it can be shown that the proportion of a particular schema in a future population is, under the appropriate conditions, $S/(S+S')$, where S and S' are independent random variables describing, respectively, the fitness of the schema and the fitness of its complement (i.e., the fitness of all schemas disjoint from the schema under consideration). (Fogel, 2000, pp. 119–120, considers this formulation.) For $S \sim U(0, \mu)$ and $S' \sim U(0, \mu')$, with $\mu > 0$ and $\mu' > 0$, compute the *expected* proportion for the particular schema.

10.6 Consider the function $L(\boldsymbol{\theta}) = \sum_{i=1}^{15} |t_i| + \prod_{i=1}^{15} |t_i|$, where $\boldsymbol{\theta} = [t_1, t_2, \ldots, t_{15}]^T$ and $\Theta = [-1.5, 1.5]^{15}$. Implement two versions of a (10, 100)-ES algorithm: one with a constant covariance matrix $\boldsymbol{D}_{\text{parent}}$, and one with a dynamic matrix changed according to (10.2). Tune the two algorithms as appropriate and enforce the constraints by regenerating $\boldsymbol{\theta}_{\text{child}}$ values as needed. Generate the initial populations randomly (uniformly) in Θ and generate the $\boldsymbol{\theta}_{\text{parent}}$

according to the idea in the discussion below (10.2). Run both algorithms for 40 independent replications of 10 iterations per replication. Determine whether there is statistical evidence for the superiority of the form using dynamic matrix updating; use the appropriate two-sample test (Appendix B) applied to the lowest loss value in the final population.

10.7 (More difficult.) Derive the formula for N_P shown in (10.4) (i.e., why "$N+2^B-1$ choose N"?).

10.8 Stirling's approximation to a factorial is $K! \approx K^K e^{-K}\sqrt{2\pi K}$. Use this formula to derive an approximation to N_P in (10.4). Compare the exact formula for N_P and the approximation at $N = 8$ and $B = 6$.

10.9 A generic form of a Markov transition matrix for a GA with selection, crossover, and possibly elitism is $\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{Q} \end{bmatrix}$, where $\boldsymbol{I}$ is an identity matrix for the absorbing states (populations), $\boldsymbol{R}$ is a matrix of probabilities for the transition from transient states to the absorbing states, and $\boldsymbol{Q}$ is a matrix of probabilities to the transient states. Let $\max_i|\lambda_i| < 1$, where λ_i is the ith eigenvalue of $\boldsymbol{Q}$. Show that the GA converges to the absorbing states as the number of iterations increase. (This can be used to show convergence or nonconvergence to an optimum, depending on whether the absorbing states include a population with an optimal chromosome.) (Hint: The matrix relationships in Appendix A [Section A.2] may be useful.)

10.10 Consider a very small problem where $N = 2$ and $B = 2$ for a bit-coded GA.
(a) Verify by enumeration that there are 2^{NB} possible states (populations).
(b) List the N_P unique states (eqn. (10.4)).
(c) Identify the absorbing states when there is no mutation.

10.11 For the two settings represented in Table 10.1 ($N = 2$, $B = 6$ and $N = 4$, $B = 4$), compute the probability of the initial population containing the optimal chromosome *without* restricting to only the N_P unique populations. That is, assume that the initial population is drawn according to a uniform distribution over all 2^{NB} possible populations. Comment on why the probabilities here may differ slightly from the corresponding probabilities in Table 10.1.

10.12 Suppose that $\Theta = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ and that there are four possible outcomes for the noise-free loss measurements, $\{L_1, L_2, L_3, L_4\}$. Demonstrate the implications of the NFL theorems by showing that the mean performance of any two algorithms is the same across all possible mappings.