Aggregating Models of Evolutionary Algorithms

William M. Spears
AI Center - Code 5514
Naval Research Laboratory
Washington, DC 20375
spears@aic.nrl.navy.mil

Abstract- This paper summarizes two useful techniques for aggregating theoretical models of evolutionary algorithms (EAs). Aggregation removes unnecessary detail from the models, producing simpler models with good predictive accuracy. The first aggregation technique is applicable to "equation of motion" models of EAs that have selection and mutation, for a particular class of interesting fitness functions. This form of aggregation introduces no error into the theoretical model. The second aggregation technique is more general – it can be applied to arbitrary Markov chain models of dynamic systems, such as EAs with selection, mutation, and recombination. No assumptions are made about the fitness functions. This form of aggregation introduces only a small amount of error into the theoretical model.

1 Introduction

The standard difficulty in modeling complex adaptive systems is that it is difficult to find the correct level of granularity. If the level of granularity is too coarse (e.g., too many simplifying assumptions have been made), the model is easy to analyze but lacks predictive ability. On the other hand, if the level of granularity is very fine (e.g., all components of the system are faithfully modeled), the model will be predictive but is often computationally intractable. The trick is to find the right level of granularity.

Naturally, the modeling of evolutionary algorithms (EAs) has the same difficulty. Models of EAs tend to either focus on individual components of the algorithms, or on the dynamics of the complete algorithms. In general, component-wise models tend to be fairly simple to analyze, but lack predictive accuracy. Complete models are necessarily more predictive, but are frustratingly difficult to analyze.

One solution to this dilemma is to find methods for aggregating complete models for EAs – the aggregation removes unnecessary detail from the complete models, producing simpler models with good predictive accuracy. Statistical mechanics models of EAs are one attempt to perform just such an aggregation, by focusing on certain "macro" quantities that are predictable e.g., see [9, 10]. The macro quantities emerge from the dynamics at the microscopic level. Unfortunately, in order to apply these methods, the EAs must be modified from those used in practice. Furthermore, various assumptions and principles are invoked that are difficult to justify. Finally, a great deal of human effort is involved in developing the models. Although the statistical mechanics technique is quite promising, it does not yet automatically aggregate EA models.

In general, the two aspects of EAs that create the most difficulty in their modeling (and in the aggregation of the models) is the inclusion of recombination and finite-population effects. By removing recombination and considering only the *expected* (infinite-population) behavior of an EA population, modeling and aggregation become much simpler. Section 2 will show that for an infinite-population EA with selection and mutation (but with no recombination), a system of C^L simultaneous equations (where C is the cardinality of the alphabet and L is the length of the individuals) can be automatically aggregated to only $L+1$ equations, for a particular problem class. No error is introduced by the aggregation. Furthermore this class of functions includes common one-peak functions, many of the two-peak trap and deceptive functions that have been analyzed in the EA community, and even problems with higher multimodality. The efficacy of this automatic aggregation is demonstrated and confirmed via comparison of the theoretical results with empirical results.

Unfortunately, applying this method of aggregation to models of finite-population EAs with recombination appears to be problematic. However, since finite-population EAs (with or without recombination) can be modeled as Markov chains e.g., see [8], an alternative approach is to find general techniques for automatically aggregating Markov chains. Section 4 summarizes a novel technique for accomplishing this form of aggregation [12]. The advantage of this technique is that it will work with arbitrary classes of fitness functions. The disadvantage is that the aggregation is not error free, although the error is often negligible.

2 Aggregating Models of EAs with Selection and Mutation

A population undergoing selection and mutation can be modeled using "equations of motion" that compute the *expected* time evolution of the proportions of the strings (individuals) in the population (equivalently this can be considered to be the evolution of an infinite-size population). In general, since this model will keep track of every possible individual, the model will require a system of C^L simultaneous equations, where L is the string length and C is the cardinality of the alphabet. Let S_i and S_j be arbitrary strings of length L in that alphabet. Let the proportion of a string S_j at time t be denoted as $p_{S_j}{}^{(t)}$. Then the expected time evolution of the system can be computed using the following equations of motion:

$$p_{S_j}{}^{(t+1)} = \sum_{S_i} p_{S_i}{}^{(t)} \frac{f(S_i)}{\overline{f}^{(t)}} p_{S_i,S_j} \quad (1)$$

Equation 1 first considers the proportions $p_{S_i}{}^{(t)}$ of all strings S_i at time t. The proportions are modified by fitness-proportional selection, where $f(S_i)$ is the fitness of S_i and $\overline{f}^{(t)}$ is the average fitness of the population at time t.[1] Finally p_{S_i,S_j} computes the probability of mutating string S_i to string S_j. The result is the expected proportion of string S_j at time $t+1$.[2]

The total system is described by C^L equations, one for each string S_j. Starting with initial proportions $p_{S_j}{}^{(0)}$, the C^L equations are iterated repeatedly to produce the expected time evolution of the system. Even with binary-string representations ($C = 2$), having binary strings of length $L = 10$ will require over 1000 equations. Clearly this makes it hard to deal with realistic problems.

It turns out that a surprising number of fitness functions f can be aggregated in a fashion that greatly simplifies the above model by reducing the number of equations and the number of terms in the equations. To see this, let the alphabet be denoted as $\mathcal{A}$ and let $\alpha \in \mathcal{A}$ be one of the C alleles. Let $\overline{\alpha}$ denote all the other alleles. It turns out that for some classes of problems (fitness functions), only the number of α's in an individual matters. Thus sets of strings with j α's form an equivalence class, and it suffices to have only $L+1$ equations, since there can be anywhere from zero to L α's in a string. This is a dramatic reduction from the C^L equations that would be required in the general case. Equation 1 appears as before:

$$p_{S_j}{}^{(t+1)} = \sum_{S_i} p_{S_i}{}^{(t)} \frac{f(S_i)}{\overline{f}^{(t)}} p_{S_i,S_j} \quad (2)$$

however, in this case S_i refers to any string with i α's, and S_j refers to any string with j α's. The fitness of any string with i α's is the same, and is denoted as $f(S_i)$. The probability of mutating any string with i α's to one with j α's is given by p_{S_i,S_j}.

Suppose $j \geq i$. This means we are increasing (or not changing) the number of α's. To accomplish this, mutation requires that $j-i$ more $\overline{\alpha}$'s are mutated to α's than α's are mutated to $\overline{\alpha}$'s. The mutation probabilities are:

$$p_{S_i,S_j} = \sum_{x=0}^{min\{i,L-j\}} \binom{i}{x} \binom{L-i}{x+j-i} \times \mu^x \left(\frac{\mu}{C-1}\right)^{x+j-i} (1-\mu)^{i-x} \left(1-\frac{\mu}{C-1}\right)^{L-j-x} \quad (3)$$

Let x be the number of α's that are mutated to $\overline{\alpha}$'s. Since there are i α's in the current string, this means that $i-x$ α's are *not* mutated to $\overline{\alpha}$'s. This occurs with probability $\mu^x(1-\mu)^{i-x}$. Also, since x α's are mutated to $\overline{\alpha}$'s then $x+j-i$ $\overline{\alpha}$'s must be mutated to α's. Since there are $L-i$ $\overline{\alpha}$'s in the current string, this means that $L-i-x-j+i = L-x-j$ $\overline{\alpha}$'s are *not* mutated to α's. This occurs with probability $(\mu/(C-1))^{x+j-i}(1-\mu/(C-1))^{L-x-j}$. The combinatorials yield the number of ways to choose x α's out of the i α's, and the number of ways to choose $x+j-i$ $\overline{\alpha}$'s out of the $L-i$ $\overline{\alpha}$'s. Clearly, it isn't possible to mutate more than i α's. Thus $x \leq i$. Also, since it isn't possible to mutate more than $L-i$ $\overline{\alpha}$'s, $x+j-i \leq L-i$, which indicates that $x \leq L-j$. The minimum of i and $L-j$ bounds the summation correctly.

Similarly, if $i \geq j$, we are decreasing (or not changing) the number of α's. Thus one needs to mutate $i-j$ more α's to $\overline{\alpha}$'s than $\overline{\alpha}$'s to α's. The mutation probabilities are:

$$p_{S_i,S_j} = \sum_{x=0}^{min\{L-i,j\}} \binom{i}{x+i-j} \binom{L-i}{x} \times \mu^{x+i-j} \left(\frac{\mu}{C-1}\right)^{x} (1-\mu)^{j-x} \left(1-\frac{\mu}{C-1}\right)^{L-i-x} \quad (4)$$

The explanation is almost identical to before. Let x be the number of $\overline{\alpha}$'s that are mutated to α's. Since there are $L-i$ $\overline{\alpha}$'s in the current string, this means that $L-i-x$ $\overline{\alpha}$'s are *not* mutated to α's. This occurs with probability $(\mu/(C-1))^x(1-\mu/(C-1))^{L-i-x}$. Also, since x $\overline{\alpha}$'s are mutated to α's then $x+i-j$ α's must be mutated to $\overline{\alpha}$'s. Since there are i α's in the current string, this means that $i-x-i+j = j-x$ α's are *not* mutated to $\overline{\alpha}$'s. This occurs with probability $\mu^{x+i-j}(1-\mu)^{j-x}$. The combinatorials yield the number of ways to choose x $\overline{\alpha}$'s out of the $L-i$ $\overline{\alpha}$'s, and the number of ways to choose $x+i-j$ α's out of the i α's. Clearly, it isn't possible to mutate more than $L-i$ $\overline{\alpha}$'s. Thus $x \leq L-i$. Also, since it isn't possible to mutate more than i α's, $x+i-j \leq i$, which indicates that $x \leq j$. The minimum of $L-i$ and j bounds the summation correctly.

In general, these equations are not symmetric ($p_{S_i,S_j} \neq p_{S_j,S_i}$), and in the absence of selection there is a tendency for strings to have roughly L/C α's (e.g., if we are using bit-strings, and the string length is 100, mutation has a tendency to produce strings with roughly 50 0's and 50 1's). When $0.0 < \mu < 1.0$ it is guaranteed that $p_{S_i,S_j} > 0.0$. Note also that if $i = j$ both equations give the same mutation probabilities, which provides a useful check on the correctness of the above equations:

$$p_{S_i,S_i} = \sum_{x=0}^{min\{i,L-i\}} \binom{i}{x} \binom{L-i}{x} \times \mu^x \left(\frac{\mu}{C-1}\right)^{x} (1-\mu)^{i-x} \left(1-\frac{\mu}{C-1}\right)^{L-i-x}$$

[1] The focus on fitness-proportional selection [5] is due solely to its mathematical simplicity. It would not be difficult to extend this work to other selection mechanisms.

[2] Mutation is defined as follows. An allele is picked for mutation with probability μ. Then that allele is changed to one of the other $C-1$ alleles, uniformly randomly.

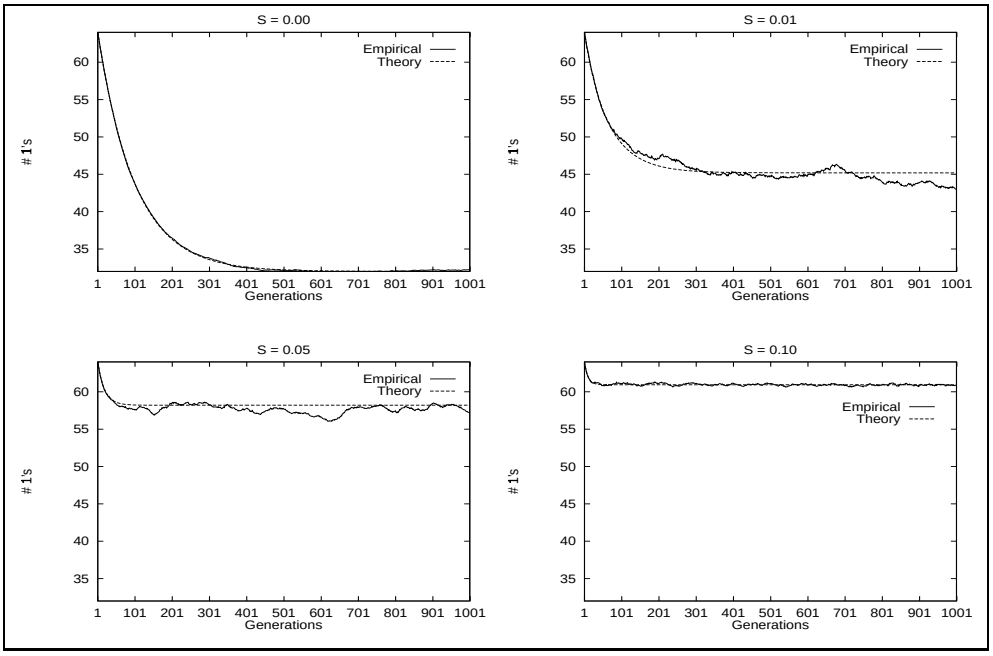

Figure 1: Selection and mutation modeled when $\mu = .005$. S takes on values 0.00, 0.01, 0.05, and 0.10.

As stated above, for some problems (fitness functions) only the number of α's in an individual matters. At first blush, this class of problems would appear to be limited to only unimodal functions (functions with one "peak" in the space). This is not true. For example, consider a familiar two-peak problem where individuals in an EA are binary strings. Traditionally one peak is at "111...111" while the other is at "000...000". However, this problem really monitors only the number of 1's. Suppose there are N 1's. If $N > L/2$ the fitness is N, else the fitness is $L - N$. There are two peaks with maximum fitness L. Many of the deceptive and trap functions investigated by Deb and Goldberg [1] fall into this class. In fact, it is also possible to create problems with an even higher number of peaks. For example, a problem might have a peak where $N = 0$, $N = L/2$, and $N = L$, thus creating a three-peak problem that depends only on one allele. Clearly this technique can be extended to more than three peaks.

Of the unimodal functions mentioned in the literature, two are especially of interest. The first is the class of "Royal Road" functions analyzed in Nimwegen et al. [7]. The Royal Road fitness function considers each individual to consist of N contiguous blocks of K bits, and the fitness of an individual is simply the number of blocks that consist of K 1's. This is analogous to having an alphabet where $C = 2^K$ in our model, where α is the allele corresponding to the binary string equivalent of K 1's. What is nice about this observation is that it appears as if many of the analytical techniques investigated in Nimwegen et al. [7] for the Royal Road function could be directly applied to the more general class of problems defined here.

The second function is from a widely studied problem in the biological community [6]. For this function the fitness of an individual is $(1 - S)^M$, where M is the distance of the individual from some optimum and S is a selection pressure ($0 < S < 1$). Thus the optimal individual has fitness 1.0 ($M = 0$), whereas non-optimal individuals have positive fitness less than 1.0.

Since the latter problem was especially designed to investigate the effects of mutation and selection on evolution, we use it as a test function for our mathematical framework. The mathematical model (consisting of Equations 2 – 4) was compared to the behavior of a standard EA (with recombination turned off) on the $(1 - S)^M$ function. Binary strings of length 64 were initialized to all 1's (which is the optimum string), and then the system was allowed to evolve. The population size of the EA was 1000. The average number of 1's in the strings was monitored for 1000 generations. Since binary strings are used the distance metric is Hamming distance. Figure 1 graphs the results when the mutation rate μ is 0.005 as the selection pressure S ranges from 0.00 to 0.10. Figure 2 graphs the results when the mutation rate μ is 0.01 as the selection pressure S ranges from 0.00 to 0.10. Both the theoretical and empirical curves are plotted. Although there is some noise due to stochastic fluctuations in the EA (the EA was only run once per problem), the agreement between the model and the experiments is quite good.

When $S > 0$, selection is active. Two observations are immediately obvious from the graphs. The first is that higher selection pressure changes the equilibrium distribution of the population, producing strings with more 1's. The second is that higher mutation rates also change the equilibrium distribution, producing strings with less 1's. Both of these results

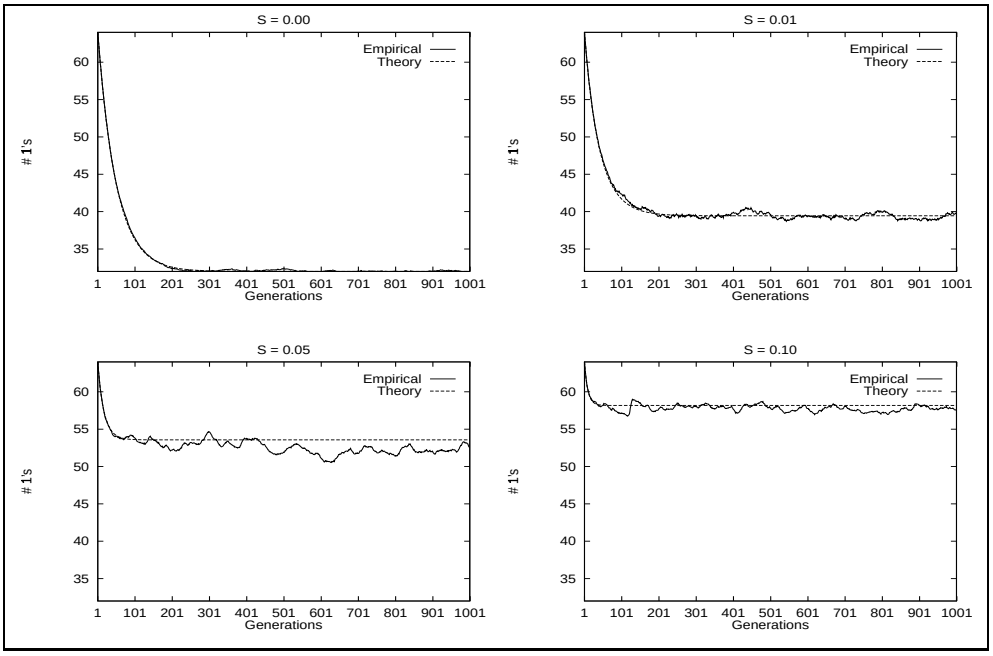

Figure 2: Selection and mutation modeled when $\mu = .01$. S takes on values 0.00, 0.01, 0.05, and 0.10.

are intuitively reasonable.

When $S = 0$, selection is not active. In this case the system is always driven to the same equilibrium distribution (50/50 1's and 0's), but higher mutation rates drive it there more quickly. The astute reader will note that the $S = 0$ graphs look suspiciously like the graphs for radioactive decay in physics. Interestingly, this is a correct observation! Although it is beyond the scope of this paper, it can be proven that the expected proportion of 1's, ${p_1}^{(t)}$, in the individuals is governed by:

$$ {p_1}^{(t)} = \frac{e^{-2\mu t} + 1}{2} $$

Thus -2μ plays a role similar to that of the decay rate in radioactive decay. The interested reader is urged to see Spears [11] for more details.

3 Aggregating Models of EAs with Selection, Mutation, and Recombination

The previous section described a model of mutation and selection in which equations of motion give the expected proportion of the strings in the population over time. In the worst case this involves C^L simultaneous equations. However, we have defined a class of fitness functions that allows for an error-free aggregation of the model that results in far fewer equations. If a fitness function depends only on one particular allele in the alphabet, only $L + 1$ equations are required.

The main advantage of this model and class of problems is that the expected behavior of reasonably large problems (i.e., problems in high-dimensionality spaces) can be theoretically evaluated with a small number of simultaneous equations. Another advantage is that the model can be easily extended to cover other forms of selection (such as linear-ranking selection), while still allowing aggregation to occur.

The lack of recombination in the above model need not be a drawback, since in fact some varieties of evolutionary algorithms, such as evolutionary programming [3, 4], do not use recombination.[3] However, if recombination is required, the equations of motion become more complex:

$$ p_{S_j}^{(t+1)} = \sum_{S_h} \sum_{S_l} \sum_{S_i} {p_{S_h}}^{(t)} \, {p_{S_l}}^{(t)} \times \frac{f(S_h)}{\overline{f}^{(t)}} \, \frac{f(S_l)}{\overline{f}^{(t)}} \, p_{(S_h \times S_l), S_i} \, p_{S_i, S_j} \tag{5} $$

Equation 5 first considers the proportions (${p_{S_h}}^{(t)}$ and ${p_{S_l}}^{(t)}$) of all pairs of strings S_h and S_l at time t. These proportions are modified by fitness-proportional selection, yielding the probability that S_h and S_l will be chosen for recombination. Recombination of S_h and S_l will produce an individual S_i, which can be mutated into string S_j. The term $p_{(S_h \times S_l), S_i}$ is the probability that the two strings S_h and S_l will be recombined to create S_i, while p_{S_i, S_j} computes the probability of mutating S_i into S_j. The third summation considers all possible strings S_i. The result is the expected proportion of string S_j at time $t+1$. Once again, the total system is described by C^L equations, one for each string S_j. Starting

[3] Although a multiparent recombination operator was proposed in [4], limitations of computer hardware in the 1960's did not allow for the implementation of the operator.

with initial proportions ${p_{S_j}}^{(0)}$, the C^L equations are iterated repeatedly to produce the expected time evolution of the system. The probability of recombination, $p_{(S_h \times S_l),S_i}$, can be derived using the "static schema analyses" in Spears [11], for a variety of recombination operators.

Unfortunately, the addition of recombination into the model dramatically worsens the complexity of each of the C^L equations. A similar difficulty can be seen in Whitley's "executable model" [13], which is similar in spirit, although it lacks mutation. Also, the addition of recombination makes aggregation much more difficult. For example, since recombination is affected by the location of alleles on an individual, monitoring the number of some allele (as was done above to aggregate the system with selection and mutation) will be problematic.[4]

A drawback of the above models is the assumption of an infinite-size population, which does not capture finite-population effects. As would be expected, an assumption of a finite population complicates the model even further, making aggregation even more necessary (and difficult). However, since finite-population EAs (with or without recombination) can be modeled as Markov chains e.g., see [8], an alternative approach is to find general methods for automatically aggregating Markov chains.

4 Aggregating Markov Chains

Many discrete systems can be described by a Markov chain model in which each state of the Markov model is some discrete state of the dynamical system. If there are N states, the Markov chain model is defined by an $N \times N$ matrix Q called the "one-step probability transition matrix," where $Q(i,j) \equiv p_{i,j}$ is the probability of going from state i to state j in one step. The t-step (transient) behavior of the system is described by the t'th power of Q, Q^t. The entries of Q^t are denoted as $p_{i,j}^{(t)} \equiv Q^t(i,j)$.

Nix and Vose [8] define a Markov chain model of a finite-population EA with selection, mutation, and recombination. Each state of the Markov model is a particular population of the EA. As would be expected, the number of states grows enormously as the population size (or string length) increases. Thus any techniques that are available for automatically aggregating Markov chains can immediately be used to aggregate the Nix and Vose model. We summarize the method of Spears [12] in this section.

Consider an example Q matrix obtained from a Markov chain of three states, where the first state is "VA", the second state is "WVA" and the third state is "NC":

$$Q = \left[\begin{array}{c|ccc} & \text{VA} & \text{WVA} & \text{NC} \\ \hline \text{VA} & p_{1,1} & p_{1,2} & p_{1,3} \\ \text{WVA} & p_{2,1} & p_{2,2} & p_{2,3} \\ \text{NC} & p_{3,1} & p_{3,2} & p_{3,3} \end{array}\right] =$$

$$\left[\begin{array}{c|ccc} & \text{VA} & \text{WVA} & \text{NC} \\ \hline \text{VA} & 0.7 & 0.1 & 0.2 \\ \text{WVA} & 0.4 & 0.2 & 0.4 \\ \text{NC} & 0.1 & 0.3 & 0.6 \end{array}\right]$$

The entries in the Q matrix, $p_{i,j}$, represent the probability that the system will transition to state j in one time step, given that it is currently in state i. Thus, if a time step is equivalent to one day, the probability of being in NC tomorrow, given that the system is in WVA today, is 0.4.

Aggregation of Markov chains is performed by continuously aggregating pairs of states into new individual combined states. Thus, the two issues are how to choose good pairs to aggregate, and how to aggregate each pair. We focus on the latter issue first, by continuing with the above example.

4.1 Aggregating Pairs of States

Suppose WVA and NC have been chosen for aggregation, creating a new combined state that represents being in either WVA or NC. Since this is a disjunctive situation, an appropriate label for the new combined state is "WVA $\vee$ NC". The probability of transitioning from VA *into* the new aggregated state is simply the sum $p_{12} + p_{13} = 0.1 + 0.2 = 0.3$. In general, if states i and j are aggregated, the new aggregated state is referred to as state $\{i \vee j\}$. The probability of transitioning from state k *into* the aggregated state is simply the sum $p_{k,\{i \vee j\}} = p_{k,i} + p_{k,j}$. Stated another way, part of the aggregation algorithm is to sum columns of probability numbers.

However, transitions *from* an aggregated state are more complicated to compute. For example, what is the probability of transitioning from state WVA $\vee$ NC into VA? In general, the probability of transitioning from the aggregated state to some other state ($p_{\{i \vee j\},k}$) must lie somewhere between $p_{i,k}$ and $p_{j,k}$, depending on how much time is spent in states i and j. Thus a weighted average of row entries in Q is called for, where the weights reflect the amount of time spent in states i and j. Unfortunately, an exact determination of these times is impossible to compute in advance. However, it turns out that we can use information from Q to estimate these times. Define m_i to be the sum of the probability numbers in the ith column of Q (we refer to this as the *column mass* of state i). Spears [12] has shown that m_i and m_j provide good estimates of the relative time spent in states i and j.

Aggregation of two states can now be expressed simply as follows. Let S denote the set of all N states, and let the non-empty sets $S_1, ..., S_{N-1}$ partition S such that one S_i contains the two chosen states, while each other S_i is composed of exactly one state. Let m_i denote the column mass of state i. Then the aggregated matrix Q' is:

$$Q'(x,y) = \frac{1}{\sum_{i \in S_x} m_i} \sum_{i \in S_x} \left[m_i \sum_{j \in S_y} p_{i,j} \right] \qquad (6)$$

This corresponds to taking a weighted average of the two rows corresponding to the two chosen states, while summing

[4] However, Annie Wu (personal communication) has shown that it may be possible to estimate $p_{(S_h \times S_l),S_i}$ for the "one allele" equivalence class, where the two parents have h and l α's and the offspring has i α's.

Repeat until no new aggregated states are created
(a) For each state i in the J set of the current aggregated model
(i) Find the most similar state j in the J set.
(ii) If $Similarity_{i,j} < \epsilon$, aggregate states i and j with Equation 6.
(b) For each state i in the non-J set of the current aggregated model
(i) Find the most similar state j in the non-J set.
(ii) If $Similarity_{i,j} < \epsilon$, aggregate states i and j with Equation 6.

Figure 3: The aggregation algorithm.

the two corresponding columns. The other entries in the Q matrix remain unchanged. It is important to note that Q' is itself a Markov chain and hence can directly replace Q in numerical computations.

4.2 Choosing Pairs of States to Aggregate

In general, aggregating an arbitrary pair of states in a Markov chain may not lead to good aggregation. How then are a pair of states chosen for aggregation? First it is necessary to define what is meant by perfect (error-free) aggregation. Then it is necessary to estimate the amount of error that is introduced by aggregating a given pair of states. Pairs of states are chosen for aggregation if they yield little estimated error.

As mentioned above, analysis of t-step transition probabilities (i.e., *transient* behavior of the Markov chain) can be realized by computing Q^t. For large Q matrices this is computationally expensive, and thus the goal is to aggregate Q and to then raise Q' to the t'th power. If the aggregation algorithm has worked well then the t'th power of the aggregated matrix Q' should be (nearly) identical to aggregating the t'th power of the unaggregated matrix Q. Perfect aggregation has occurred if $(Q^t)' = (Q')^t$.

Given the definition of perfect aggregation, it is possible to estimate how far the aggregation of any pair of states i and j will deviate from perfection. Let $Error_{i,j}$ be the estimated error introduced into Q' by aggregating states i and j. It can be shown that $Error_{i,j}$ is:

$$Error_{i,j} = (\sum_x |\beta_{i,j}(x)|)(\sum_y |\alpha_{i,j}(y)|) \equiv Similarity_{i,j}$$

where $\alpha_{i,j}(y) = p_{i,y} - p_{j,y}$ and $\beta_{i,j}(x) = (m_i p_{x,j} - m_j p_{x,i})/(m_i + m_j)$. The explanation and derivation of this estimation is beyond the scope of this paper, although it can easily be observed that $\alpha_{i,j}$ focuses on the ith and jth rows of Q, while $\beta_{i,j}$ focuses on the ith and jth columns. Pairs of states are *similar* if they will produce little error. Details can be found in Spears [12].

4.3 Aggregating Markov Chains of EAs

The algorithm can now be applied to the Nix and Vose [8] Markov model. It is possible to examine the accuracy of the aggregated Markov chains by using both Q^t and $(Q')^t$ to compute the probability distribution $p^{(t)}$ over the states at time t. To perform these computations, Q^t must be combined with a set of initial conditions concerning the EA at generation 0. The *a priori* probability of an EA being in state i at time 0 is:

$$p_i{}^{(0)} = \frac{P!}{z_{i,0}!...z_{i,r-1}!}\left[\frac{1}{r}\right]^P \quad (7)$$

where the population size is denoted as P and $r = C^L$ is the total number of different individuals. The notation $z_{i,y}$ is the number of occurrences of string y in the ith population. The probability that the EA will be in a particular state j at time t is $p_j{}^{(t)} = \sum_i p_i{}^{(0)}\ p_{i,j}{}^{(t)}$. The probability that the EA will be in a set of states J at time t is $p_J{}^{(t)} = \sum_{j \in J} p_j{}^{(t)}$.

Let J represent the set of all states that contain at least one copy of the optimum (i.e., the set of all populations that have at least one individual with the optimum function value). The Markov model is used to compute $p_J{}^{(t)}$, the probability of having at least one copy of the optimum in the population at time t. The aggregation algorithm can thus be evaluated by using both Q^t (ground truth) and $(Q')^t$ (the estimate) to compute $p_J{}^{(t)}$ for different values of t. The closer the estimate is to ground truth, the better the aggregation algorithm is working.

Since the goal is to compute probabilities involving states containing the optimum (the J set), J states should not be aggregated with non-J states. Consequently, the aggregation algorithm is run separately for both sets of states. The algorithm is shown in Figure 3. Pairs of states are aggregated if their estimated error (similarity) is less than the threshold ϵ.

Previous aggregation results [12] have been with matrices with less than $N = 1000$ states. We now show novel results with a matrix of $N = 2024$ states. The EA fitness function used for this experiment is a "Type 1" deceptive function, and $P = 21$.[5]

The setting of ϵ is crucial to the success of the experiment. As in Spears [12] a value of 0.15 yields good aggregation. This can be seen be examining Figure 4. The values $p_J{}^{(t)}$ are computed for t ranging from 1 to 100 generations, for both the aggregated and unaggregated Markov chains, and graphed as

[5] See De Jong et al. [2] for a definition of this search space.

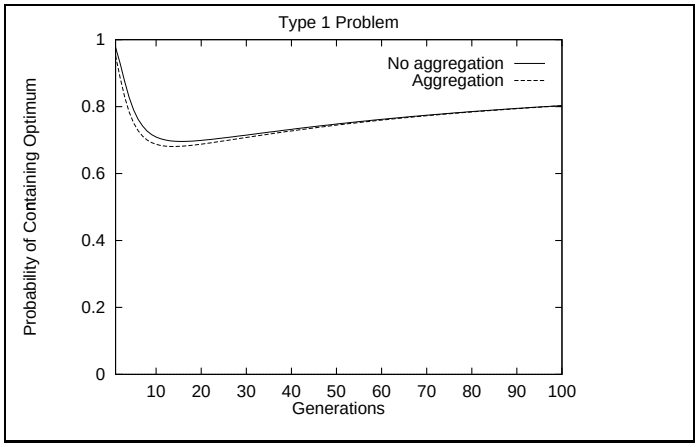

Figure 4: ${p_J}^{(t)}$ where ϵ is 0.0 (no aggregation) and 0.15 (aggregation) for $N = 2024$. The function is Type 1 deceptive.

curves. Despite the fact that 1917 of the 2024 states were removed (about 95% of the states), the aggregated results are very close to ground truth.

5 Discussion

The standard difficulty in modeling EAs is in finding the correct level of granularity. If the level of granularity is too coarse the model is easy to analyze but lacks predictive ability. On the other hand, if the level of granularity is very fine the model will be predictive but is often computationally intractable.

A solution to this difficulty is to find methods for aggregating fine-granularity models of EAs – the aggregation removes unnecessary detail from the models, producing simpler models with good predictive accuracy. This paper has outlined two different methods for aggregation. The first method is effective with "equation of motion" models of EAs that have selection and mutation. Systems of C^L simultaneous equations can be automatically aggregated to only $L+1$ equations, for a particular class of interesting fitness functions. No error is introduced by the aggregation.

The second method relies on the fact that EAs (with or without recombination) can be modeled as Markov chains. The paper summarizes an algorithm for automatically aggregating Markov chains [12] and illustrates its effectiveness on a Markov chain model of an EA with 2024 states. The aggregation is not error free, although the error produced is quite small. The advantage of this algorithm is that it will work with arbitrary classes of fitness functions.

Despite the preliminary success of these methods, there is enormous room for improvement. Future work will focus on not only improving the existing aggregation methods, but on identifying other mechanisms and opportunities for further aggregation.

Acknowledgements

I thank Diana Gordon for pointing out that a method for evaluating the Markov chain aggregation algorithm was to show that $(Q^t)' = (Q')^t$. Diana also pointed out portions of the theory that needed mathematical refinement.

Bibliography

[1] Deb, K. and Goldberg, D. (1992) "Analyzing deception in trap functions". Foundations of Genetic Algorithms, v2, 93–108.

[2] De Jong, K., Spears, W. and Gordon D. (1994) "Using Markov chains to analyze GAFOs". Foundations of Genetic Algorithms, v3, 115–137.

[3] Fogel, D. (1995) "Evolutionary computation". IEEE Press.

[4] Fogel, L., Owens, A., and Walsh, M. (1966) "Artificial intelligence through simulated evolution". John Wiley.

[5] Holland, J. (1975) "Adaptation in natural and artificial systems". University of Michigan Press.

[6] Muller, H. (1964) "The relation of recombination to mutational advantage". Mutat. Res., v1, 2–9.

[7] van Nimwegen, E., Crutchfield, J. and Mitchell, M. (1997) "Finite populations induce metastability in evolutionary search". Physics Letters A.

[8] Nix, A. and Vose, M. (1992) "Modelling genetic algorithms with Markov chains". Annals of Mathematics and Artificial Intelligence. #5, 79–88.

[9] Prügel-Bennett, A. and Shapiro, J. (1996) "The dynamics of a genetic algorithm for simple ising systems". Physica D.

[10] Rattray, L. and Shapiro, J. (1996) “The dynamics of a genetic algorithm for a simple learning problem”. Physica A.

[11] Spears, W. (1998) “The role of mutation and recombination in evolutionary algorithms”. Ph.D. Dissertation, George Mason University, Fairfax, VA.

[12] Spears, W. (1998) “A compression algorithm for probability transition matrices”. SIAM Matrix Analysis and Applications, v20, #1, 60–77.

[13] Whitley, D. (1992) “An executable model of a simple genetic algorithm”. Foundations of Genetic Algorithms, v2, 45–62.