GENETIC ALGORITHMS FOR PARTITIONING SETS

WILLIAM A. GREENE
Computer Science Department
University of New Orleans
New Orleans, LA 70148
USA
bill@cs.uno.edu

Received June 26, 2000
Revised November 20, 2000

Abstract: We first revisit a problem from the literature, that of partitioning a given set of numbers into subsets such that their sums are as nearly equal as possible. We devise a new genetic algorithm, Eager Breeder, for this problem. The algorithm is distinctive in its novel and aggressive way of extracting parental genetic material when forming a child partition, and its results are a substantial improvement upon prior results from the literature. Then we extend our algorithm to the more general setting, of partitioning a set in the case that the environment provides us a measure of the fitness of individual subsets in the partition. We apply the extension to two artificial problems, one with a targeted partition whose subsets are of very diverse sizes, and one whose subsets are the same size. Finally, we apply our approach to several map coloring problems, and obtain good results there as well. In our different stages of work, we exploit different heuristics, which are attuned to the particular partitioning problem under attack.

Keywords: genetic algorithm, set partitioning, Equal Piles Problem, map coloring.

1. Introduction

The research of this paper begins by revisiting a problem from the literature of genetic algorithms (GAs), namely, partitioning 34 particular integers into ten subsets such that the sums of the subsets are as nearly equal as possible. To solve this problem, we devise a new genetic algorithm for partitioning sets; it follows an aggressive approach for forming a child's genetic material out of that of the parental partitions; we name the algorithm Eager Breeder. Our results improve upon those in the literature. For this original problem, it is noteworthy that the goodness (fitness or, complementarily, error) of not only a partition but also each subset in it, is known. Next, we extend our approach to the more general setting where the problem is to partition sets, and the environment provides us with a measure of the fitness or error of each subset in a partition. We get good results on two artificial problems, where the targeted subsets are of very diverse sizes, and where the targeted subsets are of identical sizes. Finally, we extend our approach to several map coloring problems, and get good results there as well.

Genetic algorithms are a problem solving paradigm which apply such Darwinian evolutionary forces as survival of the fittest, mutation, and mating with crossover of genetic material, to arrive at desirable solutions for the problem at hand. As a first step, one must devise a way to represent a candidate solution in a form that resembles a chromosome, that is, as a sequence of values (often bits) that describe or characterize the candidate. Then mating with crossover of genetic material takes the form of selecting one or more points along the chromosome, cutting both parents there, and interchanging genetic material from the two parents to form a child or two. John Holland[1] (1975) is credited with inventing the area of genetic algorithms. According to his Schema Theorem, we can expect a population of individuals to evolve towards containing good solution candidates.

Our view is that genetic algorithms are fundamentally a heuristic approach. Holland's proof shows that, courtesy of the stochasticism employed, we can expect to chance upon better and better candidate solutions. An aspect we will encounter several times in this paper is that incorporating additional heuristicism which is appropriate to the particular problem at hand can help a genetic algorithm find good solutions more quickly.

2. The Equal Piles Problem

The Equal Piles Problem was defined and first studied by Jones and Beltramo[2] (1991). The problem is to partition N numbers into K subsets, such that the sums of the subsets are as nearly equal as possible. (Jones and Beltramo relate the problem to the case of objects of specified heights, which are to be stacked into disjoint piles so that the pile heights are as nearly equal as possible.) Already interesting in the abstract, such problems have practical applications as well. One instance is load balancing: given N tasks of known durations, how can they be assigned to K processors so that the work is evenly distributed?

The particular instance of the problem which Jones and Beltramo studied had 34 specific integers which were to be partitioned into ten subsets. Those 34 numbers are the following.

1.	3380	10.	648	19.	3519	27.	2787
2.	1824	11.	2588	20.	1363	28.	4635
3.	1481	12.	3380	21.	1824	29.	4068
4.	2060	13.	1952	22.	3305	30.	2992
5.	1225	14.	3832	23.	2156	31.	5932
6.	836	15.	3176	24.	3305	32.	528
7.	1363	16.	2316	25.	3049	33.	3304
8.	2705	17.	2479	26.	3980	34.	4107
9.	4635	18.	3433				

Jones and Beltramo say little about their choice of numbers, except that they want their problem instance to be challenging, and to have an optimal solution which is unique. The authors acknowledge that uniqueness must give a little ground to the fact that some of the numbers are identical, for instance, N[7] = N[20], where N[*k*] means the *k*-th number. Fewer optima means fewer points in the fitness landscape where candidate solution subpopulations can accumulate.

Now that the particular 34 numbers are known to us, we can compute the ideal subset sum, by adding these numbers and dividing by ten, the number of subsets; the result is 10,000. Knowing the ideal subset sum means that each subset in a partition can be assigned a measure of error, namely, the (absolute) difference between its actual sum and the ideal subset sum. Subset errors can be combined to give an error measure for the partition. The fitness of a subset or a partition should naturally be some value that is complementary to its error.

For the instance of the problem studied by Jones and Beltramo, there is an optimal solution for which each subset sums to 10,000 exactly. In fact, as pointed out by Falkenauer[3] (1995), and unwished for by Jones and Beltramo, there are several distinct optimal solutions, since, for instance, N[19] = N[20] + N[23].

Jones and Beltramo tried nine GAs. In their most successful effort, a partition was represented by using a permutation of the 34 numbers, and crossover was PMX crossover (Goldberg,[4] 1989), which is attuned to the crossover of permutations. This GA, on one occasion (meaning on one trial of many generations) found a best sub-optimal solution (two subset sums are off by 1), but on average the best partitions it could find had an error of 171 (in their development, the error of a partition equals the sum of the errors of its subsets). Their best GA never found an optimal solution.

Falkenauer[3] (1995) tackled this problem again. His approach has some clear differences with that of Jones and Beltramo. In particular, Falkenauer argues that for a "grouping" problem such as this one, a chromosome's genes should be taken to be the subsets of numbers, not the individual 34 numbers themselves. Thus, when parents mate, they begin by exchanging entire subsets; they do not exchange individual numbers *per se*. Additionally, Falkenauer argues that subsets in a partition have no natural order, so there is no need to arrange them as a sequence; a partition can be treated as a set of subsets versus a sequence of subsets. Falkenauer runs his Grouping Genetic Algorithm (GGA) on this problem, and gets distinctly better results than found by Jones and Beltramo. In 30 trials, each of up to 3500 generations, Falkenauer's GGA finds an optimal solution on 26 of the 30 trials, after an average of 17,784 (not necessarily distinct) partitions have been encountered. Then, after making his crossover operator greedier, he improves upon himself: his Greedy GGA finds an optimal partition on every one of 30 trials, after an average of 9,608 partitions have been encountered.

3. The Eager Breeder Algorithm

Our solution to the Equal Piles Problem has similarities with the approach of Falkenauer, but is different from it in distinct ways, most notably in how a child's genetic material is culled from that of the two parents. Also, the mutation we practice is quite distinct from that of Falkenauer's solution.

We represent a subset of the 34 numbers as an array of 34 boolean values, with the obvious interpretation that a value of *true* is at location *k* in the array if and only if the *k*-th number belongs to this subset. This representation is more space-costly than others but leads to time-economies, such as when we want to know if the *k*-th of the 34 numbers belongs to a given subset. For us, the *error of a subset* is the absolute difference between the sum of the numbers in this subset versus the ideal subset sum of 10,000. A *partition* is a list of 10 subsets. Mathematically, it makes no difference in what order we list the subsets, but for our approach it is advantageous to list them in increasing order of error, so that better subsets appear first in a partition's list of subsets. The *error of a partition* we take to be the Euclidean norm (square root of the sum of the squares) of the vector of errors of its ten subsets. There will be a population of partitions, which our genetic algorithm will subject to evolutionary forces. The fitness of an individual partition is a value complementary to its error, and is defined as follows. In the population, error values range from some LOW to some HIGH value. We define the fitness of an individual partition to be HIGH + 1 - (own error). Fitness values then range from 1 to HIGH - LOW + 1. In our approach it is advantageous to list population members in decreasing order of fitness, so that fitter individuals appear first.

Our population will evolve through a sequence of generational changes: the population P(t) at time t will evolve to the next generation P($t+1$) at time $t+1$. The Darwinian evolutionary forces which we apply are survival of the fittest, mating with crossover, and mutation. The ensuing descriptions are for our most successful efforts.

3.1. *Survival of the fittest*

Survival of the fittest surfaces in two forms. Firstly, a small number (seven percent) of the most fit individuals automatically survive into the next generation. This practice is termed *elitism*, and ensures that generation P($t+1$) contains some individuals at least as good as the best seen in the preceding generation P(t). (In point of fact, in our approach these elite survivors stand a low risk of mutation, as detailed later.) Secondly, individuals are selected for parenting with a probability equal to their relative fitness in the population; we use the standard weighted roulette wheel; see Goldberg[4]. Thus, fitter individuals are parents more frequently than less fit individuals.

3.2. *Crossover*

Mating with crossover produces one child in our approach, and culls genetic material from the parents in a novel and aggressive way, when forming a child. For this reason we name our approach Eager Breeder. Mating is a two-step process: the collecting of subsets for the child, followed by the repair of those subsets. As for collecting subsets to pass along to the child, first note that each parent has 10 subsets that comprise its partition, making 20 parental subsets altogether. We collect the 10 best distinct subsets found among the 20 parental ones. These start as the child's subsets, and are repaired in the second step.

Let us take a moment to detail the collection algorithm. Subsets in a parent are listed in order of increasing error, so more accurate subsets (ones whose sum is nearer to the ideal subset sum of 10,000) appear first. So, start two pointers, one at the beginning of each parental subset list, and let them track down their respective lists. At each step, the better subset of the pair now pointed at is copied into the child's collection (for ties, flip a coin). Figure 1 suggests this approach.

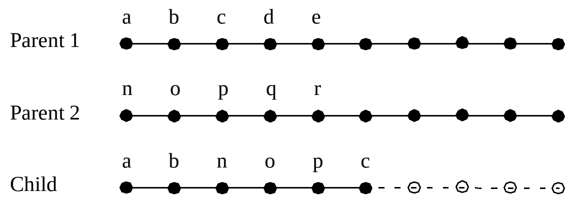

Figure 1. Only the best parental subsets enter the child.

The 10 subsets collected for the child now go through a repair phase. These 10 subsets do not necessarily form a partition of the 34 original numbers. Some subsets may overlap; some of the 34 numbers may not appear in any child subset. But we observe that for each of the 34 numbers, it belongs to just one subset from the first parent, and likewise belongs to just one subset from the second parent, because each parent is a partition. So, for each of the 34 numbers, it belongs to either 2 or 1 or 0 of the child subsets which have been collected. Repair the child subsets as follows. If a number belongs to 2 child subsets, then remove it from the more erroneous one (flip a coin in the case that the two subsets have identical error). If one of the 34 numbers belongs to just 1 child subset, then that is as desired for a partition and no adjustment is made in this case. Now round up all those 34 numbers which so far appear in no child subset at all. We want to distribute these as yet unassigned numbers into the child subsets, and will use heuristicism to do so. Namely, put that as yet unassigned number which is currently biggest into that child subset which currently has the lowest subset sum. We term this the Most Into Least (MIL) heuristic. Falkenauer employs this same insight and terms it the Equal Piles Descending heuristic. This is the same heuristic approach one would take if presented with a given box and several smaller boxes to pack inside it. One would first place the biggest smaller box into the given box, and then try to pack the others around it. Applied to the problem at hand, the intuition is that we will obtain subsets whose sums are somewhat equalized.

Of course, the crossover we have just described certainly departs from biology and from standard genetic algorithms. In the biosphere, some species, such as ourselves *homo* *sapiens*, are *diploid*: each of our chromosomes is a double helix; for mating, each chromosome separates into its two strands, one of which will pair up with the corresponding strand from the other parent and reform a double helix. The remaining species are *haploid*: a chromosome is one strand, which undergoes crossover at a cutpoint in the strand, one fragment of which rejoins the complementary fragment from the other parent. Holland's original approach for a genetic algorithm models haploid species. To our knowledge, no biological species culls genetic material for the child in a manner similar to our Eager Breeder algorithm. Moreover, note that the approach would allow three or more parents to mate and produce one child among them.

3.3. *Generational change*

To manufacture the next generation P($t+1$) of our population, out of generation P(t), we do the following. As said earlier, a small percentage (seven percent) of the fittest individuals in P(t) automatically survive into P($t+1$). Then, we build P($t+1$) up to the same size as P(t) by mating with crossover. Individual partitions in P(t) are selected for parenting, using a weighted roulette wheel to select them with a probability equal to their relative fitness; a pair of parents mate and the resulting child enters P($t+1$). Once P($t+1$) is up to the same size as P(t), we sort its partitions into decreasing order of fitness (so fitter individuals come first), subject them to degrees of mutation based upon their fitnesses, and finally after all mutation steps, we again sort the population by fitness. These steps complete the construction of P($t+1$).

3.4. *Mutation*

Once P($t+1$) is filled up to the same size as P(t), and has been sorted into decreasing order of fitness as described in the preceding paragraph, its elements are subjected to mutation. Mutation of P($t+1$) is graduated and stochastic. The fitter an individual is, the fewer mutation steps it undergoes, and each mutation occurs only after a specified probability is met.

One stochastic mutation step is realized as follows. If a specified probability (say, 50%) is met, then at random one of the 34 numbers is selected, removed from the subset it is in, and inserted into a subset which is chosen at random (the number may be returned to the same subset it was just in).

To graduate the mutation of P($t+1$), we divide it into four bands. The following scheme was found to be effective. The first band has the same size as the set of the elite survivors and consists of the fittest seven percent of the population. Each element in the first band undergoes one mutation step with the low probability of ten percent. The remaining three bands consist of the next 33, 30 and 30 percent of the population, in decreasing fitness. With 50% probability, an element in these bands undergoes, respectively, 4, 10, and 20 mutations. Thus, we expect an element in the last band to undergo 10 mutations on average. In general, generous mutation on the population's least fit individuals was found to work well. Figure 2 illustrates the approach taken with mutation.

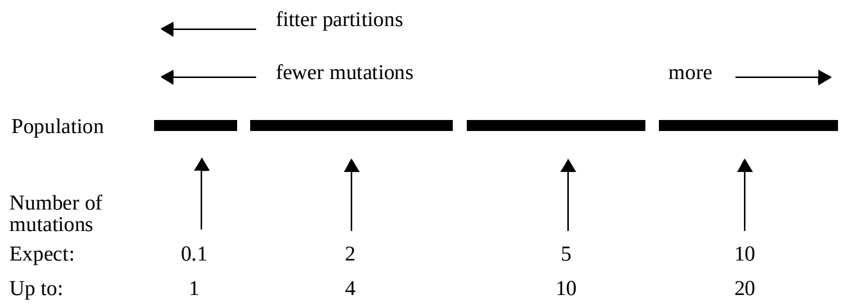

Figure 2. Mutation is graduated and stochastic.

3.5. *Results*

Now we describe our results. As usual, tuning of the various parameters, such as mutation rates and population size, produce different results. Here we will describe our best results. Population size was 250, and the mutation rates were those just detailed in the preceding paragraph. The number of trials was 30, the same as for Falkenauer, and also Jones and Beltramo, and we allowed each trial to run to a maximum of 40 generations. An optimal partition was found on 29 of the 30 trials; on the remaining trial, a best sub-optimal partition was found. These results are almost as "perfect" as those of Falkenauer. For the 29 of our trials on which we found an optimal partition, the latter was found on average generation number 12.97, which implies that on average 3,242 individuals had to be encountered before finding an optimal partition. By contrast, Falkenauer's Greedy GGA algorithm had to encounter 9,608 partitions on average before finding an optimal partition. Thus, our approach has only 33.7% of the cost of Falkenauer, a marked improvement. Table 1 summarizes these results.

Table 1. Comparison of past and present research.

	Falkenauer's Greedy GGA	Eager Breeder
Population size	50	250
Max number generations	3500	40
Trials finding optimal partition	30	29
Average # generations to find optimal partition		12.97
Average # individuals to find optimal partition	9,608	3,242

In this paragraph, for contrast we synopsize the approach taken by Falkenauer, which approach is rather different from our own. His crossover operation randomly chooses some piles from the first parent, and adds them to the pile collection of the second parent. Any piles originating from the second parent which happen to overlap a pile added from the first parent get eliminated, their elements are collected, some of these are used to start enough new piles to have ten piles altogether, then remaining values are distributed following the principle to put the biggest as-yet-unassigned value into the subset which currently has the lowest sum. His greedy crossover is similar, except that the piles which come from the first parent are first shorn of their less accurate members. The mutation operator used by Falkenauer also differs markedly from ours. He empties a randomly chosen subset, joins to those values enough more out of remaining piles to make one-fourth of the totality of values (one-fourth of 34 would be eight or nine), then he uses some of those to start a new replacement pile, and finally distributes the remaining as-yet-unassigned values by the now familiar principle.

4. The Extension

A distinctive feature of the Equal Piles Problem is that we can assess not just the goodness (fitness or, complementarily, error) of a whole partition, but also the goodness of the individual subsets in the partition. After all, once we have seen the 34 numbers and know we are to partition them into ten subsets, then we can calculate the ideal subset sum. Then the goodness of a subset is determined by how close its own sum is to the ideal subset sum.

We wanted to extend our Eager Breeder approach to a more general setting. That setting is that we are to partition a set into subsets, and there is a fitness function on the scene which reveals the goodness of each subset in a partition.

For this work we invented two artificial partitioning problems. Intuition suggests that it is harder to partition a set into subsets when those subsets are of quite diverse sizes. Our first artificial problem has subsets of diverse sizes and our second problem has subsets all of the same size. Later we will describe the second problem. The first artificial problem was manufactured as follows. We will target one particular partition out of many. For it, there is 1 subset which contains 20 specific elements (they happen to be the integers 0 to 19, but of course that is immaterial), there is 1 subset containing 10 specific elements, and continuing, 2 subsets of size 5, 3 subsets of size 2, and finally 5 subsets of size 1. That makes up altogether 51 elements which are to be partitioned into 12 subsets.

Now we need a sensible notion of fitness or error, that can tell us the goodness of any subset of our 51 elements. Let such a subset be named *S* and suppose *x* is an element of *S*. We ask, how well does *S* co-associate *x* with the other 50 elements on the scene? Some elements are supposed to be included in the subset containing *x* and other elements are supposed to be excluded from the subset containing *x*. Are they? Calculate the *co-association rating of x with respect to S* as follows. For the 50 other elements, add 1 or 0 according as subset *S* correctly co-associates x with that element; the result is a number from 0 to 50. Now define the *fitness of S* to be the average of the co-association ratings of those *x* in *S*. And we can define the *error of S* to be 50 - fitness(*S*). Now both fitness and error measures are numbers in the range 0 to 50. (Of course, we could but do not normalize to the unit real interval [0, 1].)

We note that this definition of fitness is informed but not informative. By not informative, we mean that if we are told that a certain subset has a fitness value of, say, 35, that information provides no clue as to which elements we ought to add to or remove from the subset.

From this point we largely carry over our earlier approach. We represent a subset as an array of boolean values, and represent a partition as a list of subsets, arranged into increasing order of error so that fitter subsets appear first. As before we define the error of a partition to be the Euclidean norm (square root of the sum of the squares) of the vector of errors of the subsets in the partition.

Let us review the earlier crossover operation. First, enough child subsets are collected, by culling the best subsets found among the parents. Then repair was done on these subsets, in two steps. For each set element which appears in two child subsets, we remove it from the more erroneous one. Then the set elements which so far appear in no child subsets (the “no-show” elements) get distributed into the child subsets. We focus on this last step.

For the Equal Piles Problem, no-show elements got distributed into child subsets by using the Most Into Least (MIL) principle, namely, the currently largest number gets assigned to the subset with the currently least sum. This heuristic is appropriate to the Equal Piles Problem, whose goal is subsets with equal sums. The MIL principle cannot be applied to our current problem.

With the thought in mind that subsets with small error should not be tampered with, we replaced the MIL distribution principle with another one. Namely, one after another, randomly choose a no-show element and insert it into that subset which currently is the most erroneous. This distribution heuristic was only modestly successful. On a typical run of 30 trials, the targeted partition was discovered perhaps 7 times. On the other 23 or so trials, the best partition found tended to lump together the two largest subsets, of sizes 20 and 10 (this introduces an error of 13.33).

A heuristic for distributing the no-show elements which is quite successful is next described. We introduce some stochasticism. Just before distributing the no-show elements, we calculate a weighted roulette wheel, using the then errors of the child subsets. Our wish is to favor a more erroneous subset for reception of a no-show element, with probability equal to its relative error. To distribute the no-show elements, one after another, one is chosen at random and then assigned to a subset as determined by the weighted roulette wheel.

Using the distribution heuristic just described, our best results are as follows. Population size was set at 100; on each trial we allowed a maximum of 200 generations; we ran 30 trials. On all 30 of our trials, the targeted partition was discovered, on average generation number 48.7, which implies that on average 4,870 partitions had to be encountered before the targeted one surfaced. These are very good results!, when one considers that there are a stupendous number of ways to partition 51 elements into 12 subsets. Our computerized count of the number of different ways to partition 51 elements into 12 (non-empty) subsets gave the figure of 1.97E+46, which is a staggeringly large number. Only one of these is the targeted partition, and our algorithm discovered it after encountering only 4,870 partitions on average. Table 2 summarizes these results, as well as the results achieved on our second artificial problem.

At last we will describe our second artificial problem. For it, we wanted the subsets to be all of the same size. We manufactured a set of 48 elements, and also a targeted partition consisting of 8 subsets, all of the same size 6. Again population size was 100, and a maximum of 200 generations were allowed on each of 30 trials. The targeted partition was discovered on every trial, on average generation number 20.9. This implies that on average 2,090 partitions had to be encountered before the targeted one surfaced. Thus, our earlier intuition is corroborated, namely, that it seems inherently harder to discover an apt partition when that partition's subsets are of rather diverse sizes.

Table 2. Extension of Eager Breeder to two artificial problems.
In all trials, population size = 100, and the maximum number of generations = 200.

Subset sizes:	diverse (20,10,5,5,2 2,2,1,1,1,1,1)	same (6,6,6,6, 6,6,6,6)
Number of trials	30	30
Trials finding optimal partition	30	30
Average # generations to find optimal partition	48.7	20.9
Average # individuals to find optimal partition	4,870	2,090

5. Map Coloring Problems

Map coloring problems have interested mathematicians for a very long time. The Four Color Problem was a mathematical conjecture which was not resolved[5,6] until the 1970's. Given any planar map (that is, one drawn in the two-dimensional plane), are more than four colors ever needed to color such a map? Here, to color the map means that two regions that share a boundary line cannot be colored the same. (Two regions that meet at only one or several points, as illustrated by the states Arizona and Colorado, are permitted to be the same color.) It is now known that four colors suffice for any planar map.

Map coloring problems are set partitioning problems. We next apply our earlier approach to the problem of coloring the map of the 48 contiguous United States, using four colors. (Figure 3 shows the 48 states, labeled with their standard postal abbreviations.) Thus, we need to partition the set of 48 states into four subsets, for the four colors, subject to the constraint that any two states in the same subset do not share a boundary line.

We need some measure for the goodness of a subset of the 48 states. We choose to think in terms of error, and define the error of a subset to be the number of pairs of elements within it that do incorrectly share a boundary line.

Next we brought over *en masse* the machinery of the extended Eager Breeder algorithm that was described in section 4. Recall, there the heuristic for distributing no-show elements into child subsets used a weighted roulette wheel to favor the more erroneous

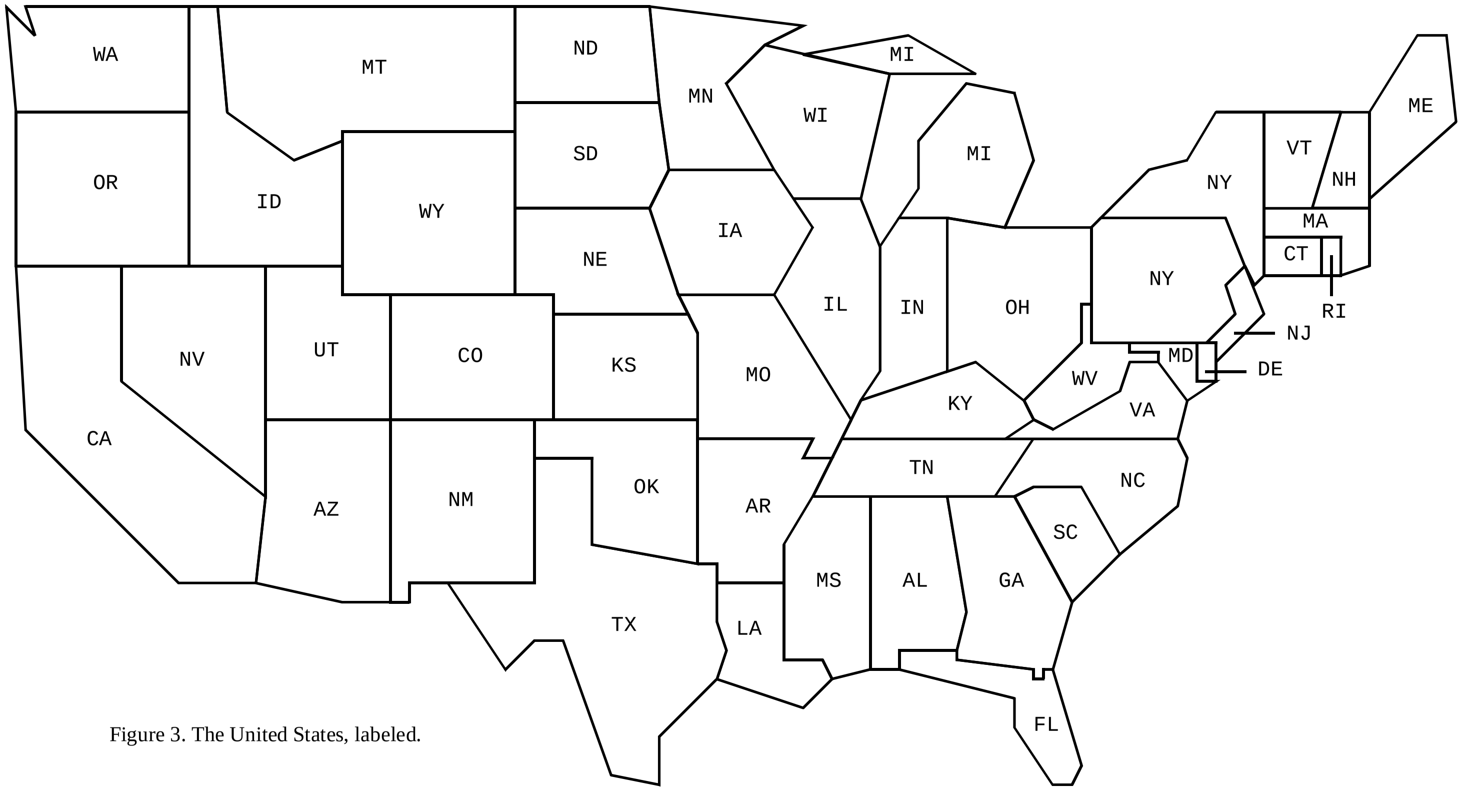

Figure 3. The United States, labeled.

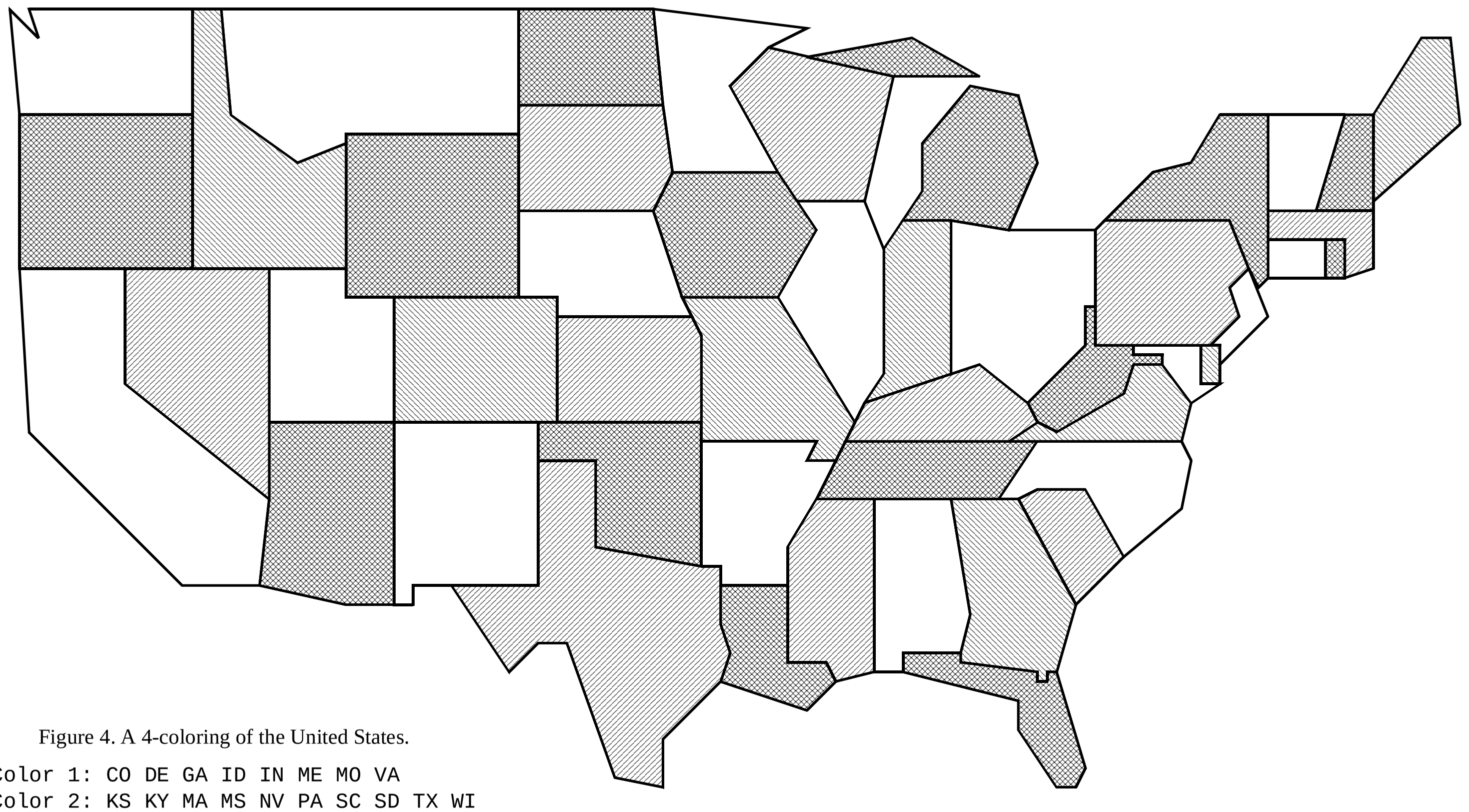

Figure 4. A 4-coloring of the United States.

```
Color 1: CO DE GA ID IN ME MO VA
Color 2: KS KY MA MS NV PA SC SD TX WI
Color 3: AZ FL IA LA MI NH NY ND OK OR RI TN WV WY
Color 4: AL AR CA CT IL MD MN MT NE NJ NM NC OH UT VT WA
```

subsets for receipt of a no-show element. When we ran this algorithm on the problem of four-coloring the United States, we found little success. Instead, as generations evolved, very many partitions arose which had three error-free subsets of just a few states, plus one large and very erroneous subset.

This led us to yet another heuristic for distributing no-show elements into child subsets. Since we have a measure of the goodness (for us, error) of a particular subset, we may ask, insertion of element *x* into subset *S* results in what change to the goodness of *S*? For distributing no-show elements into child subsets, the new heuristic we have adopted for our map coloring arena is the following. One after the other, a no-show element is chosen at random, and inserted into that subset for which insertion shows the most advantageous change in error.

This approach proved to be very successful. Our best results are described next. Population size was set at 20; on each trial we allowed a maximum of 15 generations; we ran 30 trials. On all 30 of the trials, a correct coloring was discovered, on average generation number 5.8, which implies that on average 116 partitions were encountered before chancing upon a correct one. Table 3 summarizes these results, and also the results when we reduced population size to a mere 10. Then a correct coloring was discovered on 25 of the 30 trials. A four-coloring of the United States is given in Figure 4.

Table 3. Four-coloring the United States.
In all trials, the maximum number of generations = 15.

Population size:	20	10
Number of trials	30	30
Trials finding correct coloring	30	25
Average # generations to find correct coloring on successful trials	5.8	8.6
Average # individuals to find correct coloring on successful trials	116	86

Jones and Beltramo also successfully attacked the problem of four-coloring the United States. For the problem of partitioning N objects into K subsets, they found their greatest success by representing a partition as a permutation of the N objects, which is then interpreted by a “greedy decoding”. The latter works by taking the first K elements of the permutation as initializations for the K subsets, then, one after another in order, adding remaining permuted elements into that subset which results in the best fitness. When this approach, plus PMX crossover, was applied to the map coloring problem, their results were as follows. On 30 trials, each having an initial population of 1000 random permutations, they report a correct coloring appearing already in the initial population! The greedy decoding of Jones and Beltramo has a similarity to our heuristic detailed above for distributing no-show elements in the map coloring arena. Their result of finding a correct coloring within 1000 individuals is not inconsistent with our results in Table 3, where for population size set at 20, we consistently find a correct coloring after encountering an average of 116 individuals.

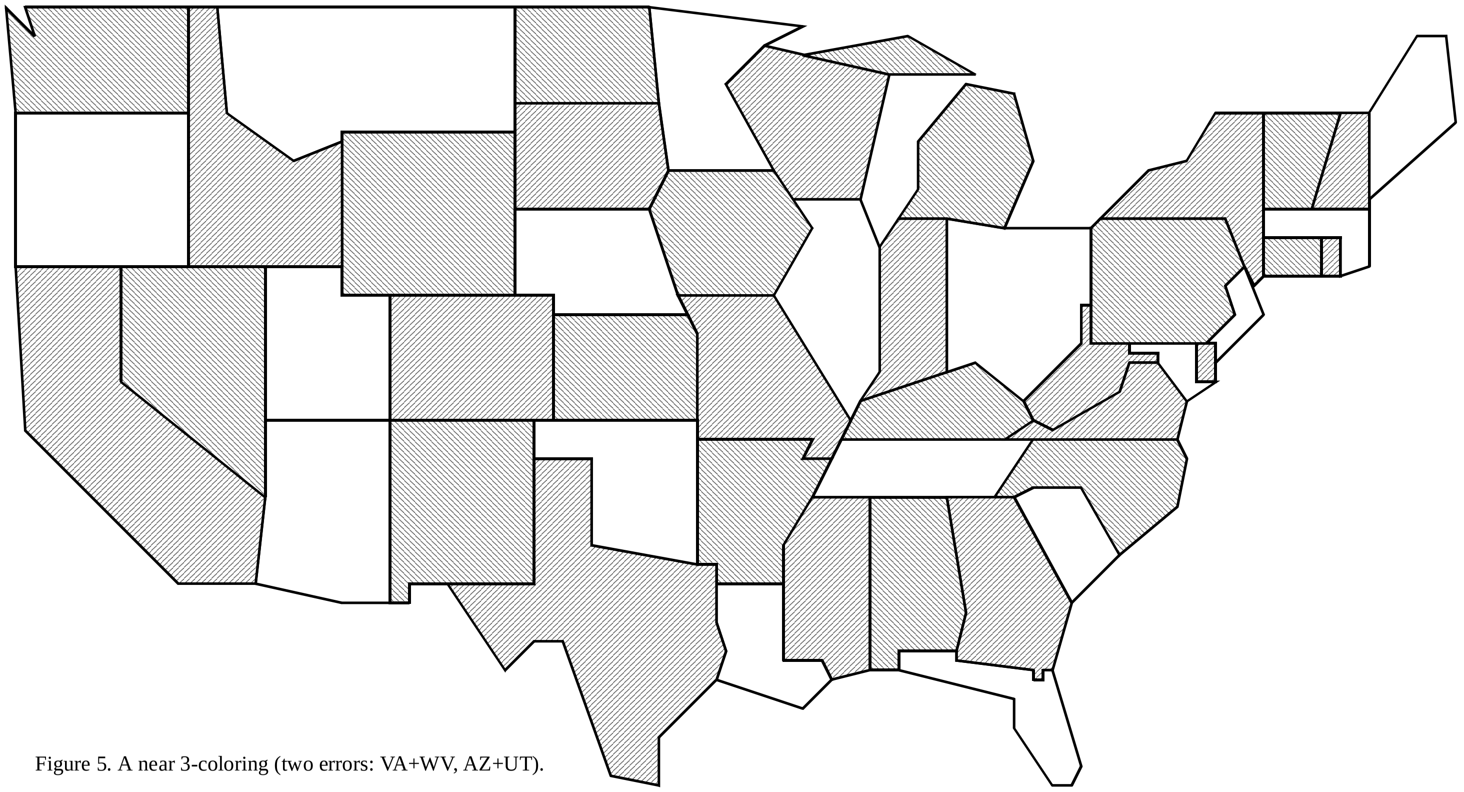

Figure 5. A near 3-coloring (two errors: VA+WV, AZ+UT).

```
Color 1: AL AR CT IA KS KY MI NV NM NC ND PA VT WA WY
Color 2: CA CO DE GA ID IN MS MO NH NY RI SD TX VA WV WI
Color 3: AZ FL IL LA ME MD MA MN MT NE NJ OH OK OR SC TN UT
```

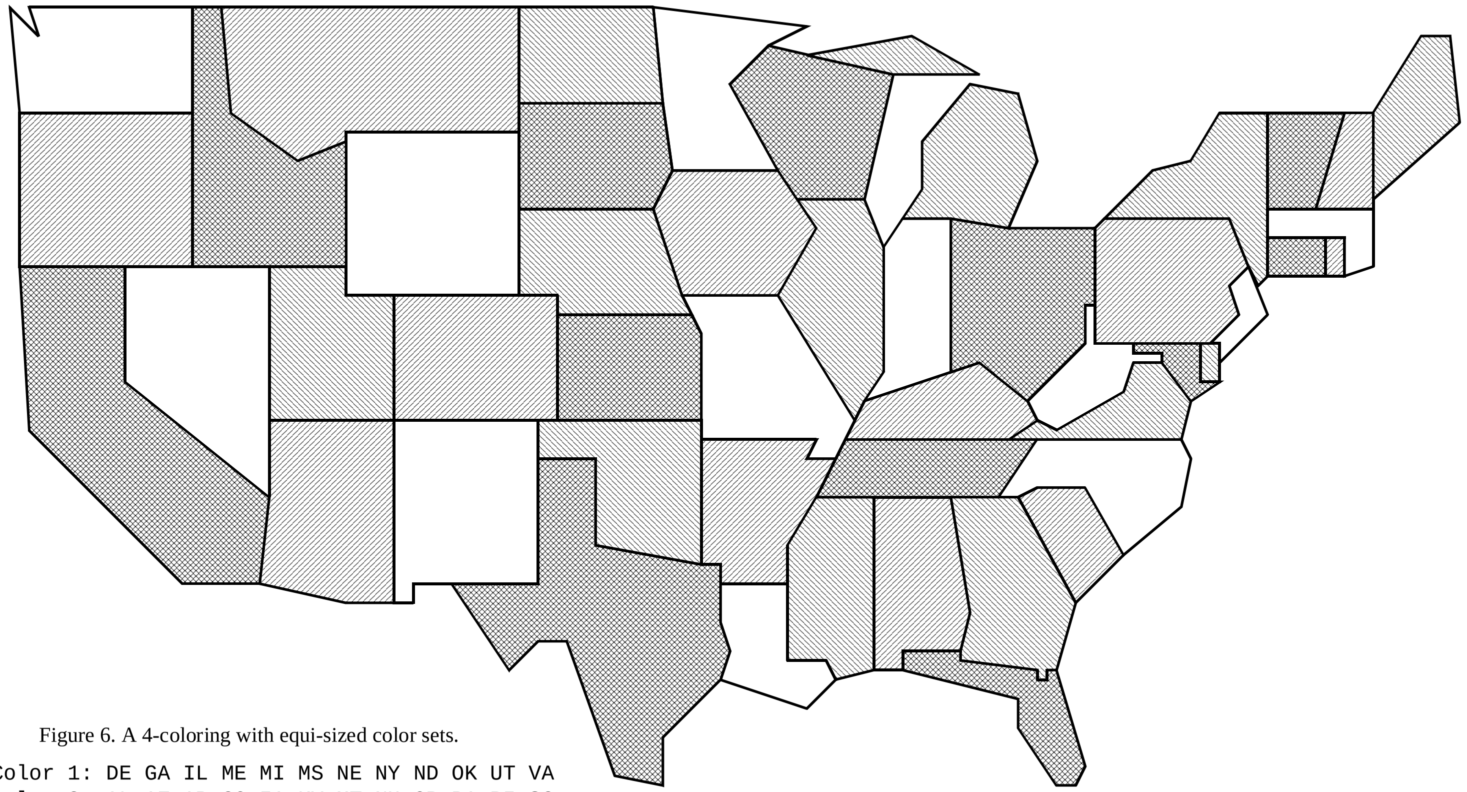

Figure 6. A 4-coloring with equi-sized color sets.

```
Color 1: DE GA IL ME MI MS NE NY ND OK UT VA
Color 2: AL AZ AR CO IA KY MT NH OR PA RI SC
Color 3: CA CT FL ID KS MD OH SD TN TX VT WI
Color 4: IN LA MA MN MO NV NJ NM NC WA WV WY
```

We next asked ourselves whether the United States can be three-colored. Initially we did not know the answer, and used our algorithm to explore three-colorings. As it turns out, a three-coloring is impossible. There are two trouble spots, in the vicinities of Utah and of West Virginia. Consider: if we color CA, OR, and NV with three (necessarily) different colors, then ID must be the same color as CA, and AZ must be the same color as OR. That leaves UT bordering three states bearing three different colors; thus, UT requires a fourth color. On the other hand, it is possible to almost three-color the United States, by allowing two pairs of adjoining states which are incorrectly colored the same color. Figure 5 shows such a three-coloring.

The solution to four-coloring which is shown in Figure 4 has subsets of different sizes. In fact, in that solution there are only half as many states colored in color 1 as in color 4! It might be desirable that there are equal numbers of states colored the different colors. We introduce this as an additional constraint.

Thus, now we wish to four-color the United States so that each different color is used on the same number of states, namely, 12. This leads us to think of each subset as having two error components. They are: a coloring error (the number of state pairs which share a boundary line yet are incorrectly included in the same color group) and a size error, which we take to be the (absolute) difference between the subset's actual size versus the ideal subset size of 12. How should we weight the two error components? In our first effort we let the error of a subset be the average of these two components, which weights them equally. The results were not as desired: over 30 trials, the best partitions found on a trial had four subsets of size 12, but the subsets included coloring errors. Obviously, coloring error needed to be given more emphasis. After several experiments, in which the coloring error component received more and more emphasis over the size error, we achieved the results we desired.

Our best results were obtained when subset error was computed by weighting coloring error at 90% and size error at 10%. Population size was set at 200; on each trial we allowed a maximum of 50 generations; we ran 30 trials. On every one of the 30 trials, we found the kind of coloring we desired, that is, a correct four-coloring in which each color is used on exactly 12 states. Such a coloring was found on average generation number 9.2, which implies that 1,840 partitions were encountered before finding one with the targeted characteristics. Figure 6 shows such a coloring. Also, Table 4 summarizes the results just reported. That table also shows that decreasing the population size did not have disastrous

Table 4. Four-coloring the US with equi-sized color sets.
In all trials, the maximum number of generations = 50.

Population size:	200	100	50	20
Number of trials	30	30	30	30
Trials finding correct coloring	30	28	26	22
Average generation to find correct coloring, on successful trials	9.2	14.3	15.2	19.4
Average # individuals to find correct coloring, on successful trials	1,840	1,430	760	388

consequences for performance. At population size 100 (and with all other parameters unchanged), on 28 of 30 trials an optimal partition was found, on average generation number 14.3, which means on those trials 1,430 partitions were encountered before finding one with the targeted characteristics. The table also shows results when population size was dropped to 50 and 20.

6. Conclusions

Our aggressive genetic algorithm Eager Breeder is an incremental improvement over the Greedy GGA algorithm of Falkenauer, for solving the problem of partitioning numbers into subsets whose sums are as nearly equal as possible. Eager Breeder has almost the same success and only one-third the cost of Greedy GGA.

We extended our approach to the more general setting where we are to partition a set, and the environment provides us a measure of the goodness of not just an entire partition but also of each subset in it. There we got excellent results on two artificial problems, the first of which was to partition a set of 51 elements into 12 subsets of very diverse sizes, and the second of which was to partition a set of 48 elements into eight subsets of the same size, six.

Finally, we applied our approach to three map-coloring problems. There again we obtained excellent results. With low cost we found a four-coloring of the United States, and a four-coloring with equi-sized color sets. As well, we found a three-coloring with a minimal number (two) of coloring errors.

Set partitioning problems appear to be fertile ground for the application of genetic algorithms, provided we can identify a good fitness or error measure for the subsets in a partition, and further provided we can identify good heuristics to apply for the particular partitioning problem at hand. In the research of this paper, we have used varied heuristics for the job of distributing as-yet-unassigned set elements to a child's subsets, halfway along in the construction of a child partition from its parents by mating with crossover.

References

[1] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI (1975).

[2] D. R. Jones and M. A. Beltramo, *Solving partitioning problems with genetic algorithms*, Proc. 4th Intnl. Conf. on Genetic Algorithms, eds. K. R. Belew and L. B. Booker, Morgan Kaufmann, San Francisco (1991) 442-449.

[3] E. Falkenauer, *Solving equal piles with the grouping genetic algorithm*, Proc. 6th Intnl. Conf. on Genetic Algorithms, ed. L. J. Eshelman, Morgan Kaufmann, San Francisco (1995) 492-497.

[4] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA (1989).

[5] K. Appel and W. Haaken, *Every map is four-colourable I: discharging*, Illinois J. Math **21** (1977) 429-490.

[6] K. Appel and W. Haaken, *Every map is four-colourable II: reducibility*, Illinois J. Math **21** (1977) 491-567.