A Genetic Algorithm for Coverage Problems

Colin Johnson
Computing Laboratory, University of Kent
Canterbury, Kent, CT2 7NF, England
C.G.Johnson@kent.ac.uk

ABSTRACT

This paper describes a genetic algorithm approach to *coverage problems*, that is, problems where the aim is to discover an example for each class in a given classification scheme.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Genetic Algorithms,Classification

1. DEFINING COVERAGE PROBLEMS

Classification is a well-studied problem where a set of objects is placed into a number of categories according to some criteria or inductive scheme. This paper describes the "opposite" problem, which we term *coverage*. That is, given a classification scheme (i.e. a function that places objects from a set into a set of classes), find example objects for each of the classes. Clearly for a dataset that is stored in a database, this is trivial; however, for problems where the examples need to be calculated or searched for, the problem becomes harder.

The coverage problem is a generalisation from many specific problems in computing and its applications. For example, computing problems such as test suite generation, randomized compilation, specialisation in multi-agent systems and information retrieval all contain problems that can be characterised as coverage problems. Furthermore, a number of applied problems such as aspects of the protein folding problem in bioinformatics, and data-driven approaches to computational mathematics also contain such coverage problems.

Coverage problems have some similarity to multimodal optimization problems. However, in coverage problems the interest is in finding *any* example for each predefined class, rather than a "best" example. Perhaps the closest similarity is with the work on computational creativity, in particular the important problem of recognizing and searching for novelty.

```
INPUT: Description of objects, classes,
  and the classification function
Generate random population P
Assign fitness of 1 to each member of P
Generate list E of exampled classes, initially empty
LOOP (until timeout or all classes covered)
  Genetic Algorithm step on P producing child population C:
    Roulette-Wheel Selection,
      based on pre-assigned fitness values
    Crossover (uniform, crossover probability=0.9)
    Mutation (probability 0.001 per string position)
  Evaluate the class of all members of C
  Determine which members of C represent unexampled classes,
    by reference to the list E
  If (all members of C are exampled)
     Reinitialise C
     Assign fitness of 1 to each member of C
  else
     Report the example-class pair for each
       unexampled member of C
     Add the newly-exampled classes to E
     Set fitnesses in P to zero
     For each parent in P:
        Add 1 to fitness for each unexampled child
     For each child in C:
        Add together the fitnesses of its two parents
  Copy C into P (members and fitness values)
END LOOP
```

Figure 1: Pseudocode description of the coverage genetic algorithm

2. A GENETIC ALGORITHM FOR COVERAGE PROBLEMS

We have devised a genetic algorithm for coverage problems. Standard mutation and recombination operators are used; however, fitness is calculated using a novel method of dynamic fitness assignment called *fitness passback* that encourages the continued production of examples that fit into classes for which an example has not yet been found.

The passback procedure works as follows. The class of each object in the current population is determined. These are then identified as *exampled* or *unexampled* based on whether an example for that class has already been discovered by a previous generation. A fitness value is then generated in the parent population, by tallying up the number of unexampled children that that parent has produced. These values are then passed onto the children and used as the fitness function for the current generation. This algorithm is detailed in figure 1.