

Materials: Genetic NIM learner demonstration program
Projectable of wall-follower robot problem and a solution
Projectable of Nilsson figures 4.2 .. 4.9
Mitchell pp. 17-18 to read; 19-21
Projectable of payoff matrix - p. 18

I. Introduction

-- -----

- A. Like neural networks, genetic algorithms represent an attempt to imitate the architecture of intelligence present in nature - in this case, the "intelligence" exhibited by a species (not individuals) as it evolves to better fit its niche in the ecosystem. Frequently, we use a genetic algorithm to evolve a program that effectively solves a given problem. (Hence, this field is sometimes called genetic programming.)
- B. A genetic algorithm is a form of search algorithm, best suited to problems where the solution cannot be found by more traditional means.
 - 1. The example we will use is not really a good example of a place where GP is useful, since the task can be programmed directly in a straight-forward way. (But it is a good example for understanding the process itself).
 - 2. However, a similar approach can be used to evolve a program that could not be easily coded directly - e.g. work that Gary Parker shared when he visited in the spring of 1999 on work using GP to evolve a program that enables a hexapod robot to walk, or evolving a solution to an optimization problem whose algorithmic solution is NP-complete.
- C. This field is fairly new. Some indication of how new it is can be seen by what I discovered when I looked for books on it.
 - 1. In 1999, there was no subject heading for "Genetic Algorithms" or "Genetic Programming" in the online card catalog from NOBLE.

(There is now for both)
 - 2. In 1999, we had one book on the subject - An Introduction to Genetic Algorithms by Melanie Mitchell. However, it was housed in the Genetics section under Biology in the library! (LC classification in the book).

(Today it's still there, however, it does not come up if you do a subject search for "Genetic Programming" or "Genetic algorithm". We do have two more books that come up under a "Genetic algorithm" subject search, one in the CS section and one in the business section. There are three other books we don't have cataloged by the Noble libraries, plus a couple of e-books).

- D. For a problem to be a good candidate for using a genetic algorithm, several things need to be true.
1. The problem can't be "all or nothing" - that is, it must be meaningful to talk about "solutions" which are less than perfect. (This does not preclude the possibility of there being a perfect solution - but it is to say that a solution that is less than perfect must still be a useful solution. Moreover, given a set of solutions, there must be some straightforward way to evaluate their relative fitness, so that it is meaningful to talk about "better" solutions.
 2. It must be possible to break a solution up into "genes" - each of which represents part of the solution - which are, at least to some extent, independent of each other.
 3. Some examples of problems which lend themselves to this approach.
 - a. Control problems (like the pole balancer we looked at in connection with neural networks or teaching a robot to walk).
 - i. A perfect solution would keep the system behaving in the desired way endlessly, but a solution that keeps it behaving the desired way for a long time is still useful. (Even we sometimes fall down while walking!).
 - ii. Solutions can be compared based on how long they keep the system behaving in the desired way before failing.
 - iii. The problem can be decomposed into "genes" representing the relationship between various percepts and actions.
 - b. Optimization problems like travelling salesman.
 - i. For large problems, we generally have to accept a good solution, even if it is not possible to find a provably optimal one.
 - ii. It is easy to compare solutions in terms of total cost.
 - iii. The relative ordering of pairs of cities can be considered "genes"
 - c. Two-player games
 - i. For an interesting game (e.g. chess or go) it is possible to be a good player without always winning.
 - ii. It is possible to compare players by percentage of wins against a pool of other players or even head-to-head play. (Though the former is probably the preferred approach for GA).
 - iii. For a complex game, the "genes" may be the various features that the player considers when choosing a move.

II. Structure of a Genetic Algorithm

- A. A GA proceeds by evolving a population of individuals, each of which represents a different possible solution to the problem at hand. (In the case of GP, each individual represents a possible program.)
1. Possible solutions to a given problem are encoded as a sequence of "genes", each of which may be (depending on the problem) an individual bit, a value from a discrete set of possible values, or a real number.
 2. The initial population is constructed by choosing values for each gene at random. It is therefore unlikely that any individual in the population constitutes a good solution to the problem.
 3. Over time, the population evolves to consist of increasingly fit individuals, until an individual representing a satisfactory solution is found.
 4. We will use the game of NIM as an example. This is not a good example of a problem where a genetic approach is really useful (since we know an algorithm for it), but it does provide a simple illustration of how genetic programming can be applied.
 - a. We will represent a strategy for playing NIM as a vector of moves corresponding to each possible state of the pile - each of which we will consider to be a gene.

For example, with a pile size limited to 10 and moves limited to taking 3 items, there would be 10 genes, each a number in the range 1 .. 3 (except that the first gene would have to be 1, and the second would have to be either 1 or 2 to comply with the rules of the game.)

In this case, one possible solution might be

1 1 3 2 2 2 1 3 1 3

This says "if the pile contains 1 item, take 1; if it contains 2, take 1; if it contains 3, take 3; if it contains 4 take 2 ..."
 - b. Obviously, the above is far from a perfect solution. However, it is still meaningful to call it a solution.

Note that it will win in some cases - even if playing against an algorithmic player - e.g. game starts out with 8 items; opponent takes 2 leaving 6; program takes 2 leaving 4; opponent takes 1 leaving 3; program takes 3 and wins.
 - c. It is, of course, easily possible to create a random population of solutions by randomly choosing values in the range 1 .. maximum move for each gene. (Except that gene 1 must be a 1, gene 2 must be a 1 or a 2 ...)
- B. A fitness function is defined that measures the extent to which each individual represents a good solution to the problem. Initially, given random individuals, the fitness function for each individual will be small; but there will be some that are better than others, and the GA will attempt to evolve their good points into the next generation of possible solutions.

1. For the NIM Example, solutions can be compared by having each play against the pool of others and measuring fitness as percentage of wins.
2. Or, in this case, since we know an algorithm we can measure the fitness of a solution as $(\# \text{ of rights}) / (\text{rights} + \text{wrongs})$ - where we don't count situations for which there is no right move $((\text{pile size}) \bmod (\text{maximum move} + 1) = 0)$.

For example, the fitness of our example solution would be evaluated as follows , where R is right, W is wrong, and - is don't care:

```

1 1 3 2 2 2 1 3 1 3
R W R - W R W - R W
4/(4+4) = 0.5

```

3. For the initial, random population, we would expect the average fitness of an individual to be $1 / (\text{maximum move})$, since this is the probability of randomly choosing the right value for a given gene, and the genes are independent of each other. However, it is likely that some individuals will be more fit than this, while others will be less.

DEMO: genetic NIM - initial random population.

C. Evolution of the population consists of a series of generations.

1. In each generation, the individuals in the population are tested and the most fit are allowed to reproduce.
2. Typically, reproduction is done by crossing two fit individuals, in the hope that their offspring will inherit the good features of each and thus be even more fit (though, of course, some inherit bad features from each parent and end up less fit.)
3. A small amount of mutation is also often allowed.
4. Note that, typically, each generation consists of a completely new collection of individuals created by crossover and/or mutation from the individuals in the previous generation-i.e. individuals "live" for only one generation. (It is also possible to allow a subset of the most fit individuals in one generation to survive unchanged to the next.)

D. Crossing is handled as follows:

1. If each individual has the same number of genes arranged in some kind of sequence - we can pick a crossover point in the sequence at random, generating two offspring - e.g.

```

Individual A:      A1 A2 A3 A4 A5 A6 A7 A8
Individual B:      B1 B2 B3 B4 B5 B6 B7 B8

```

Offspring if we cross between genes 2 and 3:

```

A1 A2 B3 B4 B5 B6 B7 B8
B1 B2 A3 A4 A5 A6 A7 A8

```

(We may choose to keep both offspring, or just one.)

2. NIM Example

a. Suppose we want to cross the solutions

1 1 3 2 2 2 1 3 1 3 and 1 2 3 3 2 1 1 2 3 2

just after the fifth gene

The "children" of this cross are

1 1 3 2 2 1 1 2 3 2 and 1 2 3 3 2 2 1 3 1 3

b. Comparing the fitness of the "children" to that of the parents is instructive

i. Fitness of the parents

1 1 3 2 2 2 1 3 1 3
R W R - W R W - R W

$$4/(4+4) = 0.5$$

1 2 3 3 2 1 1 2 3 2
R R R - W W W - W R

$$4/(4+4) = 0.5$$

ii. Fitness of the children

1 1 3 2 2 1 1 2 3 2
R W R - W W W - W R

$$3/(3+5) = 0.375$$

1 2 3 3 2 2 1 3 1 3
R R R - W R W - R W

$$5/(5+3) = 0.625$$

E. Mutation is done by randomly altering an individual gene. This may result in a solution that is less fit, more fit, or having the same fitness as the original.

1. Mutation is often important, because it may be that no individual in the initial population contains the "correct" value of some gene, or perhaps the "correct" value of a gene is lost early due to incompatibility with some other genes that are selected away later.
2. Of course, mutation can also be harmful, causing a "correct" value that was discovered by selection to be lost.
3. Mutation is usually done with a fairly small probability - e.g. (say) 1% of the individuals in the new generation may undergo mutation.

4. NIM Example:

a. Suppose we mutate 1 1 3 2 2 2 1 3 1 3 at the first gene.

Any change we make will produce a less fit individual, since that gene was "right"

b. OTOH, if we mutate this individual at the second gene, a change to 2 will produce a more fit individual, while a change to 3 will produce no change.

F. Everything is done probabilistically:

1. We have already noted that the initial population is generated randomly.
2. Some implementations may allow some individuals to survive unchanged to the next generation. In this case, the individuals that survive can be selected randomly, with probability based on fitness - i.e. the more fit individuals have a higher probability of survival. (In some implementations, the most fit individuals may be guaranteed the right to survive unconditionally).
3. The individuals that reproduce may be selected randomly, with a probability based on fitness - i.e. the more fit individuals are given a higher probability of selection and the less fit ones a lower probability. (In some implementations, the most fit individuals may be guaranteed an opportunity to reproduce.)
4. The crossover point used when crossing parents is chosen randomly.
5. Whether or not a given gene is mutated is determined randomly with a predetermined - usually quite low - probability - typically independent of fitness - and if it is mutated, the change is determined randomly.

G. Depending on the nature of the problem, repetition of the process of creating new generations may continue until an individual is found that is adjudged to be perfectly fit, or until fitness stops improving, or after a predetermined number of generations.

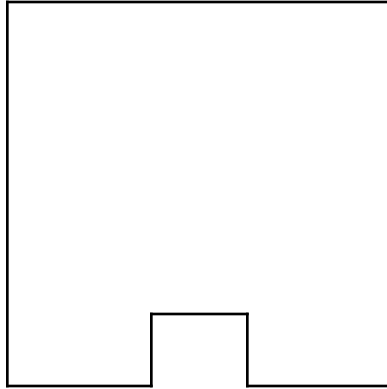
H. Demonstrate NIM Example

1. Observe population as evolution with population 500 is done generation
2. Demonstrate game
3. Demonstrate maximum pile size 20

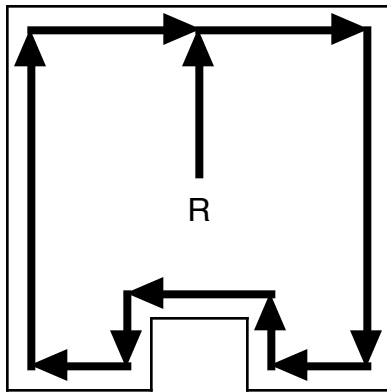
III. Another Example (Omit if Insufficient Time)

- A. The Nilsson book that we have used previously in the course (and in which you have done some reading) includes an example of genetic programming, where a program to solve a problem is evolved by genetic means.
- B. The problem is to evolve a program for a robot such that, when it is placed in an enclosed room, it moves to a wall and follows the wall around the room.

E.g. Given a room like this: (PROJECT)



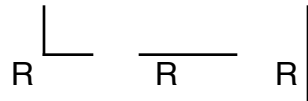
We want the robot to do something like this (though we don't care how it gets to the wall initially or whether it moves clockwise or counterclockwise). (PROJECT)



C. The primitives from which the program is to be constructed are the following.

1. Tests: PROJECT NILSSON FIGURE 4.2

- a. n, e, s, w - return true (1) if movement in the specified direction is blocked by a wall
- b. ne, se, sw, nw - return true (1) if there is a wall in the specified direction - including possibly a corner that doesn't actually block one of the robot's moves - e.g. ne would be true in all the following cases:



2. Boolean connectives:

- AND (X, Y): if X == 0 then 0 else Y
- OR (X, Y): if X == 1 then 1 else Y
- NOT (X): if X then 0 else 1
- IF (X, Y, Z): if X then Y else Z

3. Actions: north, east, south, west - move one block in the specified direction (if possible - otherwise do nothing) - then terminate the current iteration of the main loop

Note that the program that will be evolved is functional in style - e.g

```
IF ( NOT(n), north, (IF (NOT (s), south, east) ) means
if north is not blocked
  move north
else if south is not blocked
  move south
else
  move east
```

4. There is a main loop that executes the program over and over

D. Example of a program that would solve the problem:

NILSSON FIGURE 4.3 - PROJECT

Trace through how this program works in the example problem

E. Applying Genetic Programming to this problem

1. A population of random programs is created
2. For each generation, the fitness of each program in the current population is evaluated. Fitness is measured as "number of squares next to the wall that the robot visits in some number of iterations of the main loop."
 - a. In the particular case Nilsson used, the room had 32 squares next to the wall.
 - b. Fitness was measured as the number of these squares the robot visited in 60 moves from ten different random starting positions. [A fitness score of 320 would be perfection].
3. Crossover is handled by switching subtrees between parents

NILSSON FIGURE 4.4 - PROJECT

4. Mutation could be handled by selecting a random subtree and replacing it with a new randomly-grown subtree.

F. Experimental results

1. The most fit individual in Generation 0

NILSSON FIGURE 4.5 - PROJECT

2. The most fit individual in Generation 2

NILSSON FIGURE 4.6 - PROJECT

3. The most fit individual in Generation 6

NILSSON FIGURE 4.7 - PROJECT

4. The most fit individual in Generation 10 - a program that is actually a 100% solution.

NILSSON FIGURE 4.8 - PROJECT

5. Evolution of fitness over the generations

NILSSON FIGURE 4.9 - PROJECT

IV. Another Example

--- -----

A. Melanie Mitchell's book - cited earlier - discussed some experiments with using GA's to evolve a strategy for a game known as "the prisoner's dilemma".

- 1. READ Mitchell pp 17-18; TRANSPARENCY of payoff matrix (p 18)
- 2. Mitchell's book discusses experiments done by Axelrod on this game.

B. Note that any one game can be categorized in one of 4 ways (CC - both players cooperated; CD A cooperated and B defected; DC; DD).

C. Because the programs in the tournament based their strategy on the last three games played with the same player, each with a move by each player, and with each move having two possible values, a strategy must be able to cope with 64 possible histories:

3 games/history
 2 player moves/game
 2 choices/player move

For each history, it must make a choice to either defect or cooperate on the next game. Thus, a strategy may be encoded as a 64 genes, each of which is either a C or a D, each representing the choice called for by the strategy for one possible history - e.g.

choice to make if all games were CC	choice to make if first two games were CC and last was CD	choice to make if all games were DD
--	---	-------	--

For example, TIT-FOR-TAT for Player A would be encoded as

CDCDCDCDCD ... CD (i.e. A always does what Player B did on the last game)

D. Results: READ Mitchell page 19 bottom - 21 top