Local Search Self-Check

605.645 – Artificial Intelligence

The purpose of this self-check is to make sure you understand key concepts for the algorithms presented during the module and to prepare you for the programming assignment. As you work through problems, you should always be thinking "how would I do this in code? What basic data structures would I need? What operations on those basic data structures?"

Suppose we have the following states and their values under the fitness function f():

State	1	2	3	4	5	6	7	8	9	10
f()	2.45	2.78	3.14	3.31	3.23	2.98	2.72	3.09	3.37	3.26

The successors() function for a given state, n, returns the states on the left and right of the given state, if they exist, so that successors(3) returns [2, 4] while successors(1) returns [2]. find-best-child() uses successors and the evaluation function to return the single successor with the highest value.

Using the pseudocode from the module, answer the following questions.

General.

1. If the state is 5, what does successors() return?
2. If the state is 3, what does find-best-child() return?

Hill Climbing

1. Suppose we start randomly in state 7. Show and explain the exploration path of hill-climbing. Is this a global maximum?
2. Suppose we start in state 2. Show and explain the exploration path of hill-climbing. Is this a global maximum?
3. What elaborations of hill-climbing might improve the performance of hill-climbing? Are the improvements guaranteed?

Beam Search

1. Suppose we start randomly with states 2 and 7 so that k=2. Show and explain the exploration path of beam search.

Simulated Annealing

1. Suppose we find ourselves in state 4. With reference to the pseudocode, explain how simulated annealing might permit further local search.

Genetic Algorithm

The following questions deal with different aspects of the Genetic Algorithm. For your programming assignment this week, you will be implementing two versions of the Genetic Algorithm.

Representation

The Canonical (or Original) Genetic Algorithm works with an encoding. Following the analogy of biology, which uses A, T, G, C as its encoding, the Canonical GA uses 0 and 1 as the encoding. No matter what the actual values for the variables might be, they are encoding as a List of bits, 0's and 1's, called a *chromosome*. This encoding represents the *genotype* of an individual (candidate) solution (think "genes" -> "genotype"). Unlike human beings, which have 23 base *pairs* of *chromosomes*, for the Canonical GA, there is just a single *chromosome*.

And, following the analogy further, we require a function to "transcribe" the genes into the actual variable values, the *phenotype* (think "phenomenon" -> "phenotype"), the actual values of the variables we can use.

Suppose we have a problem with three variables whose integer values can range from 0 to 7. Answer the following questions:

1. What does a random (ie, arbitrary) individual for this problem look like (ie, what does the chromosome look like? Use a List).

2. What is the phenotype for your individual from #1?

3. If you had the phenotype for an individual [4, 2, 5], what does its genotype (chromosome) look like?

4. Look at the pseudocode for the genetic algorithm, do you ever need to go from phenotype to genotype (#3)?

5. What would a population of 3 individuals like those from Q1 look like? Use a List of Lists.

   (your real population needs to be much bigger!)

Crossover and Mutation

The following function:

crossover(gene-index, parent-1, parent-2)

returns two children. The gene-index specifies that the split should take place immediately before that position. So if the location were 5, then with reference to parent 1, genes 0-4 would go to the son and genes 5-9 would go to the daughter. For parent-2, genes 0-4 would go to the daughter and 5-9 would go to the son.

If the two parents are:

```
0101110110010000
1010010111101100
```

a. What are the children if the gene index is 5?
b. What are the children if the gene index is 12?
c. What are the children if the gene index is 0?

1. There are several varieties of mutation operators. One such mutation operator picks a random location and a random symbol.

Suppose we have a child from a problem with 3 possible values for each gene :

```
0120112012022111 0
```

a. What is the result of a mutation with location 9 and symbol 0?
b. What is the result of a mutation with location 2 and symbol 2?

2. If the mutation rate is 0.05, and rand() generates 0.0237, does mutation happen or not?
3. If the crossover rate is 0.80, and rand() generates 0.8976, does crossover happen or not? What if rand() generates 0.4329?

Note:

For stochastic code, it is tempting to write something like:

function foo(p, a, b):

```
    if rand() < p:
        return a
    return b # no need for an “else” if you’ve returned from the “then”
```

This makes it very difficult to test:
foo(0.75, 1, 2) -> returns ????
foo(0.75, 3, 4) -> returns ????

Consider passing the random value to foo():

```
Function foo(p, r, a, b):
    if r < p:
        return a
    return b
```

foo(0.75, 0.83, 1, 2) -> returns 2
foo(0.75, 0.29, 3, 4) -> returns 3

When you actually use the function, you call it like so foo(0.75, rand(), 1, 2)