CT421 Artificial Intelligence

Introduction

Recap

- Monte Carlo Tree Search
- Local Search - Hill climbing
- Simulated Annealing

Evolutionary Computation: genetic algorithms

- Directed search algorithms inspired by biological evolution
- Developed by John Holland in the 1970s to study the adaptive processes occurring in natural systems
- Have been used as search mechanisms to find good solutions in a range of problems

High level Algorithm

```
produce an initial population of individuals
evaluate the fitness of all individuals
while termination condition not met do
   select fitter individuals for reproduction
   recombine between individuals
   mutate individuals
   evaluate the fitness of the modified individuals
   generate a new population
End while
```

Representation of chromosomes

Many potential options

- Bit strings
- Integers
- Real numbers
- Lists of rules/instructions
- Programs

Initialisation

- Typically, start with a population of randomly generated individuals
- Use a previously saved population
- Use a set of solutions provided by a human expert
- A set of solutions provided by another algorithm

Guidelines/Rules of Thumb

- Use a data structure as close as possible to the natural representation
- Write appropriate genetic operators as needed
- If possible, ensure that all genotypes correspond to feasible solutions

Selection

- Fitness based (roulette wheel)
- Rank based
- Tournament selection
- Elitism

Crossover

- Crossover is a vital component
- It allows the recombination of good sub-solutions
- Speeds up search early in evolution

Mutation

- Mutation represents a change in the gene.
- The main purpose is to prevent the search algorithm from becoming trapped in a local maxima.

Simple example - one-max

- Suppose we want to maximize the number of ones in a string of l binary digits
- Similarly, we could set the target to be an arbitrary sting of 1s and 0s
- An individual can be represented as a string of binary digits
- The fitness of a candidate solution is the number of ones in its genetic code

Related Problem

- Maintain the same representation as before
- Modify the fitness function as follows:
  - For strings of length l with x 1s; if $x > 1$, then fitness is equal to x. Else fitness is equal to $l + k$, for $k > 0$

Knapsack Problem

Knapsack Problem

- Given two n-tuples of values $< v_0, v_1, \ldots v_n >$ and $< w_0, w_1, \ldots w_n >$, chose a set of items i from the n items such that $\sum_i v(i)$ is maximised and $\sum_i w(i) \leq T$

Knapsack Problem

- Representation?
- Fitness?
- Initialisation?
- Mutation?
- Crossover?

Schemata Theorem

Terminology

- Consider a genetic algorithm using only a binary alphabet.
- {0,1,*} is the alphabet, where is a special wild card symbol which can be either 0 or 1
- A schema is any template comprising a string of these three symbols
- For example, the schema [1*1*] represents the following four strings [1010], [1011], [1110] and [1111]

Terminology

- The *order* of a schema S is the number of fixed positions (0 or 1) presented in the schema
- For example: [01*1*] has order 3, [1*1*10*0] has order 5.
- The order of a schema is useful in estimating the probability of survival of a schema following a mutations

Terminology

- The defining length of a schema S is the distance between the first and last fixed positions in the schema
- For example, S1 = [01 $*$ 1$*$], the defining length is 3
- The defining length of a schema indicates the survival probability of the schema under crossover

- Given selection based on fitness, the expected number of individuals belonging to schema S at generation $i + 1$ is equal to the number of them present at generation i multiplied by their fitness over the average fitness.
- “above average” schema receives an exponentially increasing number of strings over the evolution

- The probability of a schemata S surviving crossover is dependent on the defining length.
- Schemata with above-average fitness with short defining lengths will still be sampled at exponentially increasing rates

- The probability of a schemata S surviving mutation dependent on the order of the schema
- Schemata with above-average fitness with low orders will still be sampled at exponentially increasing rates

- **Schema Theorem** Short, low-order, above-average schemata receive exponentially increasing representation in subsequent generations of a genetic algorithm
- Building Block hypothesis: A genetic algorithm navigates the search space through the rearranging of short, low-order, high-performance schemata, termed *building blocks*

Further Concepts

- Diversity
- Drift
- Co-evolution
- Deception
- NK landscapes
- Epistasis
- Novelty Search