CS324: Nature Inspired heuristic Search methods. **Spring 2015**

Lecture 16: March 23, 2015

Lecturer: K.R. Chowdhary *: Professor of CS (VF)*

16.1 Nature inspired Techniques

These are other search techniques based on "natural phenomena". Under this we are going to discuss two techniques: 1) *Simulated annealing*, which is based on changes in the properties of metals and alloys due to heating them to higher temperature and then slowly decreasing their temperature; and 2) Some thing based on the Darwin's theory of evolution, called *genetic algorithms*.

16.2 Simulated Annealing

Simulated annealing (SA) is a probabilistic search for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. The name and inspiration come from annealing in metallurgy.

In SA we make one change from the normal heuristic search; we attempt to minimize the function's value instead of maximizing. This is like a *valley descending* rather than *hill-climbing*.

The physical substances usually move from higher energy configuration to lower levels, so that the valley descending occurs naturally. But, there is some probability that a transition to higher energy will occur, given by,

$$p = e^{\frac{\Delta E}{kT}} \tag{16.1}$$

where ΔE is positive change in energy level, T is temperature, and k is *Boltzmann's* constant. As per this property, the probability to a large uphill move will be lower than probability of small move. Also, the probability that a large uphill move will take place, decreases as the temperature (T) decreases. In other words, the uphill moves are more possible when temperature is high, but as the temperature decreases, relatively small uphill moves are made until finally process converge to a local minimum configuration.

In search techniques, change in E (ΔE) is equal to change in heuristic function. The constant k represents the correspondence between the unit of temperature and unit of energy. Since it is constant, we take probability p in equation (16.1) as,

$$p' = e^{\frac{\Delta E}{T}}. \tag{16.2}$$

SA mimics annealing process in metallurgy by combination of random search and hill-climbing. During metallurgical annealing, alloys are cooled at a controlled rate to allow for the formation of larger crystals. Larger crystals are chemically at a lower energy state than smaller ones; alloys made of crystals in the lowest

energy state are comparably stronger and more rigid. At a high temperature, the search is a random walk, and as the temperature lowers the search gradually transits to a local search. Capturing this idea in an algorithm yields a random process over a space of configurations where the probability of actually moving to a new configuration is determined by the difference in energy and the current temperature. The algorithm 1 shows the steps for this process.

Algorithm 1 Simulated-Annealing

```
T = high
current state=randomly generated Solution
sofarbest = current state
while current state is not acceptable and stop criteria is not met do
    new state = result of applying next operator to current state
    Evaluate new state
    ΔE = (value of current) - (value of new state)
    if new state is goal state then
        Return and quit
    end if
    if score(new state) is better than score(current state) then
        current state = new state
        sofarbest = new state
    else
        current state with probability p' = new state (It is implemented by generating a random number
        r ∈ [0, 1]. If r < p', then this move is accepted, otherwise not)
    end if
    decrease T
end while
return 'sofarbest'
```

The Boltzmann probability function tends to return True at higher temperature and False at lower temperature; thus in essence the search gradually shifts from random walk to local hill climb.

16.3 Genetic Algorithms

The genetic algorithms (GAs) are search procedures based on natural selection and genetics. A GA processes a finite population of fixed-length binary strings. In practice, bit codes, *kry*-codes real (floating-point) codes, permutation (order) codes, etc. Each of these has their place, but here we examine a simple GA to better understand basic mechanics and principles.

A simple GA consists of three operators: *selection, cross-over, and mutation.* Selection is the survival of the fittest within the GA. The figure 16.1 shows the sequence of operations on a population P_n and producing next population $P_{n=1}$. On the basis of these fitnesses, the selection phase assigns the first individual (00111) zero copies, the second (111000) two, and the third (01010) one copy. The algorithm 2 shows the steps for search using GA.

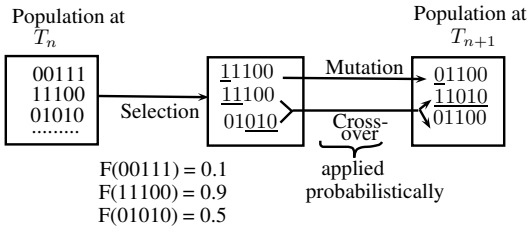

Figure 16.1: Sequence of operations in GA.

Algorithm 2 Genetic-Algorithm(Input: Initial Population)

1: **Initialize** the population with random candidate solutions
2: **Evaluate** each candidate
3: **repeat**
4: **Select** parents
5: **Recombine** pairs of parents
6: **Mutate** resulting offspring
7: **Evaluate** new candidates
8: **Select** individuals for the next generation
9: **until** termination condition/goal is reached

16.3.1 Mutation Operator

If selection and crossover provide much of the innovative capability of a GA, what is the role of the mutation operator? In a binary-coded GA, mutation is the occasional (low-probability) alteration of a bit position, and with other codes a variety of diversity generating operators may be used. By itself, mutation induces a simple random walk through string space. When used with selection alone, the combination form a parallel, noise-tolerant hill-climbing algorithm. When used together with selection and crossover, mutation acts as both insurance policy and as a hill-climber.

Example 16.1 *4-Queen Puzzle.*

Suppose we choose to solve the problem for $N = 4$. This means that the board size is $4^2 = 16$, and the number of queens we can fit inside the board without crossing each other is 4. A configuration of 4-queens can be represented as shown in figure 16.2, using 4-digit string made of decimal numbers in the range 1-4. Each digit in a string represents the position of queen in that column. Thus all queens in the left-to-right diagonal will be represented by a string 1234.

To solve this problem we take initial populations as $[1234, 2342, 4312, 3431]$. Let us recombine by randomly choosing the crossover after digit position 2. We recombine 1, 2 and 3, 4 members of population, producing $[1242, 2334, 4331, 3412]$. When this is combined with the original population, we get $[1234, 2342, 4312, 3431, 1242, 2334, 4331, 3412]$. Next, a random mutation is applied on members 3431 and 2334, changing the third digit gives 3421, 2324. Thus new population is, $[1234, 2342, 4312, 3421, 1242, 2324, 4331, 3412]$.

The fitness of a string is proportional to the inverse of number of queens giving check in each string. for example, in the configuration in figure 16.2, total number of checks of $Q_1 - Q_4$ are: $2+3+2+1 = 8$. Similarly in configuration 1234, total number of checks are 12. This is because each queen is crossing the remaining three. These numbers in the remaining seven configurations are: $8, 4, 4, 4, 6, 8, 8$. Thus fitness functions of above population of elements is $[\frac{1}{12}, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{8}]$. Thus, if we need to keep a population of size 4, of more fitter members their fitness functions are $[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{6}]$. These further combine next time, with population size of 4, having population of $[4312, 3421, 1242, 2324]$. This sequence can go on until goal is found in one of the configuration. □

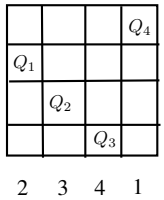

Figure 16.2: 4-Queens' Board configuration.