32002: AI (Natured Inspired Search: Sim. Ann., GA) **Spring 2014**

Lecture 30: March 25, 2014

Instructor: K.R. Chowdhary *: Professor of CS*

30.1 Nature Inspired Search Techniques

These are the search techniques based on natural phenomena. Under this we are going to discuss tow 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. 2) The well known Darwin's theory of evolution, called genetic algorithms.

30.1.1 Simulated Annealing

Annealing is process of treatment of metal or alloy by heating to a predetermined temperature, holding for a certain time, and then cooling to room temperature to improve ductility and reduce brittleness. The process annealing is carried out intermittently during the working of a piece of a metal to restore ductility lost through repeated hammering or other working. Annealing is also done for relief of internal stresses. Annealing temperature varies with metals and alloys, and with properties desired, but must be done with in a range, that prevents the growth of crystals.

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.

At each step, the SA heuristic considers some neighboring state $s^{'}$ of the current state s, and probabilistically decides between moving the system to state $s^{'}$ or staying in state s. These probabilities ultimately lead the system to move to states of lower energy. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given number of iterations have exhausted.

While discussing SA we make one change from the normal heuristic search; we attempt to minimize the function's value instead of maximizing. This 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,

$$p = e^{\frac{\Delta E}{kT}} \tag{30.1}$$

where ΔE is positive change in energy level, T is temperature, and k is Boltzmanns 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.

The rate at which system cools is called annealing schedule. If cooling occurs too fast, the physical system will will form stable regions high energy. That is, local but not global minima is reached. If however, a slower schedule is used, a uniform crystaline structure, which corresponds to a global minimum, will develop.

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 30.1 as,

$$p^{'} = e^{\frac{\Delta E}{T}}. \tag{30.2}$$

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.

The algorithm 1 describes the 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 perturbation function perturbs more when the score is worse. The idea behind this is the same as discarding all the cards in poker when your hand is hopeless and discarding one or two when your hand is hopeful. Whether this is a good idea is dependent on the value of M in Algorithm. We speculate our perturbation function allows us to use a smaller M for a shorter execution time.

The Boltzmann probability function tends to return True at higher temperature and False at lower temper-

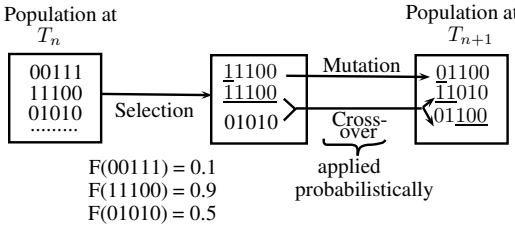

Figure 30.1: Sequence of operations in GA.

ature; thus in essence the search gradually shifts from random walk to local hill climb.

SA is a widely used heuristic to NP-complete problems that appear in real life from job-shop scheduling to ground water remediation design.

Most analysis of search algorithms usually focuses on the worst-case situation. There are relatively few discussions of the average performance of heuristic algorithms, because the analysis is usually more difficult and the nature of the appropriate average to study is not always clear. However, as the size of optimization problems increases, the worst-case analysis of a problem will become increasingly irrelevant, and the average performance of algorithms will dominate the analysis of practical applications. This large number limit is the domain of statistical mechanics, and hence of simulated annealing.

30.2 Genetic Algorithms

These search procedures, based on the mechanics of natural selection and genetics, are finding increasing application to difficult search, optimization, and machine-learning problems across a wide spectrum of human endeavor. 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 30.1 shows the sequence of operations on a population P_n and producing next population $P_{n=1}$. Operation of the genetic algorithm. A population of three individuals is shown. Each is assigned a fitness value by the function F. 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. After selection, the genetic operators are applied probabilistically; the first individual has its first bit mutated from a 1 to a 0, and crossover combines the second two individuals into two new ones. The resulting population is shown in the box labeled T_{n+1}. The algorithm 2 shows the steps for search using GA.

There are many ways to achieve effective selection, including ranking, tournament, and proportionate schemes, but the key notion is to give preference to better individuals. For example, in two-party tournament selection, pairs of strings are drawn randomly from the parental population, and the better individual places an identical copy in the mating pool. If a whole population is selected in this manner, each individual will participate in two tournaments and the best individual in the population will win both trials. The median individual will typically win one trial and the worst individual does not win at all.

Of course, for selection to function there must be some way of determining what is good. This evaluation can come from a formal objective function, or it can come from the subjective judgment of a human observer or critic. As the tournament selection example makes clear, the primary requirement is for a partial ordering.

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

The population holds representations of possible solutions. It usually has a fixed size and is a multi-set. Selection operators usually take whole population into account i.e., reproductive probabilities are relative to current generation and diversity of a population refers to the number of different fitnesses.

If we were to do nothing but selection, GAs would not be very interesting because the trajectory of populations could contain nothing but changing proportions of strings contained in the original population. In fact, if run repeatedly, selection alone is a fairly expensive way of - with high probability-filling a population with the best structure of the initial population.

30.2.1 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.

30.2.2 GA Applications

The simplest GAs are discrete, non-linear, stochastic, highly dimensional algorithms operating on problems of infinite varieties. Due to this GAs are hard to design and analyze.

The nature of problems GAs can solve are:

- GAs can solve hard problems quickly and reliably,
- GAs are easy to interface to existing simulations and models,
- GAs are extensible, and
- GAs are easy to hybridize.

Because GAs use very little problem specific information, they are remarkably easy to connect to extant application code. Many algorithms require high degree of interconnection between the solver and the objective function. For example, dynamic programming requires a stage-wise decomposition of the problem that not only limit its applicability, but can require massive rearrangement of system models and objective functions.

GAs on the other hand have clean interface, requiring no more than the ability to propose a solution and receive its evaluation.

Although there are many problems for which the genetic algorithm can evolve a good solution in reasonable time. There are also problems for which they are not suitable, such as problems in which it is important to find the exact global optimum. The domains for which one is likely to choose an adaptive method such as the genetic algorithm are precisely those about which we typically have little analytical knowledge, they are complex, noisy, or dynamic (changing over time). These characteristics make it virtually impossible to predict with certainty how well a particular algorithm will perform on a particular problem, especially if the algorithm is nondeterministic, as is the case with the genetic algorithm. In spite of this difficulty, there are fairly extensive theories about how and why genetic algorithms work in idealized settings.

Example 30.1 *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 30.2, 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 2. We recombine 1, 2 and 3, 4 members of population, producing [1242, 2334, $4331, 3442]$. When this is combined with the original population, we get $[1234, 2342, 4312, 3431, 1242, 2334, 4331, 3442]$. Next, a random mutation is applied on members 3431 and 2334, changing the third digit gives 3421, 2324. Thus new population is, $[1234, 2342, 4312, 3421, 1242, 2324, 4331, 3442]$.

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 30.2, total number of checks of $Q_1 - Q_4$ are: $2+3+2 = 7$. 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: $8, 4, 4, 4, 6, 8, 7$. Thus fitness functions of above population in the same overs of elements is $[\frac{1}{12}, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{7}]$. Thus, if we need to keep a population of size 4 of more fitter members with fitness functions $[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{6}]$ to further combine next time, with population size of 4, it is $[4312, 3421, 1242, 2324]$. This sequence can go on until goal is found in one of the configuration. □

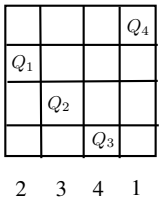

Figure 30.2: 4-Queens' Board configuration.

References

[RNO05] S. Russel and P. Norvig, "Artificial Intelligence - A Modern Approach," *2nd Edition, Pearson Education, India*, 2005. (chapter 4)