CS486: Artificial Intelligence

Homework 3 (15 pts)
Local and Metaheuristic Search
22 Sep @ 1630

Instructions

This is an individual assignment; however, *you may receive assistance and/or collaborate without penalty, so long as you properly document such assistance and/or collaboration in accordance with DAW.*

Answer the questions below and submit a hardcopy with DAW coversheet and acknowledgment statement to your instructor by the due date.

Problem 1: Simulated Annealing

a. What search algorithm does simulated annealing perform like when $-\delta/t$ approaches ∞? Briefly explain why.

Random search. Substituting $+\infty$ into the simulated annealing algorithm where $-\delta/t$ is used results in comparing x, a random value in the range $(0, 1)$, to a value that will always be greater than 1, guaranteeing that $x < \exp -\delta/t$ will be true and the randomly-selected neighbor of s_0 will be accepted. Of course, the only way this could happen is if t were a very small negative value, which is one reason why temperature values should only be positive. The other way $-\delta/t$ could approach $+\infty$ is if δ were a large negative number, but large negative values will never be used for δ in this expression because a value of $\delta < 0$ means the neighbor is, in fact, better than s_0 and is automatically accepted.

Because these conditions aren't meant to actually be reachable, the way this question was asked might have been confusing and you might have assumed it meant to ask how the algorithm performs when $-\delta/t$ approaches $-\infty$. This could be caused by an incredibly poor-quality candidate solution (large δ) or very small positive temperature. In this case, it becomes increasingly unlikely that any value for x will be less than $\exp -\delta/t$, so only better solutions will be accepted and the algorithm behaves like hill climbing (specifically, first-choice hill climbing).

b. Suppose you expect most randomly-generated solutions for a problem you intend to use simulated annealing on to be poor, so you choose τ_0 to be 0.5. Your goal is to optimize the function by *minimizing* its objective function. You randomly generate 10 initial solutions and calculate their costs ($f(s_0)$) and, for each one, randomly select one neighbor ($f(s)$) and calculate its cost, yielding the table below:

$f(s_0)$	$f(s)$
55	37
88	4
39	8
63	13
75	9
53	17
47	11
42	53
20	54
14	38

According to the formula, presented in class, for calculating a good initial temperature, what should t_0 be, based on these trial runs?

The average of $f(s) - f(s_0)$ for the ten pairs of values is -25.2.

$t_0 = -(\overline{cost_{expected}})/\ln \tau_0$
$t_0 = -(-25.2)/\ln(0.5)$ $t_0 \approx -36.4$

If you use that value, how will your simulated annealing algorithm behave?

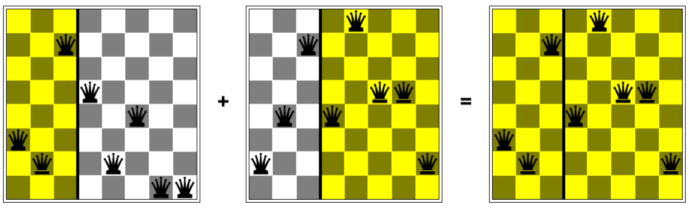
Random search again.

Explain what is going on here and why.

In the trial runs performed, almost all of the neighbors of the randomly-chosen solutions are improvements resulting in a recommendation of a negative initial temperature which is problematic for simulated annealing (see above). This result should tell you that this problem may not be a good problem for applying metaheuristics.

Problem 2: Genetic Algorithms

Consider the n-queens crossover operation example, below, from the slide at the end of the lesson on genetic algorithms.



As shown, the encoding in use allows invalid configurations to exist (i.e., queens are threatened in both the parents and the child). We aren't given the encoding in use, but suppose it were a vector of integers such that each position of the vector represented a column of the board, and the value in the vector was an integer in the range $[0, 1, \ldots, 7]$ indicating the row, numbered from top to bottom, that the queen in that column was in. Then we would have candidate solutions as follows:

$$\begin{aligned}\text{Left Parent} &= \langle 5, 6, 1, 3, 6, 4, 7, 7\rangle \\ \text{Right Parent} &= \langle 6, 4, 1, 4, 0, 3, 3, 6\rangle \\ \text{Child (shown)} &= \langle 5, 6, 1, 4, 0, 3, 3, 6\rangle\end{aligned}$$

a. "Fix" the two parents using the following algorithm:
For each column from left to right, if the queen in that column shares a row with another queen in a lower-numbered column, move the queen in the higher-numbered column to the lowest-numbered valid row (i.e., the first row that doesn't have a queen in a lower-valued column).

   For example, in Left Parent, the first queen you would have to move would be the one in column 4 (columns numbered starting at 0) because she shares a row with the queen in column 1. The lowest-numbered valid row for the queen in column 4 is row 0.

   What are the new encodings for the two parents after the fix?

   Left Parent = $\langle 5, 6, 1, 3, 0, 4, 7, 2\rangle$
   Right Parent = $\langle 6, 4, 1, 2, 0, 3, 5, 7\rangle$

b. The single-point crossover operation shown in the example does not work if we want to ensure that all configurations in the population remain valid. What is an appropriate crossover technique presented in class that would work (in all cases) for this encoding scheme?

   This problem requires a permutation crossover. One technique we covered in class is uniform order-based crossover. Edge recombination is another technique that is listed on the slide, but we didn't cover it in detail. A good reference, for those interested is at `http://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/EdgeRecombinationCrossoverOperator.aspx`. Note: the graphics on the slide do not illustrate edge recombination, they illustrate a mutation technique that will return a new, valid permutation.

c. Apply your chosen crossover technique to the two parents to generate their two children. What are the resulting encodings of the children? If your technique requires randomly-generated values, make something up and tell me what value(s) you used.

Using uniform order-based crossover:

Let the random template string be: $\langle 11010110 \rangle$.

Blue indicates values that will not change position in the crossover.

$\langle 5, 6, 1, 3, 0, 4, 7, 2 \rangle \Rightarrow \langle 6, 4, 1, 3, 0, 5, 7, 2 \rangle$

$\langle 6, 4, 1, 2, 0, 3, 5, 7 \rangle \Rightarrow \langle 6, 4, 1, 2, 0, 3, 5, 7 \rangle$

For this random template string, notice that there is no change to the right parent since the positions to be re-ordered appear in the same order in both parents!

d. What would be an appropriate mutation technique, presented in class, for this problem?

Reversing a substring in the individual being mutated is shown on the lesson slides. You could also swap two. Random shuffling of the entire permutation or even a substring would probably lose too much information about the value of the state.