Sched088
CSCI 5511 001
Oct 5, 2021

Assignment 2 Analysis

Instructions:

Generate a large number of random initial states and run each of your algorithms on those states. Calculate the average number of steps overall for each of the algorithms to find a solution. (If an algorithm fails to find a solution, don't count that.) Writeup your results and put them in a file named report.txt (or report.pdf)

Begin your report with brief instructions about how to run your different functions. Include a short discussion about any parameters that you needed to set to get the results that you are reporting on (i.e. how did you choose your temperature and schedule, etc.) and also include any comments about difficulties encountered.

Your report can be short - a page or two should be enough.

Analysis:

Some guidance on how to run the algorithms is commented into the code. Critically only one algorithm type can be run at a time. This is controlled by which run-type variable is set to true. The run-type variables are on lines 19-21. When the variable ‘steepest_ascent’ is set to True the function ‘hillclimb_sa’ will be called. When the variable ‘first_choice’ is set to True the function ‘hillclimb_fc’ will be called. Finally, when the variable ‘annealing’ is set to True the function ‘sim_anneal’ will be called.

All data below was gathered from the variables set as shown in table 1.

Variable Name	Value	Definition
n	8	This is the nxn dimension of the board and is also equivalent to the number of queens.
run_count	5000	This is the number of times a random board is generated and the ‘True’ algorithm is run
iterations	1000, 3000	This is the number of times hillclimb_fc and sim_anneal will loop looking for a better solution before breaking out of the loop and returning ‘no solution found’

Table 1: Variable names and settings used for analysis

Findings:

hillclimb_sa:

Percent Solutions: 15.02%
Average Steps for Solution: 2.15

hillclimb_fc:

This algorithm can be limited by the number of iterations allowed. This is shown by the first run setting where the number of iterations was set to 100. There is a 5% drop in the percent solutions output. Because only one queen is moved at a time and there are no random resets there is a limit to the percentage of solutions that will be found as the number of iterations approaches infinity. This is displayed by how close the percent solutions are for both 1000 and 3000 iterations.

> Iterations: 100
Percent Solutions: 9.3%
Average Steps for Solution: 5.86
> Iterations: 1000
Percent Solutions: 14.1%
Average Steps for Solution: 6.07
> Iterations: 3000
Percent Solutions: 14.2%
Average Steps for Solution: 6.07

sim_anneal:

This algorithm can be tuned for performance by changing the number of iterations, the initial temperature, and the annealing rate. The impact of the number of iterations is at a higher value than the hillclimb_fc function. This is shown by looking at the difference between the 1000 and 3000 iterations below. Both were run with temperature = 500 and anneal rate = 0.99

> Iterations: 1000
Percent Solutions: 21.1%
Average Steps for Solution: 1448
> Iterations: 3000
Percent Solutions: 92.0%
Average Steps for Solution: 1652

I chose a relatively high temperature and low annealing rate (closer to 1.0 the slower the temp decrease) as this would, with enough iterations, exhibit behavior similar to both first-choice hill climbing (when temperature ⇒ 0) and random walk (when temperature is ⇒ infinity). By leveraging the idea of random walk we can increase the likelihood of identifying the global maximum out of the many local maximums.

Additional Comments:

Having not programmed with Classes until this course I had some difficulties catching up on how these are used. I appreciated the parts of the logic that could be applied from the steepest ascent function to the others.

I had some difficulties at first with how the solution function called the annealing function which is why it is handled differently than for how the solution function calls the hill climbing functions. I am interested in tuning the parameters for the annealing function because with the current structure I believe that it is pretty inefficient. I did some reading on temperature and annealing settings for this function and was interested to see that sometimes it seems better to have a very quickly cooling function. This is not the approach I took.