CSE P 571 Programming Assignment 1

Alan Ludwig

# Algorithm 1 and Algorithm 2 Choices

Algorithm 1 was defined for us to be “Greedy Hill Climbing with Random Start”. What that means is that a valid set of bids is chosen at random. Then, for some small neighborhood around the current location the value is examined and the neighboring set of bids corresponding to the greatest value is chosen as the next step. This process is repeated until no higher value can be found. For this assignment, the heart of the algorithm is the choice of the neighborhood function. For my implementation of algorithm 1 I chose my neighborhood as “all the valid solutions that can be reached by removing one bid from the current solution and replacing it with another available bid”. The code for finding non-conflicting bids was taken directly from the sample code used to find a random initial solution.

For algorithm 2 I first chose to optimize the existing code. The heart of the optimization was to replace the set operation “intersection” for the Java HashSet with a “disjoint” operation. This change alone produced an approximately 10x speedup. The next optimization was to change the order of the comparisons so that the match for company was checked first and only if the company was not a match. This was a simple integer comparison and much cheaper than the more costly set operation. Making this next optimization resulted in an approximately 2x speedup in the code. Of all the changes I made to the code, this simple optimization made more difference in the quality of the results than any of the tweaking I did to the AI algorithms. However, I did also change the AI algorithms. After trying a few different approaches I found that making adjustments to my neighborhood function had the greatest impact. In addition to the notion of a neighborhood being “the replacement of a single bid with another bid” I expanded the neighborhood function to look for two other cases. The first was to consider looking for adding an additional bid if a valid bid could be added to the set. The second was to consider the case where a single bid could be replaced by two bids that together had a greater value than the first. If I have more time I would have looked next at adding code that would replace two existing bids with another bid with a greater value than the two bids being replaced.

# Experimental Results

For my experiment I ran each algorithm over each of the benchmark sets and compared the results. Overall algorithm2 had mixed results over algorithm 1. Algorithm 1 proved sufficient to find the best possible results for five of the benchmark problems: 1, 2, 8, 11, and 17. I assume that it found the best because every run had the same answer. Algorithm 2 was in fact able to find the maximum for 7 different problems: 1, 2, 6, 8, 9, 11, 17, and 20. For benchmark 3, 7, 10, 12, 14, 16, 18, algorithm 2 outperformed algorithm 1. And interestingly, for benchmark 4, 5, 13, 15 and 19 algorithm1 outperformed algorithm2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Alg1 | | Alg2 | |
| File | Regions | Bids | Value | Iterations | Value | iteration |
| 1 | 77 | 29 | $ 1,834,885.00 | 214,234.00 | $ 1,834,885.00 | 281,695.10 |
| 2 | 53 | 56 | $ 2,468,164.00 | 213,967.90 | $ 2,468,164.00 | 293,171.40 |
| 3 | 214 | 567 | $ 16,152,356.40 | 1,819.40 | $ 16,218,169.10 | 764.50 |
| 4 | 292 | 1052 | $ 23,971,863.50 | 525.80 | $ 23,736,856.00 | 95.80 |
| 5 | 237 | 616 | $ 13,236,703.20 | 695.90 | $ 13,192,202.60 | 193.70 |
| 6 | 270 | 1638 | $ 15,073,058.80 | 378.80 | $ 15,278,555.00 | 1,138.20 |
| 7 | 151 | 781 | $ 14,289,466.60 | 2,745.00 | $ 14,394,380.30 | 975.00 |
| 8 | 86 | 73 | $ 3,591,175.00 | 77,557.70 | $ 3,591,175.00 | 79,118.90 |
| 9 | 202 | 184 | $ 9,014,063.30 | 16,307.20 | $ 9,071,379.00 | 27,625.80 |
| 10 | 46 | 327 | $ 5,503,741.00 | 16,139.90 | $ 5,563,480.50 | 17,010.90 |
| 11 | 66 | 1405 | $ 6,726,263.00 | 4776.20 | $ 6,726,263.00 | 6,051.40 |
| 12 | 336 | 1442 | $ 16,703,432.90 | 1243.80 | $ 16,974,906.90 | 2,676.20 |
| 13 | 334 | 1978 | $ 17,693,302.70 | 939.80 | $ 17,470,113.60 | 156.80 |
| 14 | 237 | 1840 | $ 14,258,551.60 | 1198.20 | $ 14,401,872.90 | 1,313.60 |
| 15 | 85 | 332 | $ 6,955,489.60 | 21459.50 | $ 6,950,680.50 | 12,591.20 |
| 16 | 92 | 221 | $ 8,801,520.60 | 36374.80 | $ 8,855,352.70 | 17,277.90 |
| 17 | 344 | 1081 | $ 108,672,208.00 | 1588.90 | $ 108,672,208.00 | 5,641.25 |
| 18 | 298 | 440 | $ 18,420,570.60 | 8403.50 | $ 18,841,051.88 | 11,410.38 |
| 19 | 366 | 194 | $ 12,568,792.50 | 24995.20 | $ 12,549,112.38 | 11,250.88 |
| 20 | 358 | 1590 | $ 23,014,116.09 | 848.55 | $ 23,077,114.00 | 4,345.75 |

# Analysis of what worked and what didn’t

Based on the information above, algorithm 2 sounds like a big improvement over algorithm 1. However, a closer look at the data reveals a different story. For example in each case that algorithm 2 found the optimal solution in all 10 runs algorithms 1 also found the maximum at least once in the 10 runs. It is also true that in every one of these instances algorithm 2 was able to able to complete more iterations in the allotted time than algorithm 1. For the purpose of this discussion a single iteration begins with a random start and ends when the neighborhood function can’t find a higher value in the neighborhood. So, more iteration means more random starts. Given this, it is possible that all the improvements in the performance of Algorithm 2 came from the ability to complete more iterations, rather than being attributable to a stronger algorithm

And, it is also true that in every one of the cases where algorithm 1 performed better than algorithm 2 that algorithm 1 also completed more iteration. In the end, adding the extra code to make for more sophisticated neighborhood functions resulted in less iteration than the simpler algorithm and thereby gave fewer opportunities to find a good random start location. The more sophisticated neighborhood function appeared to scale less well for large numbers of bids.