**CMSC 170 MACHINE PROBLEM 2 -** *Constrained Problems*

**Group Members:**

**PART 0. CONSTRAINTS**  **20 points**

1. (**2 pts**) Describe how the 8 solutions for magic square(N=3) are *similar*, and how they are *different*.

N = 3

2 7 6

9 5 1

4 3 8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 9 4

7 5 3

6 1 8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4 3 8

9 5 1

2 7 6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4 9 2

3 5 7

8 1 6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6 1 8

7 5 3

2 9 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6 7 2

1 5 9

8 3 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

8 1 6

3 5 7

4 9 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

8 3 4

1 5 9

6 7 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

There are 4 pairs which are partnered because of similarity in number arrangements but differ on their orientation. If orientation probably matters, there are only 4 solutions for magic\_square(N=3).

1. (**1 pt**) How many solutions are there for magic series(N=5)?

There are 0 solutions found for magic\_series(N=5).

1. (**2 pts**) Find all solutions for magic series(N=6). How many iterations did it take?

There are 823,543 iterations for magic\_series(N=6) with solution:

N = 6

Index 0 1 2 3 4 5 6

Series 3 2 1 1 0 0 0

Count 3 2 1 1 0 0 0

1. (**2 pts**) What is the total weight and total value of the optimal solution for knapsack(test\_case=1)?

The total weight and total value of the optimal solution for knapsack(test\_case=1) are 15 and P7 respectively.

1. (**2 pts**) How many iterations and solutions were found for knapsack(test\_case=3)?

There are 1,048,576 iterations and 584,501 solutions found for knapsack(test\_case=3).

1. (**3 pts**) How many solutions were found for vertex\_cover(test\_case=3,4,5)?

There are 35 solutions found for vertex\_cover(test\_case=3).

There are 76 solutions found for vertex\_cover(test\_case=4).

There are 235 solutions found for vertex\_cover(test\_case=5).

1. (**2 pts**) What are the vertices in the optimal solution for vertex\_cover(test\_case=3)?

The vertices in the optimal solution for vertex\_cover(test\_case=3) are [‘B’, ‘D’, ‘E’, ‘F’, ‘H’].

1. (**4 pts**) Fill in the table to compare the no. of iterations and running time when using all possible combinations of values vs. using permutations of values for plants and magic square(N=3).

|  |  |  |
| --- | --- | --- |
|  | No. of Iterations | Running Time |
| Plants, Combination | 3,125 | 1 second |
| Plants, Permutation | 120 | 0 seconds |
| Magic Square(3), Combination |  |  |
| Magic Square(3), Permutation | 362,880 | 208 seconds |

*Note:* For (magic\_square, comb.), just compute the no. of iterations and estimate the running time.

1. (**2 pts**) Discuss the results above briefly.

**PART 1. BACKTRACKING 30 points**

1. (**5 pts**) Fill in the table to compare the number of iterations when using brute force vs backtracking.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Problem** | **solution\_limit = 0** | | **solution\_limit = 1** | |
| **Brute-Force** | **Backtracking** | **Brute-Force** | **Backtracking** |
| plants | 120 | 331 | 73 | 91 |
| magic\_square(3) | 362,880 | 21,331 | 69,075 | 3,691 |
| magic\_series(4) | 3,125 | 6 | 1,426 | 6 |
| knapsack(2) | 32,768 |  | 1 |  |
| vertex\_cover(4) | 1,024 |  | 416 |  |

1. (**3 pts**) Discuss: (1) which situations did brute-force have more iterations than backtracking? (2) which situations did brute-force have fewer iterations than backtracking and why did this happen?

1. (**2 pts**) In solution\_limit = 1, did any solver find the optimal solutions for knapsack and vertex cover? Explain why this happened.
2. (**6 pts**)Fill in the table to compare the number of iterations when using backtracking with and without filtering. Quantify the improvement caused by filtering.

*Example:* From 500 to 100, the improvement is 80% (reduced by 400, 400 / 500 = 0.8)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **solution\_limit = 0** | | | **solution\_limit = 1** | | |
|  | **No Filtering** | **With Filtering** | **Improvement** | **No Filtering** | **With Filtering** | **Improvement** |
| plants | 331 | 68 | 79% | 91 | 24 | 74% |
| einstein | 2,363 | 531 | 78% | 210 | 66 | 69% |
| magic\_square(3) | 21,331 | 2,378 | 89% | 3,691 | 421 | 89% |
| magic\_series(4) | 6 | 6 | 0% | 6 | 6 | 0% |
| knapsack(2) |  |  |  |  |  |  |
| vertex\_cover(4) |  |  |  |  |  |  |

1. (**2 pts**) Discuss the importance of filtering based on the results above.
2. (**6 pts**) Fill in the table to compare the no. of iterations when using backtracking with filtering (BT + F) and with filtering + ordering (BT + F + O). Quantify the improvement caused by ordering.

*Example:* From 500 to 100, the improvement is 80% (reduced by 400, 400 / 500 = 0.8)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **solution\_limit = 0** | | | **solution\_limit = 1** | | |
|  | **BT + F** | **BT + F + O** | **Improvement** | **BT + F** | **BT + F + O** | **Improvement** |
| plants |  |  |  |  |  |  |
| einstein |  |  |  |  |  |  |
| magic\_square(3) |  |  |  |  |  |  |
| magic\_series(4) |  |  |  |  |  |  |
| knapsack(2) |  |  |  |  |  |  |
| vertex\_cover(4) |  |  |  |  |  |  |

1. (**2 pts**) Discuss the importance of ordering based on the results above.
2. (**2 pts**) In solution\_limit = 0, are there any problems which showed no improvement even after ordering was applied? Explain why this happened to these problems.
3. (**2 pts**) Compare the no. of iterations for brute force solver vs. backtracking solver with filtering and ordering. Discuss which solver is better.

**PART 2. LOCAL SEARCH 30 points**

1. (**5 pts**) Fill in the table to compare the total number of iterations of the different neighborhoods for each problem. Color the cell red if the solver didn’t find a feasible solution (score > 0).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **change1** | **change2** | **swap2** | **min-conflict** | **max-min-conflict** |
| plants | 49 | 3 | 231 | 48 | 53 |
| einstein | 957 |  | 702 | 2,020 | 1,754 |
| magic\_square(3) | 131 | 112 | 571 | 2,020 | 1,429 |
| magic\_series(4) | 86 | 42 | 600 | 600 | 600 |

1. (**2 pts**) From the results above, which neighborhood is the best for these problems in general?
2. (**3 pts**) Fill in the table to compare the total number of iterations and best score of the different neighborhoods for each problem.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **change1** | | **change2** | | **swap2** | |
|  | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| knapsack(3) |  |  |  |  |  |  |
| vertex\_cover(5) |  |  |  |  |  |  |

1. (**2 pts**) From the results above, which neighborhood is the best for knapsack and vertex cover, in general? Compare the best score obtained, and use the number of iterations as tie-breaker.

1. (**6 pts**) Fill in the table to compare the number of iterations and best score of different hill climbing variants for each problem.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Hill Climbing** | | **Hill Walking** | | **Random Walking** | |
|  | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| plants |  |  |  |  |  |  |
| einstein |  |  |  |  |  |  |
| magic\_square(3) |  |  |  |  |  |  |
| magic\_series(4) |  |  |  |  |  |  |
| knapsack(3) |  |  |  |  |  |  |
| vertex\_cover(5) |  |  |  |  |  |  |

1. (**2 pts**) For the constraint satisfaction problems, which hill climbing variant worked best? Explain.
2. (**2 pts**) For the optimization problems, which hill climbing variant worked best? Explain.
3. (**1 pt**) From the results above, which was the worst hill climbing variant in general? Explain.
4. (**4 pts**) Fill in the table to compare the number of iterations and best score of different tabu tenures. Which was the best tabu tenure for the knapsack problems? For vertex cover?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **tenure=0** | | **tenure=3** | | **tenure=5** | | **tenure=7** | |
|  | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| knapsack(2) |  |  |  |  |  |  |  |  |
| knapsack(3) |  |  |  |  |  |  |  |  |
| vertex\_cover(3) |  |  |  |  |  |  |  |  |
| vertex\_cover(5) |  |  |  |  |  |  |  |  |

1. (**3 pts**) Based on the results, discuss the importance of tabu search and choosing the right tenure.

**PART 3. ADVANCED LOCAL SEARCH 20 points**

1. (**4 pts**) Fill in the table to compare the iterations and best score of different neighbor generators.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **change1** | | **change2** | | **swap2** | | **custom** | |
|  | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| maxone(16) |  |  |  |  |  |  |  |  |
| knapsack(3) |  |  |  |  |  |  |  |  |
| vertex\_cover(5) |  |  |  |  |  |  |  |  |

1. (**2 pts**) For each problem above, which neighbor generators performed best? Discuss the results.
2. (**2 pts**) Did the custom neighbor generators perform well in general? Explain why or why not.
3. (**6 pts**) Fill in the table to compare the number of terations and best score of the different configs tested: stochastic local search (SLS) vs simulated annealing (SA), different alpha values for SA,

neighbor generators, and hill walk vs hill climb.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **KNAPSACK(99)** | **Stochastic LS** | | **SA, alpha = 0.5** | | **SA, alpha = 0.75** | | **SA, alpha = 0.95** | |
| **Hill Walk** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| change1 |  |  |  |  |  |  |  |  |
| change2 |  |  |  |  |  |  |  |  |
| swap2 |  |  |  |  |  |  |  |  |
| custom |  |  |  |  |  |  |  |  |
|  | **Stochastic LS** | | **SA, alpha = 0.5** | | **SA, alpha = 0.75** | | **SA, alpha = 0.95** | |
| **Hill Climb** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| change1 |  |  |  |  |  |  |  |  |
| change2 |  |  |  |  |  |  |  |  |
| swap2 |  |  |  |  |  |  |  |  |
| custom |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **VERTEX COVER(99)** | **Stochastic LS** | | **SA, alpha = 0.5** | | **SA, alpha = 0.75** | | **SA, alpha = 0.95** | |
| **Hill Walk** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| change1 |  |  |  |  |  |  |  |  |
| change2 |  |  |  |  |  |  |  |  |
| swap2 |  |  |  |  |  |  |  |  |
| custom |  |  |  |  |  |  |  |  |
|  | **Stochastic LS** | | **SA, alpha = 0.5** | | **SA, alpha = 0.75** | | **SA, alpha = 0.95** | |
| **Hill Climb** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** | **Iterations** | **Best Score** |
| change1 |  |  |  |  |  |  |  |  |
| change2 |  |  |  |  |  |  |  |  |
| swap2 |  |  |  |  |  |  |  |  |
| custom |  |  |  |  |  |  |  |  |

1. (**2 pts**) Which alpha values worked best for knapsack and vertex cover, in general? Discuss.
2. (**4 pts**) Which configuration produced the best score for stochastic local search and for simulated annealing for knapsack and vertex cover?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stochastic Local Search** | **Best Score** | **Neighbor Generator** | **Hill Walk or Climb** |  |
| Knapsack(99) |  |  |  |  |
| Vertex Cover(99) |  |  |  |  |
| **Simulated Annealing** | **Best Score** | **Neighbor Generator** | **Hill Walk or Climb** | **Alpha** |
| Knapsack(99) |  |  |  |  |
| Vertex Cover(99) |  |  |  |  |

**PART 4. GENETIC ALGORITHMS 15 points**

1. (**10 pts**)Which configuration produced the best solution for knapsack and vertex cover?

|  |  |  |
| --- | --- | --- |
|  | **KNAPSACK(99)** | **VERTEX COVER(99)** |
| Best Score |  |  |
| Population Model |  |  |
| Population Size |  |  |
| Parent Selection |  |  |
| Max Parent Similarity |  |  |
| Crossover |  |  |
| Crossover Probability |  |  |
| Mutation |  |  |
| Mutation Probability |  |  |

1. (**5 pts**)Discuss why the best configurations (population model, selection, crossover, mutation, etc) for knapsack and vertex cover performed well.