Heuristic Analysis

Below table shows the result of experiments run for Problem 1, 2 and 3 using various search algorithms and heuristics. A highlighted row suggests the best algorithm for the particular problem.

- **Problem 1 :**

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time (seconds)
breadth_first_search	43	56	180	6	0.061871709
depth_first_graph_search	12	13	48	12	0.017733145
greedy_best_first_graph_search with h_1	7	9	28	6	0.009315114
astar_search with h_ignore_preconditions	41	43	170	6	0.070290673
astar_search with h_pg_levelsum	11	13	50	6	3.735148089

- **Problem 2**

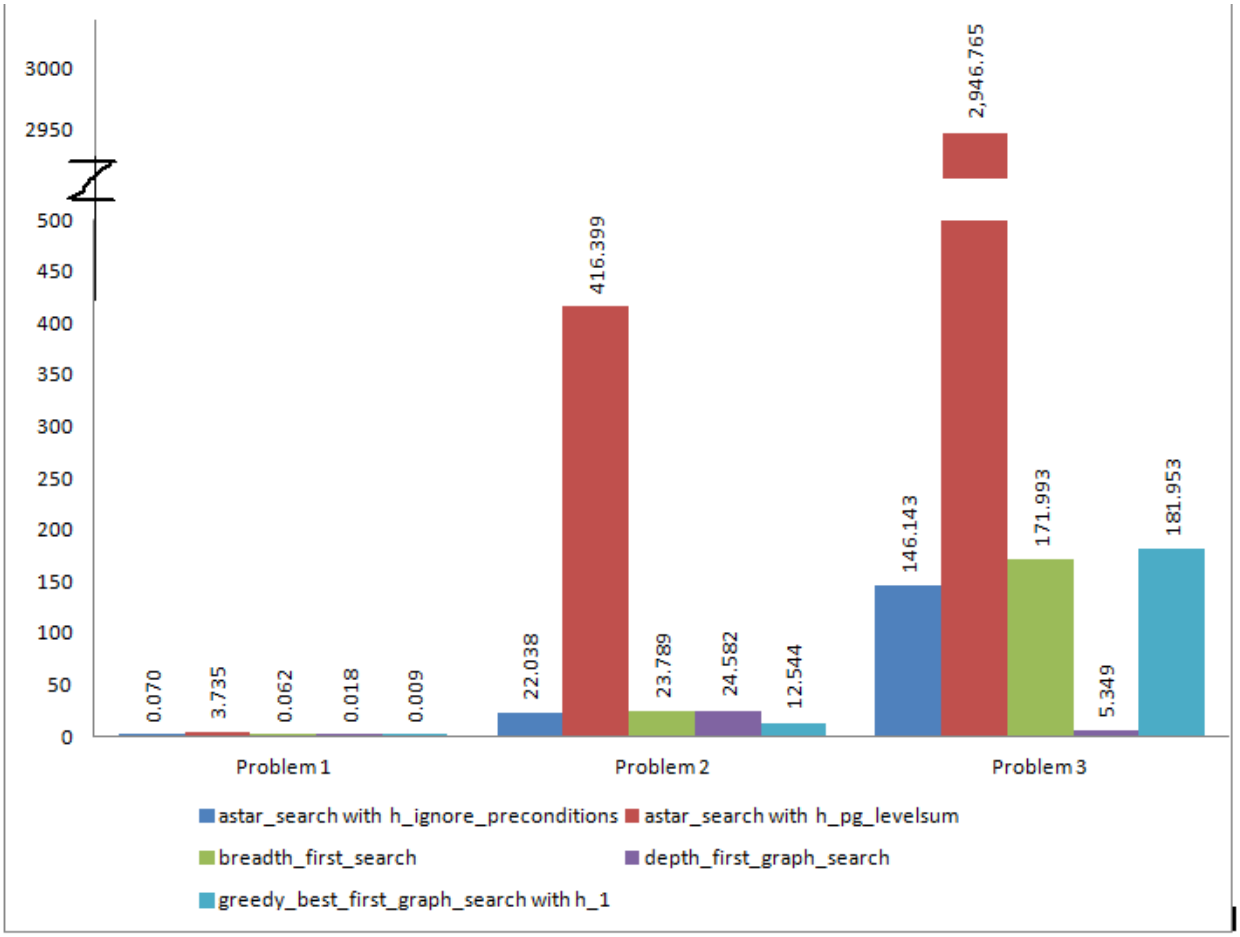
Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time (seconds)
breadth_first_search	3343	4609	30509	9	23.78861763
depth_first_graph_search	1669	1670	14863	1444	24.58232153
greedy_best_first_graph_search with h_1	998	1000	8982	17	12.5444117
astar_search with h_ignore_preconditions	1506	1508	13820	9	22.03829516
astar_search with h_pg_levelsum	86	88	841	9	416.399386

- **Problem 3**

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time (seconds)
breadth_first_search	14663	18098	129631	12	171.9934452
depth_first_graph_search	592	593	4927	571	5.348756515
greedy_best_first_graph_search with h_1	5578	5580	49150	22	181.9527718
astar_search with h_ignore_preconditions	5118	5120	45650	12	146.1432913
astar_search with h_pg_levelsum	408	410	3758	12	2946.76504

Performance Analysis:

Following bar chart is to provide a visual description on performance of various search algorithms for Problem1, 2 and 3. Y-Axis represents time elapsed in seconds.



- **Breadth First Search:**

  - **Node Expanded**
    Breadth First search is an uninformed search because it does not know how far the program is from the goal. If the successors are not the goal, then BFS expands each of those nodes, and this loop continues until the goal node is reached. So the number of node expanded grows exponentially. That's why the number of node expanded, new nodes and goal tested are higher in the above experiment.

  - **Optimality**
    Breadth First search is complete, means it would find the goal if the goal exists. BFS may be optimal if the path cost is same in each direction.

  - **Time Elapsed**
    Since BFS is a uninformed search and also there is not any heuristic involved in choosing a node out of all possible nodes at a particular level, the breadth first search runs fast.

- **Depth First Graph Search:**

    - **Node Expanded**
      Depth First Graph Search is also an uninformed search method. It starts from the root node and explores as far as possible along each branch before backtracking. Depth first search usually requires expanding lesser number of nodes than BFS. This is mainly because DFS does not always expand out every single node at each depth. The above experiment result for DFS shows the same.

    - **Optimality**
      Depth first search is not optimal as DFS returns a path as soon as the goal is reached; however, this path is not always the shortest path since the result is achieved by going down the long path to reach the goal. We can see it in the above experiment which shows that path length is largest for DFS searches.
      DFS is also not complete since it does not guarantee to return result even if it exists, since DFS may be stuck in a search loop.

    - **Time Elapsed :**
      Similar to BFS, the DFS also do not have any heuristic evaluation to run at each node so the search runs fast. Time taken depends on how much depth algorithm has to search through to reach the goal.

- **Greedy Best First Search :**

    - **Node Expanded:**
      Greedy Best First Search is also an uninformed search but it expands the fewer nodes as it expands the node that is estimated to be closest to goal. However this decision is never revised, eventually accepting suboptimal solutions. The plan length in above experiment shows the suboptimal plan.

    - **Optimality:**
      Greedy Best First Search is not optimal due to the above reason and not complete since it may stick in a loop and might never find the goal even if that exists.

    - **Time Elapsed:**
      The algorithm expands lower number of nodes, but as it perform a heuristic process to estimate distance of each node and then choose a node closest to the goal, the overall timings are similar to BFS search

- **A* Ignore Precondition**

    - **Node Expanded:**
      A* is an informed search in which a heuristic gives an idea that how far the search is from its goal. It uses the A* search with a heuristic which ignore precondition of an action and consider number of unsatisfied goals as the minimum number of actions needed to take. This is a simple heuristic used to choose the most promising node first to expand.

    - **Optimality:**
      The solution achieved is complete and optimal.

    - **Time Elapsed:**
      The heuristic to select next node to expand is very simple. It uses count of unsatisfied goal and uses it to choose next promising node. The search is faster and it also takes less amount of memory

- **A* Level Sum**

    - **Node Expanded:**
      It avoids expanding paths that are already expensive, and choose to expand most promising node first. That's why the number of node expanded, new nodes and goal tested are least in the above experiment result.

    - **Optimality:**
      The solution is complete and optimal.

    - **Time Elapsed**
      The time complexity increases exponential with path length. For each node we need to calculate the level sum, which is a complex calculation. So the time taken to find the solution becomes very high if the path length is longer.

Best Search Algorithms Analysis:

Best algorithm for a particular problem is highlighted in the table shown in the first page of this document.

- **Problem 1:**
  For first problem, “greedy_best_first_graph_search with h_1“ is fastest and expand least number of nodes. So this can be the best algorithm for Problem 1.

- **Problem 2:**
  “astar_search with h_ignore_preconditions” is faster than “astar_search with h_pg_levelsum”. It expand lesser number of nodes in comparison to BFS and DFS and it is also optimal in comparison to “greedy_best_first_graph_search with h_1” and DFS. So for this problem “astar_search with h_ignore_preconditions” is the better choice.

- **Problem 3:**

  Similar to the Problem 2, “astar_search with h_ignore_preconditions” is better in optimality and faster than others. So for this problem also “astar_search with h_ignore_preconditions” would work better.

- **Overall:**
  For Problem 2 and 3, “astar_search with h_ignore_preconditions” is cleary the best search algorithm to select. For Problem 1 also, “astar_search with h_ignore_preconditions” would be a good option since it guarantees optimality. Apart from it, the algorithm is an informed search algorithm with custom heuristic, which can give a significant advantage in terms of speed and memory usage. So overall it would be better to use “astar_search with h_ignore_preconditions” for any of these problems.