International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 07 | July - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Hill-Climbing Search Artificial Intelligence

Dr. Suneel Pappala, Associate Professor,
Information Technology,
Lords Institute of Engineering & Technology,
Hyderabad Osmania University, Telangana, INDIA.

Abstract:
This paper presents a simple optimization algorithm. The algorithm will keep moving towards the direction of increasing (or decreasing) values of the objective function until it reaches a local maximum or minimum. hill climbing may not always find the global maximum or minimum. Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. The state-space landscape is a graphical representation of the hill-climbing algorithm. Stochastic hill climbing does not examine for all its neighbor before moving. Hill climbing is a simple optimization algorithm used in Artificial Intelligence (AI) to find the best possible solution for a given problem.

Keywords: Optimization, Hill climbing Search, Evaluation, Termination, heuristic function**.**

Hill climbing is a simple optimization algorithm used to find the maximum or minimum of a mathematical function by iteratively making small adjustments to the solution. The algorithm is named after the metaphor of climbing a hill, where the goal is to reach the highest point (maximum) or lowest point (minimum) on the landscape of the function.
The basic idea behind hill climbing is straightforward:
Initialization: Start with a random or arbitrary solution as the current best solution.
Evaluation: Evaluate the current solution by calculating the value of the objective function (the function to be optimized).
Neighborhood Generation: Generate neighboring solutions by making small adjustments or perturbations to the current solution. These adjustments are typically made in a specific direction or within a certain range.
Selection: Choose the best solution among the current solution and its neighbors based on the value of the objective function.
Termination: Repeat steps 3 and 4 until a stopping condition is met. This condition can be a maximum number of iterations, reaching a predefined threshold for the objective function value, or some other criterion.
The algorithm will keep moving towards the direction of increasing (or decreasing) values of the objective function until it reaches a local maximum or minimum. The final result will depend on the initial starting point and the neighborhood exploration strategy. It's important to note that hill climbing may not always find the global maximum or minimum, especially when the landscape of the objective function is complex and contains multiple peaks and valleys. It is sensitive to the initial starting point and can be easily trapped in local optima. To overcome these limitations, various extensions and modifications to the basic hill climbing algorithm have been proposed, such as simulated annealing, genetic algorithms, and particle swarm optimization, which use different strategies for exploration and exploitation to search the solution space more effectively. search agent moves in the direction of increasing value of heuristic function i.e., uphill. Only the current state and the next best state are kept active and other states are terminated. If the next state is a goal state, search stop otherwise the search continues until it finds a goal state or reaches a peak where no neighbor

has a higher value. Hill- climbing does not look ahead, beyond the immediate neighbors of the current state and therefore, may stop at local maxima, or a ridge or a plateau. A general approach of hill climbing search is

Procedure Hill climbing

```
Procedure Hill_climbing

1  s ← {root}
2  while s ≠ Φ do
3  begin
4          p← Pop(s);
5          if (p= = goal)
6                  stop;
7          d 1← descendent (p);
8          for each element d1 ∈ d in descending order of f(d) do
9          begin
10                 if NOT-VISITED (d1) then
11                         Push(s, d1);
12         end
13 end
End_ Hill_climbing
```

Hill-climbing often gets stuck at:

(i) Local maxima - A local maxima are a peak that is higher than each of its neighboring states. Hill climbing that reaches the vicinity of a local maximum and moves upward towards the peak, but then can stuck there itself finding nowhere to go.

(ii) Ridges - It is a sequence of local maxima, which poses difficulty in choosing the best maxima for this technique.

(iii) Plateau - A plateau is an area of search space where the evaluation function is flat. A hill-climbing search might be unable to find its way off the plateau.

Hill Climbing Algorithm for Artificial Intelligent:

1. Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
2. Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
3. It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
4. A node of hill climbing algorithm has two components which are state and value.
5. Hill Climbing is mostly used when a good heuristic is available.
6. In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Features of Hill Climbing:

Following are some main features of Hill Climbing Algorithm:

1. **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
2. **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
3. **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

State-Space Diagram for Hill Climbing:

The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.
On Y-axis we have taken the function which can be an objective function or cost function, and state-space on the x-axis. If the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum. If the function of Y-axis is Objective function, then the goal of the search is to find the global maximum and local maximum.
Different regions for state space landscape:
Local Maximum: Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.
Global Maximum: Global maximum is the best possible state of state space landscape. It has the highest value of objective function.
Current state: It is a state in a landscape diagram where an agent is currently present.
Flat local maximum: It is a flat space in the landscape where all the neighbor states of current states have the same value.
Types of Hill Climbing Algorithms:

1. Simple hill Climbing
2. Steepest-Ascent hill-climbing
3. Stochastic hill Climbing

Simple hill Climbing: Simple hill climbing is the simplest way to implement a hill climbing algorithm. It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state.

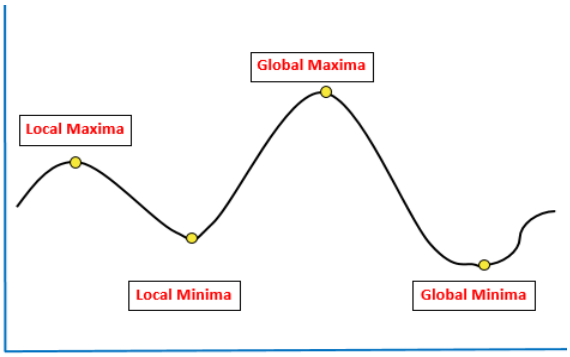

Fig: Simple Hill Climbing

This algorithm has the following features:

- Less time consuming
- Less optimal solution and the solution is not guaranteed.

Algorithm for Simple hill Climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:
    - If it is goal state, then return success and quit.
    - Else if it is better than the current state then assign new state as a current state.

- Else if not better than the current state, then return to step2.

- **Step 5:** Exit.

Steepest -Ascent Hill Climbing:

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state.

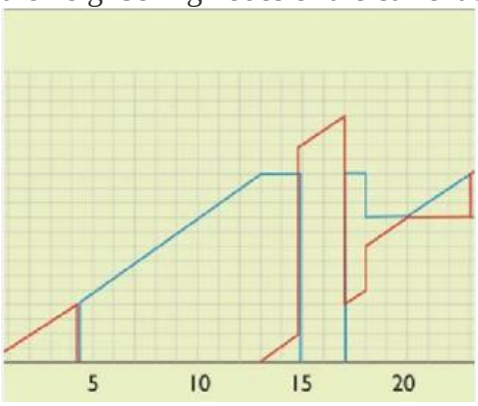

Fig: Steepest -Ascent Hill Climbing

This algorithm consumes more time as it searches for multiple neighbors.

Algorithm for Steepest -Ascent Hill Climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
- **Step 2:** Loop until a solution is found or the current state does not change.
  - Let SUCC be a state such that any successor of the current state will be better than it.
  - For each operator that applies to the current state:
    - Apply the new operator and generate a new state.
    - Evaluate the new state.
    - If it is goal state, then return it and quit, else compare it to the SUCC.
    - If it is better than SUCC, then set new state as SUCC.
    - If the SUCC is better than the current state, then set current state to SUCC.
- **Step 5:** Exit.

Stochastic Hill climbing:

Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

Hill climbing is a simple optimization algorithm used in Artificial Intelligence (AI) to find the best possible solution for a given problem. It belongs to the family of local search algorithms and is often used in optimization problems where the goal is to find the best solution from a set of possible solutions.

- In Hill Climbing, the algorithm starts with an initial solution and then iteratively makes small changes to it in order to improve the solution. These changes are based on a heuristic function that evaluates the quality of the solution. The algorithm continues to make these small changes until it reaches a local maximum, meaning that no further improvement can be made with the current set of moves.

- There are several variations of Hill Climbing, including steepest ascent Hill Climbing, first-choice Hill Climbing, and simulated annealing. In steepest ascent Hill Climbing, the algorithm evaluates all the possible moves from the current solution and selects the one that leads to the best improvement. In first-choice Hill Climbing, the algorithm randomly selects a move and accepts it if it leads to an improvement, regardless of whether it is the best move. Simulated annealing is a probabilistic variation of Hill Climbing that allows the algorithm to occasionally accept worse moves in order to avoid getting stuck in local maxima.

Hill Climbing can be useful in a variety of optimization problems, such as scheduling, route planning, and resource allocation. However, it has some limitations, such as the tendency to get stuck in local maxima and the lack of diversity in the search space. Therefore, it is often combined with other optimization techniques, such as genetic algorithms or simulated annealing, to overcome these limitations and improve the search results.

Advantages of Hill Climbing algorithm:

1. Hill Climbing is a simple and intuitive algorithm that is easy to understand and implement.
2. It can be used in a wide variety of optimization problems, including those with a large search space and complex constraints.
3. Hill Climbing is often very efficient in finding local optima, making it a good choice for problems where a good solution is needed quickly.
4. The algorithm can be easily modified and extended to include additional heuristics or constraints.

Disadvantages of Hill Climbing algorithm:

1. Hill Climbing can get stuck in local optima, meaning that it may not find the global optimum of the problem.
2. The algorithm is sensitive to the choice of initial solution, and a poor initial solution may result in a poor final solution.
3. Hill Climbing does not explore the search space very thoroughly, which can limit its ability to find better solutions.
4. It may be less effective than other optimization algorithms, such as genetic algorithms or simulated annealing, for certain types of problems.

Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence.

Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem. This solution may not be the global optimal maximum.

- In the above definition, **mathematical optimization problems** imply that hill-climbing solves the problems where we need to maximize or minimize a given real function by choosing values from the given inputs. Example-Travelling salesman problem where we need to minimize the distance traveled by the salesman.
- 'Heuristic search' means that this search algorithm may not find the optimal solution to the problem. However, it will give a good solution in a **reasonable time.**
- A **heuristic function** is a function that will rank all the possible alternatives at any branching step in the search algorithm based on the available information. It helps the algorithm to select the best route out of possible routes.

References

1. Abdullah, S., & Alzaqebah, M. (2014). An adaptive artificial bee colony and late acceptance hill-climbing algorithm for examination timetabling. Journal of Scheduling, 17, 249–262.
2. Abuhamdah, A. (2010). Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. International Journal of Computer Science and Network Security, 10, 192–200.
3. Boese, K. D., Kahng, A. B., & Muddu, S. (1994). A new adaptive multistart technique for combinatorial global optimizations. Operations Research Letters, 16, 101–113.
4. Burke E. K., & Bykov, Y. (2008). A late acceptance strategy in hillclimbing for exam timetabling problems: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT2008), Montreal, Canada.
5. Burke E. K., & Bykov, Y. (2012). The Late Acceptance Hill Climbing heuristic. Technical report CSM-192, Computing Science and Mathematics, University of Stirling, UK.
6. Burke, E. K., Bykov, Y., Newall, J., & Petrovic, S. (2004). A timepredefined local search approach to exam timetabling problems. IIE Transactions, 36(6), 509–528.
7. Burke, E. K., McColumn, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research, 176, 177–192.
8. Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140, 266–280.
9. Bykov Y., & Petrovic, S. (2013). An initial study of a novel Step Counting Hill Climbing heuristic applied to timetabling problems: Proceedings of 6th Multidisciplinary International Scheduling Conference (MISTA 2013), Gent, Belgium.
10. Carter, M. W., Laporte, G., & Lee, S. (1996). Examination timetabling: algorithmic strategies and applications. Journal of the Operational Research Society, 47, 373–383.
11. Di Gaspero, L., & Schaerf, A. (2001). Tabu search techniques for examination timetabling. Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science. Berlin: Springer.
12. Erben, W. (2001). A grouping genetic algorithm for graph colouring and exam timetabling. Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science. Berlin: Springer.
13. Goerler, A., Schulte, F., & Voss, S. (2013). An application of late acceptance hill climbing to the traveling purchaser problem. Computational logistics, Lecture Notes in Computer Science. Berlin: Springer.
14. Gogos, C., Goulas, G., Alefragis, P., Kolonias, V., & Housos, E. (2010). Distributed Scatter Search for the examination timetabling problem. PATAT 2010 Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, Belfast, UK.
15. Jackson W., Özcan, E., & Drake, J. H. (2013). Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. The 13th UK Workshop on Computational Intelligence, Guildford, UK.
16. Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation. Part II, graph partitioning. Operations Research, 37(3), 865–892. Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1991). Optimization by simulated annealing: an experimental evaluation; Part II, graph coloring and number partitioning. Operations Research, 39(3), 378–406;
17. McCollum, B., McMullan, P. J., Parkes, A. J., Burke, E. K., & Abdullah, S. (2009). An extended great deluge approach to the examina tion timetabling problem: Proceedings of the 4th Multidisciplinary

International Conference on Scheduling: Theory and Applications (MISTA09), Dublin, Ireland, 424–434.

18. McCollum, B., Schaerf, A., Paechter, B., McMullan, P. J., Lewis, R., Parkes, A. J., et al. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing, 22, 120–130
19. Özcan, E., Bykov, Y., Birben, M., & Burke, E. K. (2009). Examination timetabling using late acceptance hyper-heuristics: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC'09), Trondheim, Norway Petrovic, S., Patel, V., & Young, Y. (2005). University timetabling with fuzzy constraints. Practice and Theory of Automated Timetabling V, Lecture Notes in Computer Science. Berlin: Springer
20. Thompson, J. M., & Dowsland, K. A. (1996). Variants of simulated annealing for the examination timetabling problem. Annals of Operations Research, 63, 105–128.
21. Tierney, K. (2013). Late acceptance hill climbing for the liner shipping fleet repositioning. Proceedings of the 14th EU/ME Workshop, 21– 27.
22. Verstichel, J., & Vanden Berghe, G. (2009). A late acceptance algorithm for the lock scheduling problem. Logistic Management, 2009(5), 457–478.
23. Vancroonenburg, W., & Wauters, T. (2013) Extending the late acceptance metaheuristic for multi-objective optimization. Proceedings of the 6th Multidisciplinary International Scheduling conference: Theory & Applications (MISTA2013). Ghent, Belgium.
24. Yuan, B., Zhang, C., & Shao, X. (2015). A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. Journal of Intelligent Manufacturing, 26, 159–168.