The hill climbing algorithm with random restarts typically works as follows:

1. The algorithm starts with an initial solution.

2. It evaluates the fitness of the initial solution.

3. The algorithm generates a set of neighboring solutions (typically by slightly modifying the current solution).

4. It evaluates the fitness of each neighboring solution.

5. The algorithm selects the best neighboring solution (i.e., the one with the highest fitness) and sets it as the current solution.

6. If the current solution is the global maximum, the algorithm terminates. Otherwise, it repeats steps 3-5 until either the global maximum is found **or a termination condition is met** (e.g., a maximum number of iterations is reached).

When the algorithm reaches a local maximum, it performs a random restart by generating a completely new random solution and starting again from step 2. This increases the likelihood of finding the global maximum by exploring a different area of the solution space.

During each iteration of the algorithm, the hill climbing with random restarts approach checks all the neighboring solutions and selects the best one as the current solution. However, if the algorithm gets stuck at a local maximum, it generates a completely new random solution to explore a different part of the solution space and improve the chances of finding the global maximum.

Variable Neighborhood Search (VNS) is an extension of hill climbing that incorporates multiple neighborhood structures in the search process. While both algorithms share some similarities, there are some key differences in their implementation.

Here is a general outline of the differences in the code between Hill Climbing with Random Restarts and Variable Neighborhood Search:

1. Hill Climbing with Random Restarts:

- Initialize a starting solution randomly

- Set a maximum number of iterations or restarts

- Until termination criterion is met, do the following:

- Generate a neighbor solution using a neighborhood structure

- If the neighbor is better than the current solution, update the current solution

- If the neighbor is worse or equal, continue searching using the same neighborhood

- If the algorithm reaches a local minimum, perform a random restart by generating a new random starting solution and continue the search

- If the maximum number of iterations or restarts is reached, terminate and return the current solution