Tabu Search

- Tabu search is a meta-heuristic search method use to guide and control actual heuristic function of the search algorithm.

- The basic idea of Tabu search is to restrict the previously visited moves/states in search spaces which do not give a better solution.

- Tabu search deterministically allow the Tabu moves/states in order to prevent the search algorithm to get stuck in local optima (Maxima or Minima).

- In Tabu search, nodes in CLOSED are allowed to be used again so that the moves/states which are restricted can be considered if any earlier move saved in CLOSED is better than the current best moves.

- The CLOSED list can be implemented as circular list with k-nodes. The CLOSED list acts here as TABU list.

Terminologies:

- Tabu list
  - The Tabu list is the list that stores a fixed amount of recently made moves.
  - These moves are stored in short-term memory.
  - Short-term memory stores the moves based on recency of occurrence, i.e, most recently mode moves are stored in form of Tabu list (CLOSED list)

Example:

a) Swapping nodes in a graph.
b) Changing a bit from 0 and 1 or vice-versa.
c) Inserting or Deleting edges in a graph.

Tabu Tenure (TT)

- Tabu tenure is the number that tells how long a move stays in the Tabu List so that move cannot be made again as it was already recently visited.

- Tabu tenure (TT) can be implemented as :

  1) **Static**: where TT is to be a constant

  2) **Dynamic:** where TT is randomly chosen between some value TT_min and TT_max

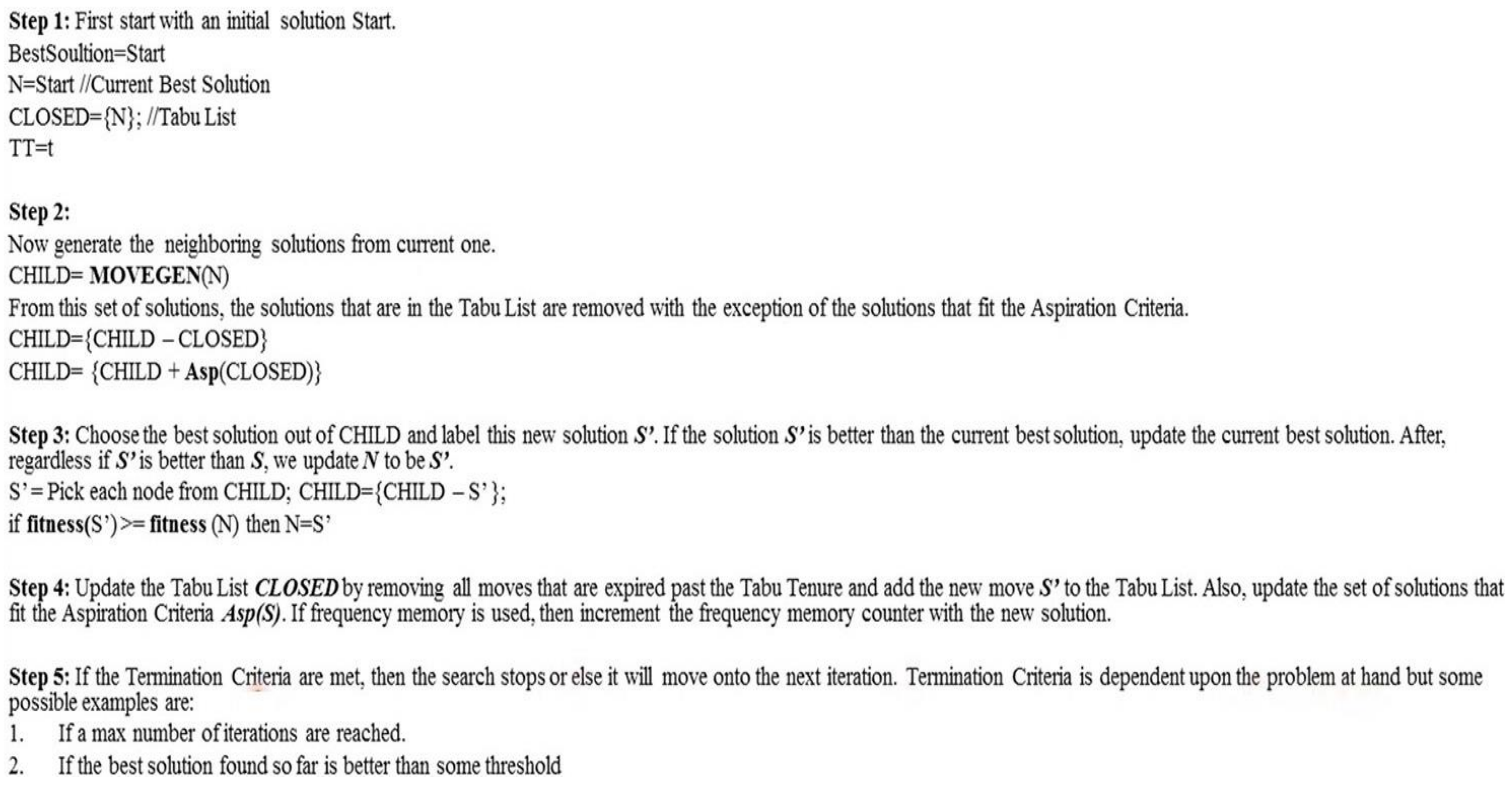
Aspiration Criteria

- Allows the move to be made even if its in the Tabu list
- Some Example are :

    1) If the solution of move in Tabu list is better than the current best solution.

    2) If the Tabu tenure value of a particular move is reached to zero.

Algorithm



Step 1: First start with an initial solution Start.

BestSoultion=Start

N=Start //Current Best Solution

CLOSED={N}; //Tabu List

TT=t

Step 2:

Now generate the neighboring solutions from current one.

CHILD= **MOVEGEN**(N)

From this set of solutions, the solutions that are in the Tabu List are removed with the exception of the solutions that fit the Aspiration Criteria.

CHILD={CHILD – CLOSED}

CHILD= {CHILD + **Asp**(CLOSED)}

Step 3: Choose the best solution out of CHILD and label this new solution ***S'***. If the solution ***S'*** is better than the current best solution, update the current best solution. After, regardless if ***S'*** is better than ***S***, we update ***N*** to be ***S'***.

S' = Pick each node from CHILD; CHILD={CHILD – S'};

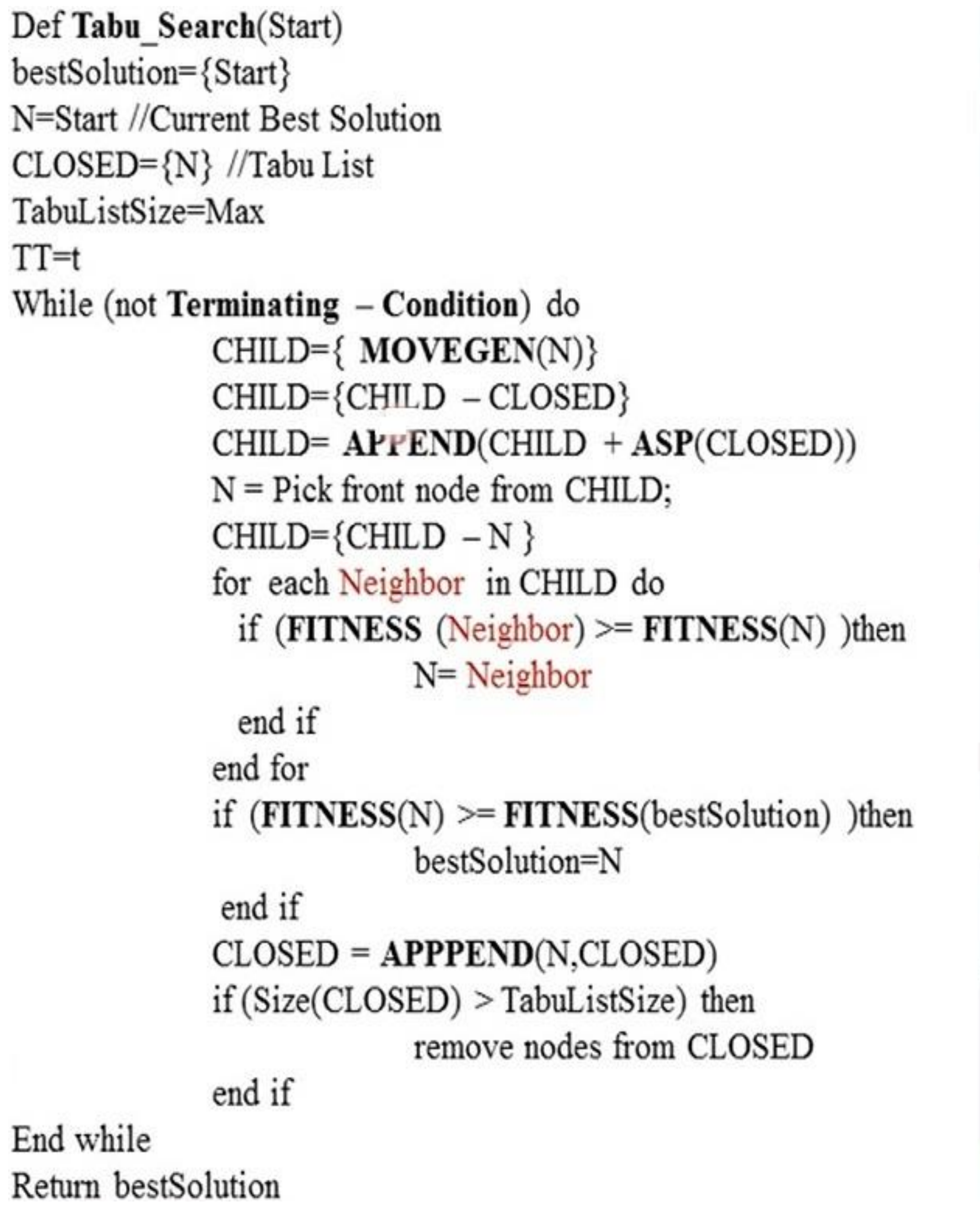
if **fitness**(S') >= **fitness** (N) then N=S'

Step 4: Update the Tabu List ***CLOSED*** by removing all moves that are expired past the Tabu Tenure and add the new move ***S'*** to the Tabu List. Also, update the set of solutions that fit the Aspiration Criteria ***Asp(S)***. If frequency memory is used, then increment the frequency memory counter with the new solution.

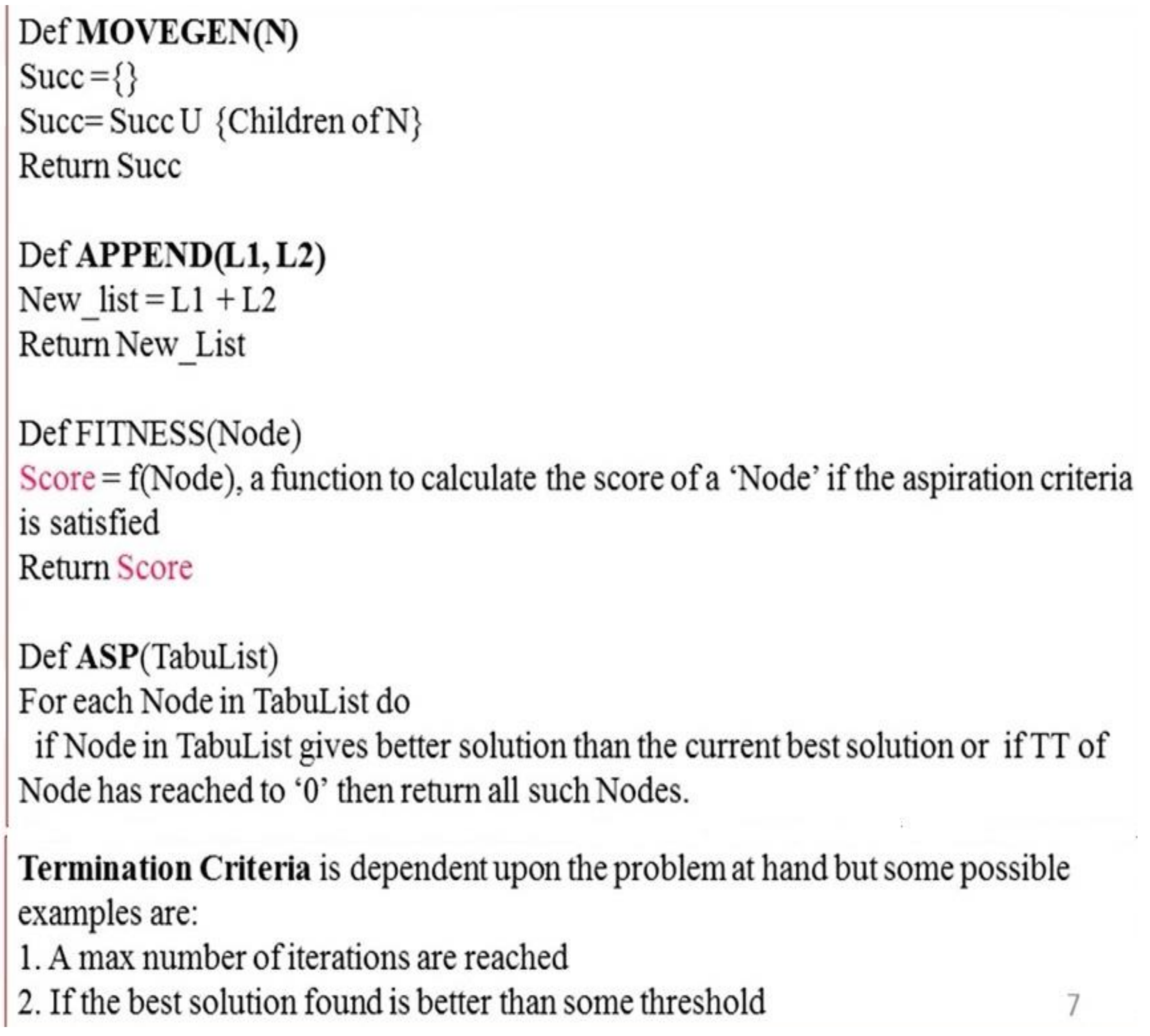
Step 5: If the Termination Criteria are met, then the search stops or else it will move onto the next iteration. Termination Criteria is dependent upon the problem at hand but some possible examples are:

1. If a max number of iterations are reached.
2. If the best solution found so far is better than some threshold

Algorithm



```
Def Tabu_Search(Start)
bestSolution={Start}
N=Start //Current Best Solution
CLOSED={N} //Tabu List
TabuListSize=Max
TT=t
While (not Terminating – Condition) do
            CHILD={ MOVEGEN(N)}
            CHILD={CHILD – CLOSED}
            CHILD= APPEND(CHILD + ASP(CLOSED))
            N = Pick front node from CHILD;
            CHILD={CHILD – N }
            for each Neighbor in CHILD do
              if (FITNESS (Neighbor) >= FITNESS(N) )then
                        N= Neighbor
              end if
            end for
            if (FITNESS(N) >= FITNESS(bestSolution) )then
                        bestSolution=N
             end if
            CLOSED = APPPEND(N,CLOSED)
            if (Size(CLOSED) > TabuListSize) then
                        remove nodes from CLOSED
            end if
End while
Return bestSolution
```



```
Def MOVEGEN(N)
Succ={}
Succ= Succ U {Children of N}
Return Succ

Def APPEND(L1, L2)
New_list = L1 + L2
Return New_List

Def FITNESS(Node)
Score = f(Node), a function to calculate the score of a 'Node' if the aspiration criteria is satisfied
Return Score

Def ASP(TabuList)
For each Node in TabuList do
 if Node in TabuList gives better solution than the current best solution or if TT of Node has reached to '0' then return all such Nodes.
```

Termination Criteria is dependent upon the problem at hand but some possible examples are:

1. A max number of iterations are reached
2. If the best solution found is better than some threshold

Frequency Memory :

- This is a long term memory use to hold frequency of occurrence of each solution which was picked since the beginning of the search.

- The move that was made more frequently without solution is less likely to be picked again.
  This is known as diversification and would promote more diverse solutions.

- Diversification can be:

  a) **Restart diversification:** allows to use the moves that rarely appeared in the current solution. It restarts the search from this move.

  b) **Continuous diversification:** allows to use the moves that rarely appeared in the current solution as they have low value of frequency of occurrence and these moves may have a higher probability of giving solution.

Advantages

1. The Tabu List helps algorithm to revert back to old solution if it is better than current solution.
2. Hill climbing *exploits* the gradient information but Tabu search uses explorative approach to pull out the search from trap of local optima by picking Tabu-moves.
3. It is used to solve both discrete and continuous solutions.

Disadvantages

1. Time complexity is high as already restricted moves can be used again.
2. Many parameters used in algorithm are to be tuned with each other which is a difficult task.