COL333 Assignment 1

Raja Kumar 2021CS10915

November 29, 2023

Optimization Problem using Greedy Hill Climbing with Random Restart

Initialization:

- Initiate the optimization process using Greedy Hill Climbing with Random Restart.
- Randomly select a permutation of locations, denoted as "init_state."
- The first z locations in "init_state" correspond to the zones and are allocated sequentially.

Cost Calculation:

- Compute the cost of the initial state using a complexity function with $O(z^2)$ complexity.

Pair Generation:

- Define a vector named "all_pairs" comprising pairs representing all possible mappings of zones and locations.
- The size of "all_pairs" is $(z \times l)$, where z is the number of zones, and l is the number of locations.

Hill Climbing Step:

- In each iteration of the hill climbing step:
  1. Randomly generate a number, r, between 1 and $l \times z$.
  2. Use r to select a pair (i, j) from the "all_pairs" vector.
  3. Swap the locations at indices i and j in the "init_state."

Cost Evaluation:

- Calculate the partial contribution to the cost before and after the location swap in O(z) complexity

Improvement Check:

- Check for an improvement in the total cost after the swap.
- If an improvement is observed, restart the iterative step.
- If no improvement occurs after $l \times z$ swaps, proceed to the next step.

Random Restart:

- In case of no improvement after $l \times z$ swaps, initiate a random restart.
- Generate a new permutation of locations in "init_state."
- Recalculate the cost for the newly generated state.

Iteration Control:

- Continue the iterative process, incorporating hill climbing steps and random restarts, until the allocated time is exhausted.

Another Approach:

In this variant, two random indices i and j (bounded by $1 \leq i \leq z$ and $1 \leq j \leq l$) are generated. The locations at these indices in "init_state" are swapped, and the partial cost impact is assessed. If no cost improvement is observed after a specified number of swaps, the entire process is restarted. This method introduces a dynamic element through random swapping and employs a restart mechanism for optimization.

Probabilistic Swapping:

- Incorporate a probabilistic approach in the swapping process:
- With probability $\frac{l-z}{l}$, swap unallocated and allocated locations.
- With probability $\frac{z}{l}$, swap allocated locations.

Conclusion:

- In our exploration, it has been determined that the performance of the algorithm improves as the number of climbing steps increases.
- This approach introduces randomness through random swapping, providing a dynamic element to the optimization process.
- The probabilistic approach further diversifies the search space, contributing to the overall efficacy of the algorithm.