Name: User-name:

Algorithm A: hill climbing search

Algorithm B: simulated annealing

Description of enhancement of Algorithm A:

The first enhancement, the introduction of a distance cache, targets the redundancy of distance computations in the TSP. Imagine a scenario with a distance matrix that holds the distances between 50 different cities. In such a scenario, the solver might need to compute the distance between city A and city B multiple times throughout the execution. Normally, each time this distance is needed, the solver would compute it again, wasting computational resources. With the introduction of a distance cache, after the solver computes the distance between city A and city B for the first time, it stores the computed value in the cache. Subsequently, when the distance between city A and city B is needed, the solver retrieves it from the cache, saving computational time and resources. This enhancement is particularly useful for large-scale problems where computational efficiency matters.

The second enhancement, the early stopping strategy in the Hill Climbing algorithm, adds an element of efficiency in finding optimal solutions. Imagine a scenario where our TSP solver is exploring a neighborhood of 100 possible tours. Traditionally, the Hill Climbing algorithm would examine all 100 before deciding on the next step. If the third tour in this neighborhood is better than the current solution, the traditional algorithm would still go through the remaining 97 tours before acknowledging it. However, with the early stopping strategy, the solver stops the exploration right when it finds the third tour is better. It immediately moves to this better solution and starts exploring its neighborhood, eliminating the need to evaluate the remaining 97 tours, thus saving computational time and resources.

The proposed enhancements have been successful in reducing the execution time of the TSP solver. The use of a distance cache significantly reduced redundant computations, which in turn led to a noticeable decrease in the total execution time. Similarly, the introduction of the early stopping strategy improved the performance of the Hill Climbing algorithm, allowing us to find better solutions in less time.

Description of enhancement of Algorithm B:

The first significant enhancement is the modification of the heuristic calculation. The heuristic calculation serves as an estimation of the cost of an optimal tour for a given state, and it is essential for the efficient execution of the algorithm. In the original code, the heuristic calculation involves computation of the cost of remaining edges in the tour and the cost of the minimum spanning tree (MST) which are both computationally intensive tasks. These tasks are performed every time a heuristic calculation is needed, leading to repeated computations. The enhancement modifies the heuristic calculation by precomputing the MST and storing the edges, which effectively reduces the redundant computation of MST during the heuristic calculation. In a TSP scenario involving a large number of cities, this precomputation can lead to a significant reduction in the total execution time.

The second major enhancement is the improvement in the generation of the initial tour. In the original code, the initial tour is generated randomly. This could potentially result in the exclusion of some cities from the initial tour. However, in the enhanced code, the initial tour is a shuffled version of the list of all cities. This ensures that all cities are included in the initial tour, improving the completeness of the tour and enhancing the chances of finding a better solution earlier in the algorithm execution.

These enhancements, particularly the precomputation of the MST edges, can reduce the computational overhead of the heuristic calculation, thus improving the overall efficiency and speed of the TSP solver. The optimized initial tour generation ensures a better starting point for the algorithm, leading to improved algorithm performance.