Algorithm 1: Local Search Algorithm
/* Generate Initial Solution */
1 currentPath = ConstructionHeuristic()
/* Make Local Improvements */
2 bestPath = currentPath
3 numIterNoImprove = 0
4 currentMetric = random metric
5 T = starting temperature
6 for iterations = 1 to maxIterations do
7 | T = T · ΔT
8 | Z = numIterNoImprove \(2^{\text{numIterNoImprove}}\)
9 | /* Perform 2-Opt Operation on currentPath */
10 | /* Remove Vertices from currentPath */
11 | while currentPath is infeasible OR rand() \(\leq Z\) do
12 | /* Insert Vertices to currentPath */
13 | neighborPath = Insert(currentPath, currentMetric)
14 | /* Update currentPath and bestPath */
15 | if ΔR = neighborPath.reward – currentPath.reward then
16 | currentPath = neighborPath
17 | if currentPath.reward > bestPath.reward then
18 | bestPath = currentPath
19 | numIterNoImprove = 0
20 | else
21 | numIterNoImprove = numIterNoImprove + 1
22 | if numIterNoImprove > maxIterNoImprove then
23 | currentMetric = new random metric
24 | numIterNoImprove = 0
25 | end
26 | end
27 | end
28 return bestPath

6 DSOP Algorithms

In this section, we describe a branch-and-bound algorithm and a local search algorithm that solves DSOPs.

6.1 Branch-and-Bound Algorithm

We provide a depth-first branch-and-bound algorithm, where the root of the search tree is the start vertex and the children of a vertex are all the unvisited vertices minus the exit vertex. The branch of an arbitrary vertex thus represents the path from the start vertex to that vertex. The value of a vertex is the sum of rewards of all vertices along its branch. The algorithm prunes the subtree of a vertex if it fails to satisfy our risk-sensitive criterion. For example, assume that a vertex \(v_k\) is on the branch \(\pi = (v_1, v_2, \ldots, v_k)\), where vertex \(v_i\) is on the \(i\)-th position on the branch. The algorithm prunes the subtree rooted at vertex \(v_k\) if the condition in Equation 1 is not satisfied if one appends the exit vertex to the end of the path. The algorithm returns the vertex with the largest value and the branch of that vertex with the exit vertex appended at the end of the path as the best solution that satisfies the risk-sensitive criterion.

6.2 Local Search Algorithm

Unfortunately, the branch-and-bound algorithm suffers from scalability issues as the size of the search tree is exponential in the number of vertices in the graph. We thus introduce a local search algorithm that is based on the standard two-phase approach—a construction heuristic to generate an initial solution followed by local improvements on that solution.

6.2.1 Construction Heuristic

The construction heuristic is a greedy insertion algorithm that greedily inserts the best unvisited vertex at the best position in the current path according to a given metric. The algorithm begins with the path that starts at the start vertex and immediately exits at exit vertex, and it terminates when it can no longer insert any attraction at any position without violating the condition in Equation 1.

In this paper, we use the following metric to evaluate the value of inserting vertex \(v_i\) at position \(p\): \(\frac{\Delta R}{1 + \Delta P}\), where \(\Delta R\) and \(\Delta P\) is the gain in reward and probability, respectively, for inserting vertex \(v_i\) at position \(p\). Thus, \(\Delta R = R(v_i)\), which is the reward of vertex \(v_i\), and \(\Delta P = \hat{P}(a_n \leq H) - \hat{P}(a_n \leq H)\), where \(\hat{P}(a_n \leq H)\) and \(\hat{P}(a_n \leq H)\) is the probability of arriving at the exit vertex before and after insertion, respectively. We use the same approach of multiplying transition matrices described in Section 6.1 to speed up the computation of the probabilities. Finally, we add 1 to the gain in probabilities such that the denominator is greater than 0.

This metric is motivated by similar metrics in knapsack problems, namely the utility of an item is the ratio between the reward and size of that item [23]. We have also tried 4 other variants of the above metric, namely (1) \(\frac{1}{1 + \Delta P}\), (2) \(\frac{\Delta R}{1 + \Delta P}\), (3) \(\frac{(\Delta R)^2}{1 + \Delta P}\), (4) \(\frac{\Delta R}{\sqrt{1 + \Delta P}}\), where we ignored the effects of rewards in (1) and probabilities in (2), and we amplified the effects of rewards in (3) and probabilities in (4). However, our chosen metric was shown to outperform these 4 variants empirically.

6.2.2 Local Improvements

We use a hybrid approach that consists of a variable neighborhood search combined with simulated annealing to locally improve our initial solution found by the construction heuristic. Algorithm 1 shows the pseudocode of this algorithm. After constructing the ini-