ICRA2026 Late Breaking Results Poster presented at
2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Task Correlation and Edge-Cost Estimation for Robotic Task Sequencing Problem in Torus C-Space

Phayuth Yonrith, *Student Member, IEEE* and Ayoung Hong, *Member, IEEE*

***Abstract*— Robotic manipulators are increasingly used in diverse applications, ranging from industrial automation to human-centered tasks such as grocery picking and packaging, where they are often required to perform sequences of tasks while maintaining motion optimality and collision-free operation over long horizons. This type of problem is known as the Robotic Task Sequencing Problem. Most existing works address this problem by reducing it to the Traveling Salesman Problem (TSP) and within a purely Euclidean framework, neglecting the robot's inherently non-Euclidean toroidal $\mathcal{C}$-space topology. This simplification limits the selection of feasible configurations and may lead to failure, suboptimal, or detoured motions. In this paper, we propose a robotic task sequencing problem solver that incorporates the robot's natural $\mathcal{T}^n$ topology and joint limits.**

I. Introduction

The Robotic Task Sequencing Problem (RTSP) requires a robot to determine an optimal sequence of motion that visits multiple task spaces while avoiding obstacles. In general, this problem is formulated as a Generalized Traveling Salesman Problem (GTSP). However, solving the RTSP as a whole GTSP is intractable as the number of tasks increases. Previous approaches [1], [2], often reduce the problem into a TSP. Although this simplification makes the problem easier to solve, it does not accurately estimate the edge costs, neglects the reachability between tasks, and fails to consider the torus $\mathcal{C}$-space. As a result, potentially better candidate configurations that could reduce the overall cost may be discarded.

In this paper, our main contributions are as follows:

- Task correlations estimation using the nearest neighbor method to reduce the number of GTSP edges.
- Edge costs estimation using an implicit sparse Random Geometric Graph (RGG).

II. Robotics Task Sequencing Problem

A. Problem Definition

We define the RTSP problem as a GTSP. In this problem, a single pose in the task-space contained a cluster of unique configurations in the $\mathcal{C}$-space. Furthermore, each unique configuration has multiple copies of self-redundancy induced by the torus representation in Euclidean space. The problem is modeled as a symmetric undirected weighted graph for simplicity, $G \leftarrow \{V, E\}$. The vertices set V contains a set of unique IK configurations and their torus copies $q_i \in Q$. Edges set E contain a collision-free path connecting each $e_{ij} \in E$. The problem graph visualization is shown in Fig.1.

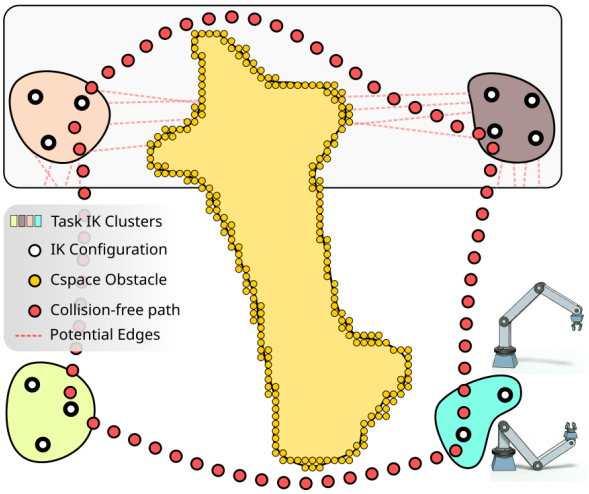

Fig. 1. RTSP Problem Definition Model Visualization. Each task contains a unique IK configuration inside its cluster shape. An edge connection between vertices exists in the collision-free space shown in a red dotted path. The tour must be collision-free and minimize the total length along the tour.

B. Proposed Algorithm

Since the problem arises from the robotics manipulator properties, we can use this fact to simplify the problem to a tractable one.

We seek to minimize the distance of a collision-free tour between task poses. Given a set n tasks as task-space poses $h \in H$ defined by position and quaternion.

$$h_i \leftarrow \{x, y, z, q_x, q_y, q_z, q_w\}$$

Each task h_i has a set of valid inverse kinematics solutions $q \in Q$. This is typically solved using a geometry-based analytical inverse kinematics solver. Each q_i is a unique configuration solution that solves $h_i \leftarrow \text{forwardkinematic}(q_i)$. For a general 2-DOF robot, there are up to 2 unique solutions (elbow up/down); a 6-DOF robot can have up to 8 unique solutions. For a redundant robot (DOF$>$ 6), we discretize the free DOF and solve for a unique $q_i \leftarrow \{\phi_1, \phi_2, ..., \phi_j\}$.

Since the robot arm joint is revolute, naturally its $\mathcal{C}$-space is a torus. Due to the physical joint limit, the space representation is now Euclidean. Therefore, a configuration has multiple copies of each solution in each space. For each q_i, there are ϕ_i that have the same exact robot configuration

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00340040).

Phayuth Yonrith and Ayoung Hong are with the Department of Mechanical Engineering, Chonnam National University, Gwangju 61186, Republic of Korea ahong@jnu.ac.kr

but different joint values θ. These configurations can be computed from the Algorithm 1.

Algorithm 1 Find Alternative Configuration

```
function AltConfig(q)
    v ← {−2kπ..., −2π, 0, 2π, ..., 2kπ}
    Q_feasible ← {∅}
    q_wrap ← WrapToPi(q)
    Q_shifted ← {v × v × ...(n)}
    for q_shifted ∈ Q_shifted do
        q_alt ← q_wrap + q_shifted
        if IsInJointLimit(q_alt) then
            Q_feasible ← Q_feasible ∪ {q_alt}
    return Q_feasible
```

Algorithm 2 RTSP Solver

```
function RTSP Solver(H, q_home)
    Step 1: Task Correlation Estimation (H, q_home)
    Step 2: Edge-Cost Estimation (Q)
    Step 3: GTSP Solve (q)
    Step 4: Tour Refinement (q)
    return σ Collision-free Path Tour
```

The overview of our proposed approach is shown in Algorithm 2. The computationally intensive operation that consumes most of the computation time is Edge-Cost Estimation, since each edge requires a cost value for the GTSP solver to provide a solution. To get the collision-free cost for each edge, the path planner must be called. Depending on the number of tasks and time limit, it is unreasonable to called path planner for all edges. Therefore, they must be reduced and estimated.

The first component that determines the number of edges is the number of tasks. If the task can be eliminated, it will, in turn, drastically reduce the number of edges. The second is the number of valid IK configurations. If the configuration is eliminated, it reduces the size of the problem, as well as the edges to all other task configurations and their torus copies, which are no longer valid.

Step 1: Task Correlation Estimation is performed to reduce the edges number. Tasks that are close to each other can be correlated and therefore most likely to exist in the optimal solution and vice versa. The nearest neighborhood technique is used on the task-space pose H to find correlation using the distance metric below. The neighborhood of the task will have its edges cost estimated the further away have ∞ cost assigned to their edges.

$$
\begin{aligned}
d(h_1, h_2) &= \sqrt{e_t^2 + \lambda e_r^2} \\
e_t &= ||t_2 - t_1||_2 \\
e_r &= \cos^{-1}\left(\frac{trace(R_1^T R_2) - 1}{2}\right)
\end{aligned}
\tag{1}
$$

The neighborhood is considered a union of K_{nn} and R_{nn} sets $H_{nn} \leftarrow H_r \cup H_k$ The key intuition is that R_{nn} is robust when the h are packed near each other making the task aware of its surroundings. K_{nn} is robust when h is further away making outlier find its neighbor. When the tasks are tightly packed, R_{nn} will dominate K_{nn} and disregard the further connection.

Step 2: Edges cost estimation is performed to determine an approximate cost of edge. It is costly to compute and thus requires an algorithm to be designed in a way that produces a quick result near the optimal in the given time. The implicit RGG data structure and lazy collision evaluation is used. First, a set of collision-free nodes is sampled from the $\mathcal{C}$-space with Poisson disk sampling. The node and edges are sparsified and searched with Dijkstra for approximate cost.

Step 3: After all costs are obtained, we solve the GTSP with the cost using the off-the-shelf solver GLKH.

Step 4: Finally, the result tour from the GTSP is then used to sequence the configuration and compute the collision-free tour. We can perform a fresh plan from the sequence, or use the estimated configuration path from step 2 and refine it using STOMP for smooth motion.

III. Results

We prepare a 2D Planar Robot Experiment to demonstrate and compare the algorithm to RobotTSP [1]. RoboTSP found a solution with a cost of 31.781, while constrained to follow the taskspace TSP within $\pm\pi$ joint limits. Our proposed algorithm found a solution with a cost of 25.309 without a taskspace constraint and using a $\pm 2\pi$ range.

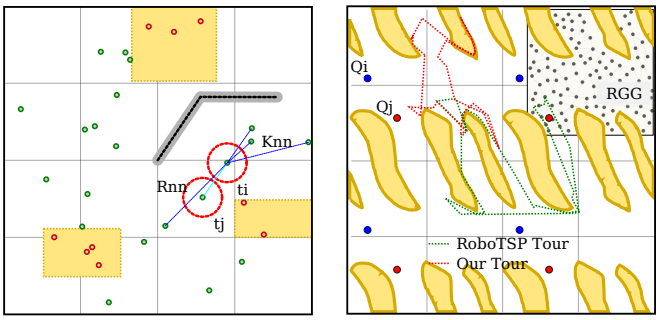

Fig. 2. (Task-space) Valid (green) and invalid (red) task-space poses to be visited. The effect K_{nn} (blue line) and R_{nn} (red circle) in finding the neighborhood of the task. ($\mathcal{C}$-space) The effect of self-redundancy of torus in $\mathcal{C}$-space is shown in four red and blue dots. The final tour comparison between algorithms is shown in green and red dashed lines.

IV. Conclusion

The proposed algorithm allows the RTSP to be solved as a GTSP without the need to be reduced. It allows the robot to fully operate near its physical joint limit. Thus, it is able to find a better tour with a lower cost. Due to the extra estimation step, our algorithm takes a higher time to compute. However, this inspires us to solve the estimation problem in parallel to fasten the computation time.

References

[1] F. Suárez-Ruiz, T. S. Lembono, and Q.-C. Pham, "Robotsp–a fast solution to the robotic task sequencing problem," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1611–1616.

[2] C. Wong, C. Mineo, E. Yang, X.-T. Yan, and D. Gu, "A novel clustering-based algorithm for solving spatially constrained robotic task sequencing problems," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 5, pp. 2294–2305, 2020.