

# A Novel Clustering-Based Algorithm for Solving Spatially-Constrained Robotic Task Sequencing Problems

Cuebong Wong\*, Carmelo Mineo, Erfu Yang, *Member, IEEE*, Xiu-Tian Yan, and Dongbing Gu, *Senior Member, IEEE*

**Abstract**—The robotic task sequencing problem (RTSP) appears in various forms across many industrial applications and consists of developing an optimal sequence of motions to visit a set of target points defined in a task space. Developing solutions to problems involving complex spatial constraints remains challenging due to the existence of multiple inverse kinematic solutions and the requirements for collision avoidance. So far existing studies have been limited to relaxed RTSPs involving a small number of target points and relatively uncluttered environments. When extending existing methods to problems involving greater spatial constraints and large sets of target points, they either require substantially long planning times or are unable to obtain high-quality solutions. To this end, this paper presents a clustering-based algorithm to efficiently address spatially-constrained RTSPs involving several hundred to thousands of points. Through a series of benchmarks, we show that the proposed algorithm outperforms the state-of-the-art in terms of solution quality and planning efficiency for large, complex problems, achieving up to 60% reduction in task execution time and 91% reduction in computation time.

**Index Terms**—Robotic task sequencing, manipulation, optimal planning, autonomous inspection.

## I. INTRODUCTION

**R**OBOTIC task sequencing is an important consideration in modern industrial robotics. In many applications, a manipulator is required to perform a large number of repetitive and onerous tasks as efficiently as possible to maximise throughput. Some examples of these include free-form surface inspection, drilling, spray-painting, screw fastening and spot-welding [1], [2]. Developing optimized task sequences and motion plans for tasks that consist of visiting several hundred

\*Corresponding author.

This research was funded by the Engineering and Physical Sciences Research Council (EPSRC) under its Doctoral Training Partnership Programme (DTP 2016-2017 University of Strathclyde, Glasgow, UK) and partly supported by the Advanced Forming Research Centre (University of Strathclyde, Glasgow) through the Route to Impact funding scheme.

C. Wong is with the National Nuclear Laboratory, Havelock Rd, Workington, CA14 3YQ, United Kingdom (email: cuebong.wong@uknpl.com).

E. Yang and X-T. Yan are with the Department of Design, Manufacturing and Engineering Management, University of Strathclyde, 75 Montrose St, Glasgow, G1 1XJ, United Kingdom (e-mail: erfu.yang@strath.ac.uk; x.yan@strath.ac.uk).

C. Mineo is with the Dipartimento di Ingegneria, Università di Palermo, Viale delle scienze, Edificio 8, 90128 Palermo, Italy (e-mail: carmelo.mineo@unipa.it).

D. Gu is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, C04 3SQ, United Kingdom (e-mail: dgu@essex.ac.uk).

to thousands of points can be both challenging and time-consuming, yet this is still predominantly performed offline by a skilled programmer. This heavily limits the usability of robots for applications involving: (i) rapid deployment in unique, one-of-a-kind scenarios, where substantial offline planning is particularly costly, (ii) unstructured environments beyond the carefully designed industrial shopfloor, where access is often restricted by surrounding obstructions, and (iii) tasks that are subject to high variability, which may render existing offline plans invalid.

These problems have motivated several efforts in the robotics research community to develop autonomous solutions to the Robotic Task Sequencing Problem (RTSP). The RTSP consists of finding a sequenced series of collision-free motions to optimally visit a set of target points and closely resembles the classic algorithmic Travelling Salesman Problem (TSP) [3]. However, the RTSP involves additional complexity introduced by a manipulator's *kinematic redundancy*. That is, there exists more than one robot configuration from which the robot can reach a given target point in *task space*. Early work in RTSP avoided this problem by arbitrarily choosing a single configuration for each target. However, this commonly led to highly sub-optimal solutions as configuration space (C-space) information were not considered. More recent approaches accounted for kinematic redundancy by objectively selecting configurations that led to higher quality task sequences. However, until now these methods have either required substantially long planning times or are unable to find high-quality solutions when applied to spatially-constrained problems.

Building upon these prior works, we present the Cluster-RTSP algorithm<sup>1</sup>, a novel approach to solving RTSPs. Through a series of experimental evaluations, we show that our proposed algorithm is capable of finding higher quality solutions in spatially-constrained RTSPs while requiring less computation time than existing approaches. In addition, we describe a short case study consisting of a mock surface inspection of pipes using a KUKA KR 6 R900 Sixx robot and provide a discussion on considerations for implementation that would be relevant to practitioners.

The contributions of this paper are as follows: (i) we introduce a new C-space heuristic for configuration selection that determines a *best-fit* configuration for each target point

<sup>1</sup>An open-source implementation of our algorithm is available at: <https://github.com/Cuebong/Cluster-RTSP>.

taking into account the requirement to optimise the global task sequence, (ii) we present a new formulation of the RTSP as a Clustered TSP (CTSP) in the C-space, showing that by solving the RTSP in this representation, it is possible to achieve significantly faster planning speeds than what is currently achievable in RTSP literature, and (iii) we propose a new algorithm called Cluster-RTSP and show experimentally that it can achieve up to 60% reduction in task execution time and up to 91% in computation time when compared to a representative state-of-the-art approach. To the best of our knowledge, this work is the first to consider RTSPs that involve a combination of hard spatial constraints and substantially large sets of target points (up to 1500 targets).

The remainder of this paper is organized as follows. Section II provides a review of related literature in RTSP, while in Section III we present the proposed algorithm. Benchmarking results based on simulation are presented and discussed in Section IV. Section V presents a case study on the application of Cluster-RTSP to surface inspection of pipes, Section VI describes the limitations of the current work, and Section VII concludes this paper.

## II. RELATED WORKS

To highlight the current challenges of RTSP, we discuss briefly the progression of research developments in literature to date. The work in [4] was one of the first to consider the use of TSP for robotic applications, where task sequencing for an industrial robot was solved as an asymmetric TSP. However, this early work only considered a single configuration for each target point and neglected the potential motion cost reductions made possible by kinematic redundancy.

In [5], configuration selection from among multiple inverse kinematic (IK) solutions was addressed by formulating the problem as a Generalised TSP (GTSP), such that IK solutions belonging to the same point were grouped and the objective was to find a tour that visited one configuration in each group. This was evaluated on a 3 degrees of freedom (DoF) robot. The authors in [6] addressed the configuration selection problem for an RTSP consisting of a 6-DoF robot by using a constraint optimization model to solve the RTSP formulated as a multi-objective optimisation problem. For problems involving 12 targets 6 IK solutions each, their method required several hours of computation time to obtain a solution. Zacharia et al. [7] took a different approach and proposed an encoding for a Genetic Algorithm (GA) that contained both the task sequence and corresponding configurations in the optimization. In their experiments, the computation time for a 6-DoF robot with a problem size of 50 points was approximately 1,800 seconds.

Robotic laser welding applications was addressed in [8], where the RTSP was formulated as a modified TSP with Neighbourhoods (TSPN) and solved in task space. In the reported experiments, a fixed planning duration of 600 seconds was allocated to solve the TSPN, considering problems involving up to 71 points. The authors assumed that the operational area was free of obstacles due to the nature of a purpose-designed fixture, but noted that in practice collisions do exist when designs prioritize other objectives. To compensate for

this, their approach required a path correction procedure in post-processing to guarantee a collision-free solution.

To account for collision avoidance, the authors in [9] formulated the RTSP as a combined Set Covering Problem (SCP) and TSP (SCTSP). Their algorithm consisted of a travelling cost evaluation procedure that computed a valid motion plan for every possible pose-to-pose movement (only one pose was generated for each target point). For a vision-based inspection experiment consisting of a clutter-free environment and 400 potential target points, the computation time was reported to be approximately 3 hours, with the majority of this time consumed by the computation of pose-to-pose motion plans. The authors later extended this work to redundant robotic systems [10], where multiple kinematic solutions exist for each target point. While computation times for this method were not reported, one could expect a computation time several folds greater than their initial work. Ultimately, their findings highlighted the impracticalities of exhaustively computing the cost of trajectories between all pose-to-pose motions.

In view of this, the authors in [11] handled collision avoidance in RTSPs by creating a probabilistic roadmap that contained the specified target configurations and solving for a tour of the points based on the connectivity of the roadmap. One limitation of their work was the assumption that configurations were known a priori, without addressing how they were chosen. Alternatively, the authors in [12] and [13] extended the work in [7] by applying a *Bump Surface* concept to capture obstacle occupancy information in the 2D and 3D search space, respectively, as a single mathematical entity, which was encoded into a single objective function and solved using a GA. However, in their experiments only small problem sizes of up to 15 target points were considered.

Gueta et al. [14] sought to reduce the computation time required for solving large sequencing problems through clustering. Their proposed method solved RTSPs by first dividing task space points into a fixed number of clusters according to their topological locations. The RTSP then became a problem of finding a tour across clusters and the subsequent visiting sequences for navigating points in each cluster. Likewise, we also apply clustering techniques to reduce computation time when solving RTSPs. However, we differentiate our method from [14] by applying clustering in the C-space.

More recently, the RoboTSP algorithm [15] was shown to solve large sequencing problems by several orders of magnitude less computation time while producing solutions of similar quality when compared to other existing approaches. It achieved this by first solving for a task space tour of the goal points as a TSP and then determining the best robot configuration for each target point by applying Dijkstra's algorithm to a graph composed of configurations connected to those that correspond to the predecessor/successor target points in the resulting tour. A limitation of this method is the inability to account for obstacles when finding the optimal task sequence. Hence the quality of solutions found by RoboTSP may be sub-optimal for spatially-constrained problems.

On the other hand, the work in [16] demonstrated the use of the Lin-Kernighan-Helsgaun (GLKH) solver to address RTSPs formulated as an Equality GTSP. Here the objective was to

trace a number of open and closed contours using a remote laser processing system. The RTSP consisted of optimizing the sequence of entry and exit points to access each contour sequentially. For a 7-DoFs redundant system, 100 configurations were sampled at each entry and exit point. The GLKH solver enabled the optimization of the sequence while accounting for kinematic redundancy. Crucially, the authors noted that the GLKH solver did not guarantee a globally optimal solution, but was able to find better solutions over increasing iterations. In their study consisting of a combined total of 52 entry and exit points and 3000 GLKH iterations, solutions were obtained in approximately 15 hours of computation time.

Evidently, it remains a challenge to find optimal solutions to RTSPs comprising of many points in cluttered environments within practical times. To this end, the Cluster-RTSP algorithm was developed to advance the state-of-the-art towards this goal.

### III. CLUSTER-RTSP ALGORITHM

#### A. Problem Formulation

We formally define the RTSP as follows. Let  $T$  be the task space that contains the set of target points  $P_n = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$  such that  $\mathbf{p}_i \in T$ . Let  $C$  be the C-space of the robot and  $C_{obs} \subset C$  be the set of configurations that are in collision with the set of obstacles  $O$ . The set of collision-free configurations is then given as  $C_{free} \subset C/C_{obs}$ . Let  $\mathbf{q} \in C_{free}$  be a single collision-free configuration in the C-space and  $Q_i = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k\}$  be the set of valid IK solutions that reach target point  $\mathbf{p}_i \in P_n$ . Define  $Q' = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$  as the unordered set of assigned configurations such that  $\mathbf{q}_i \in Q_i$  and  $S = (\mathbf{q}_{\{1\}}, \mathbf{q}_{\{2\}}, \dots, \mathbf{q}_{\{n\}})$  as an ordered sequence of  $Q'$ .

Letting  $\sigma_{s \rightarrow g}$  be a trajectory between a start configuration  $\mathbf{q}_s$  and a goal configuration  $\mathbf{q}_g$  and letting  $\Omega_S$  be the short-hand representation for  $\Omega(S)$ , denote  $\Omega_S = (\sigma_{\{0\} \rightarrow \{1\}}, \sigma_{\{1\} \rightarrow \{2\}}, \dots, \sigma_{\{n-1\} \rightarrow \{n\}})$  as the ordered set of trajectories that provide the motions required to visit each configuration contiguously in  $S$ . Finally, denote  $\Sigma = \{S_1, S_2, \dots, S_m\}$  as the set of all valid configuration sequences and define a cost function  $f: \Omega_S \rightarrow \mathbb{R}_+$  that maps an ordered set of trajectories for  $S$  to a real positive cost value. The RTSP is then formally defined as follows:

**Definition (RTSP).** *Given the set of obstacles  $O$  and the set of target points  $P_n$ , find the optimal configuration sequence  $S^*$  and the optimal set of trajectories  $\Omega_{S^*}$  such that  $f(\Omega_{S^*}) = \min_{S \in \Sigma} f(\Omega_S)$  and  $\sigma_{s \rightarrow g} \subset C_{free}$  for all  $\sigma \in \Omega_{S^*}$ .*

Since the primary objective of most robotic sequencing tasks is to maximize efficiency, we use the task execution time, obtained as the sum of the individual trajectory durations, as the cost function  $f$ .

#### B. The Proposed Algorithm

Our proposed algorithm is shown in Algorithm 1. It solves the RTSP using the following steps:

- 1) The set of all collision-free IK solutions  $Q_i$  is computed  $\forall \mathbf{p}_i \in P_n$ . In our implementation, we use the *IKFast* kinematics solver available on *OpenRave* [17].

#### Algorithm 1 Cluster-RTSP

**Input:** Set of  $n$  target points  $P_n$  and home configuration  $\mathbf{q}_0$   
**Output:** Ordered configuration sequence  $S$  and corresponding set of trajectories  $\Omega$

```

1:  $S \leftarrow \mathbf{q}_0, Q \leftarrow \mathbf{q}_0$ 
2: for all  $\mathbf{p}_i \in P_n$  do
3:    $Q.append(getAllIKSolutions(\mathbf{p}_i))$ 
4: end for
5:  $Q' \leftarrow configurationSelection(Q)$ 
6:  $X_K \leftarrow configurationClustering(Q')$ 
7:  $gtour \leftarrow globalTSP(clusters, X_K, \mathbf{q}_0)$ 
8: for all  $idx \in gtour$  do
9:    $Q'_c \leftarrow X_K[idx]$ 
10:   $entry, exit \leftarrow getEntryExitPoints(Q'_c)$ 
11:   $tour \leftarrow localTSP(Q'_c, entry, exit)$ 
12:   $S.append(Q'_c[tour])$ 
13: end for
14:  $S.append(\mathbf{q}_0)$ 
15: for all  $\mathbf{q}_{\{i\}} \in S$  do
16:   $trajectory \leftarrow planTrajectory(\mathbf{q}_{\{i\}}, \mathbf{q}_{\{i-1\}})$ 
17:   $\Omega.append(trajectory)$ 
18: end for

```

- 2) A configuration selection procedure determines the best-fit configuration  $\mathbf{q}_i$  for all  $\mathbf{p}_i$  according to a similarity heuristic derived from C-space metrics. (See Line 5)
- 3) A data clustering algorithm is applied to  $Q'$  to form *configuration clusters*, where all configurations within a cluster lie in close proximity in  $C$ . (See Line 6)
- 4) The RTSP is solved as a CTSP to obtain  $S$  by finding the optimal tour across clusters and the locally optimal sequence for visiting configurations contiguously within each cluster. (See Lines 7-13)
- 5) Collision-free trajectories are computed for each pose-to-pose motion in  $S$ . In our implementation we use the bi-directional RRT [18] available in *OpenRave* [17] to obtain collision-free trajectories that satisfy joint limits and velocity & acceleration constraints.

1) *Configuration Selection:* The configuration selection procedure obtains best-fit configurations according to a heuristic *similarity measure*  $\phi$  as follows. Given the input set of valid collision-free IK solutions  $Q_i$  for all target points in  $P_n$ , the dissimilarity function  $\delta$  is defined as the weighted squared Euclidean distance in C-space between two configurations:

$$\delta(\mathbf{q}, \mathbf{q}') = \sum_{j=1}^{DoF} w_j (q'_j - q_j)^2 \quad (1)$$

where  $w_j$  is a positive weight for joint  $j$  and is derived from the relative maximum displacement of any point on the robot when actuated at the corresponding joint [15].

Now suppose  $Q$  is the *population* of all valid configurations comprising of the IK solutions  $\forall \mathbf{p}_i \in P_n$ . The similarity measure  $\phi$  for any configuration  $\mathbf{q} \in Q$  is given by (2):

$$\phi(\mathbf{q}_i) = b_\delta \cdot \bar{\delta}(\mathbf{q}_i) + b_0 \cdot \delta(\mathbf{q}_i, \mathbf{q}_0) \quad (2)$$

where  $\mathbf{q}_0$  is the home configuration,  $b_\delta$  and  $b_0$  are biases such that  $b_\delta + b_0 = 1$ , and  $\bar{\delta}(\mathbf{q}_i)$  is the mean dissimilarity of  $\mathbf{q}_i$  from

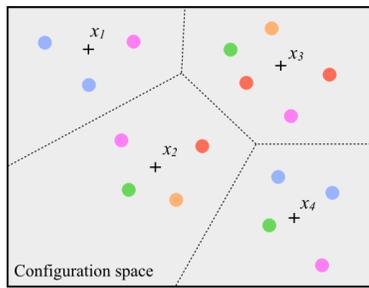


Fig. 1. Illustration of configuration groups in 2D representation. Markers of the same color represent configurations associated to the same target point. The group means are shown as crosses.

the population  $Q$ . Accordingly, a configuration with a high  $\phi$  value lies further away from  $\mathbf{q}_0$  and the population  $Q$  in C-space (the selection biases configurations that lie closer to the home configuration).  $\delta(\mathbf{q}_i)$  could be obtained by computing  $\delta(\mathbf{q}_i, \mathbf{q}_j)$  for all  $\mathbf{q}_j \in Q|_{j \neq i}$ , but this is expensive for large values of  $n$ . Instead, we apply an initial instance of data clustering<sup>2</sup> to divide  $Q$  into  $K$  groups that are described by the mean of configurations,  $x$ , and the number of configurations in the group,  $r$  (see Fig. 1). Letting  $r_n = |Q|$ , the approximated mean dissimilarity  $\bar{\delta}_w(\mathbf{q})$  can then be obtained using (3).

$$\bar{\delta}_w(\mathbf{q}) = \sum_{c=1}^K \frac{\delta(\mathbf{q}, \mathbf{x}_c) \cdot r_c}{r_n} \quad (3)$$

The best-fit configuration for target point  $p_i$  is chosen from among the set of IK solutions  $Q_i$  according to  $\phi$  by applying *iterative configuration reduction* as follows. For each non-singular set of IK solutions  $Q_i|_{i=1}^n$ , compute  $\phi(\mathbf{q})$  for all  $\mathbf{q} \in Q_i$  and remove  $m_i$  configurations with the largest  $\phi$  values from  $Q_i$  and  $Q$  (where  $m_i$  is determined from (4)). This procedure is iteratively applied until every set converges logarithmically to one configuration, which forms the unordered set of configurations  $Q'$ . These iterations remove the effects of poor configurations on the similarity measure of good candidate configurations by ensuring the assignment of best-fit configurations is informed by updated  $\phi$  values.

$$m_i = \max(1, \lfloor \ln |Q_i| \rfloor) \quad (4)$$

The configuration selection procedure returns the unordered set  $Q'$ , containing one allocated configuration for each target point  $\mathbf{p}_i \in P_n$ , which is passed as input to the next step of the algorithm. A reduced example of the output returned by this procedure for a small set of target points is shown in Fig. 2.

2) *Configuration clustering*: Given the input set  $Q'$ , an optimal sequence  $S$  could be obtained by adopting a TSP formulation. However, numerous works have observed that the complexity of a TSP is exponential in practice [15], [19], [20]. The Cluster-RTSP overcomes these challenges by employing clustering techniques to partition  $Q'$  into smaller sequencing sub-problems, which are collectively faster to solve.<sup>3</sup>

<sup>2</sup>The method used for data clustering is described in Section III-B2.

<sup>3</sup>This separate instance of clustering provides a best-fit assignment of clusters for the reduced set of configurations,  $Q'$ , in contrast to the groups originally assigned during configuration selection.

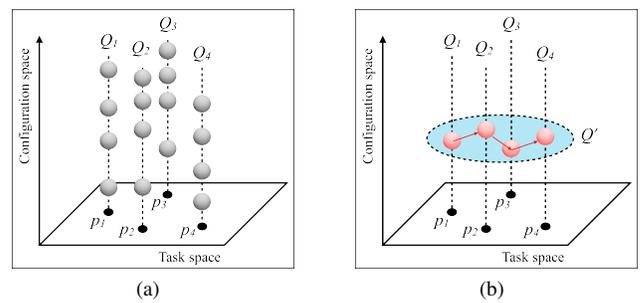


Fig. 2. Illustration of the configuration selection procedure. (a) Four target points in the task space and corresponding configurations. (b) Selected configurations and an example sequence for visiting each configuration (where the latter is obtained from later steps).

We choose to apply the X-means algorithm [21] for clustering, which is an extension of the well-known  $k$ -means clustering algorithm. While both algorithms adopt a spherical Gaussian model assumption,  $k$ -means requires the number of clusters  $K$  to be specified explicitly. X-means removes this limitation by computing the optimal number of clusters from a pre-defined range  $[K_{min}, K_{max}]$  to best fit a set of data.

The X-means algorithm uses a model selection criterion to evaluate the quality of fit between a set of clusters and a set of configurations. Starting with an initial instance of  $k$ -means clustering where  $K = K_{min}$ , the algorithm iterates through each resulting cluster and determines whether better fit can be achieved by bisecting this (*parent*) cluster into two *children* clusters. If the fitness improves,  $K$  is increased by one. This continues up to the upper bound  $K_{max}$ . A second instance of  $k$ -means is applied to obtain the membership of all configurations given the final  $K$  value.

The model selection criterion must adequately determine whether an additional cluster would result in overfitting. We adopt the Bayesian Information Criterion (BIC) formulation presented in [22], which was originally used by the authors who introduced the X-means algorithm [21]. The BIC is made up of a component based on the likelihood function and additionally penalizes the number of parameters in the model (i.e. the number of DoFs,  $d$ , and the number of clusters,  $K$ ) to avoid overfitting. It adopts the spherical Gaussian assumption for  $k$ -means and is obtained by:

$$BIC(M_a) = \hat{l}_a(Q') - \frac{h_a}{2} \cdot \log |Q'| \quad (5)$$

where  $M_a$  is the  $a^{\text{th}}$  model,  $h_a$  is the number of parameters for model  $a$  and  $\hat{l}_a(Q')$  is the log-likelihood of the set of selected configurations. Let  $\hat{\sigma}^2$  denote the maximum likelihood estimate for the variance under identical spherical Gaussian assumption:

$$\hat{\sigma}^2 = \frac{1}{r_n - k} \sum_{i=1}^{r_n} (\mathbf{q}_i - \mu_i) \quad (6)$$

Where  $\mu_i$  is the centroid associated to the  $i^{\text{th}}$  configuration and  $r_n = |Q'|$ . By denoting  $Q'_c$  as the set of points in cluster  $c$  and  $r_c = |Q'_c|$ , the log-likelihood of a cluster is given by:

$$\hat{l}(Q'_c) = -\frac{r_c}{2} \log(2\pi) - \frac{r_c \cdot d}{2} \log(\hat{\sigma}^2) - \frac{r_c - k}{2} + r_c \log r_c - r_c \log r_n \quad (7)$$

Now suppose a model A is the parent cluster and model B represents two children clusters. Model B is considered better than model A when the inequality in Eq. (8) holds.<sup>4</sup>

$$BIC(M_B) > BIC(M_A) \quad (8)$$

After applying configuration clustering, the set of clusters  $X_K = \{Q'_1, Q'_2, \dots, Q'_K\}$ , where  $Q'_c \subset Q'$ , is returned.

3) *Clustered Travelling Salesman Problem*: Given the set of configuration clusters  $X_K$ , we solve for a configuration sequence by treating the problem as a CTSP, which was first introduced by Chisman in 1975 [24]. In a standard CTSP, the set of clusters may be visited in any order, but every point within each cluster must be visited contiguously (otherwise the problem reverts to the classic TSP).

Following the benchmarking results from [15] for TSP solvers, we apply the 2-Opt algorithm [25] separately at the inter-cluster and intra-cluster level using the Euclidean distance metric in C-space as the estimated cost for moving between two configurations. To apply the 2-Opt algorithm at the inter-cluster level, we generate a pairwise distance matrix between all clusters by selecting the lowest distance between any two points in each cluster pair. Here we include a dummy cluster containing only  $q_0$  that is predefined as the start and end of the cluster visiting order to take into account the cost of advancing between the home configuration and the first and last clusters, respectively. Once a visiting order for each cluster is determined, the pairs of points corresponding to the lowest distance between clusters are used as pre-specified entry and exit points for each TSP instance at the intra-cluster level.

Finally, the configuration sequence  $S$  is obtained by aggregating the local sequences of configurations with the visiting order of clusters. This information can then be used to obtain the set of collision-free trajectories by calling an appropriate motion planner for each successive configuration in  $S$ .

### C. Complexity Analysis

Suppose  $R$  is the upper-bound on the number of feasible IK solutions for any input target point  $\mathbf{p}$ . For  $n$  target points, the complexity for computing all IK solutions is  $\mathcal{O}(Rn)$ . For configuration clustering, finding the globally optimum clustering of a set of points using k-means is known to be an NP-hard problem [26]. However, practical implementations of the algorithm commonly define a number of iterations,  $t$ , where each run of the algorithm is initialized with randomized centroids. Thus in practice, the complexity of clustering is  $\mathcal{O}(tkdn)$ , where  $k$  is the number of clusters and  $d$  is the dimensionality of the problem. For configuration selection, let  $K$  be the upper-bound on the number of clusters obtained by clustering. The number of queries to the dissimilarity

<sup>4</sup>For further details on the derivation and explanation of the BIC, we direct readers to [23].

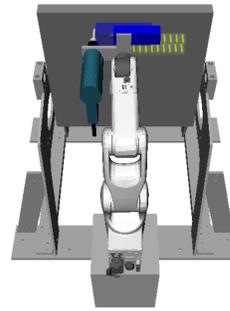


Fig. 3. The Airbus Shopfloor Challenge environment originally used to benchmark RoboTSP [15].

computation in a single iteration gives  $\mathcal{O}(Kn)$ . Since the configuration selection converges logarithmically, the overall complexity of step 2 of our algorithm is  $\mathcal{O}(Kn \log n)$ .

The theoretical proof of the time complexity of the 2-Opt algorithm is complex. Several authors have conducted studies to analyse its complexity under different instances based on the type of distribution of points [19], [20]. We highlight in particular that the analysis presented in [19] derived an upper bound of  $\mathcal{O}(n^{4+\frac{1}{3}})$  for Euclidean instances with a uniform distribution and an arbitrary dimension  $d \geq 2$ . However, as noted by the authors, there is a notable gap between theory and experimental observations. Nevertheless, by applying a CTSP strategy to 2-Opt, the upper bound on step 4 of our algorithm can be approximated as  $\mathcal{O}(km^{4+\frac{1}{3}})$ , where  $m \ll n$ .<sup>5</sup>

The last step of our algorithm has a linear complexity,  $\mathcal{O}(n)$ , that is determined by the computational complexity of motion planning, which varies considerably depending on the spatial constraints present in the environment. We show empirically that for planning problems involving substantial spatial constraints, the majority of computational resources are consumed by motion planning.

## IV. BENCHMARKING IN SIMULATION

In this section we evaluate the performance of our algorithm against the state-of-the-art RoboTSP algorithm [15] as a baseline using simulation-based tasks implemented on OpenRAVE [17]. All experiments were conducted on a system with Intel® Core i5 3320M 2.6 GHz processor with 8 GB RAM.

### A. Benchmarking on Airbus Shopfloor Challenge

We first evaluate our proposed algorithm against RoboTSP on the *Airbus Shopfloor Challenge* task originally used by the authors of RoboTSP to benchmark their algorithm. The task involved drilling a set of holes across a panel mounted on a jig as shown in Fig. 3. While the environment setup exposes the robot to potential collision, the nature of the task to drill along a planar surface means that the robot does not encounter significant spatial constraints that hinder its motion between target points. Hence we consider the spatial constraints for this task to be relatively relaxed. The evaluations were conducted on the Denso VS060 6-DoF industrial robot, with RoboTSP

<sup>5</sup>This approximation assumes an equal distribution of points across clusters.

TABLE I  
PARAMETER SETTINGS FOR ALGORITHM IMPLEMENTATION

RoboTSP		Cluster-RTSP	
Parameter	Value	Parameter	Value
Motion planner	Bi-RRT	Motion planner	Bi-RRT
TSP metric	Euclidean (task space)	TSP metric	Euclidean (C-space)
C-space metric	Max joint difference	Config. select metric	Weighted Sq. Euclidean
-	-	Clustering metric	Sq. Euclidean
-	-	$b_\delta, b_0$	{0.9, 0.1}
-	-	$K_{min}, K_{max}$	{3, 40}

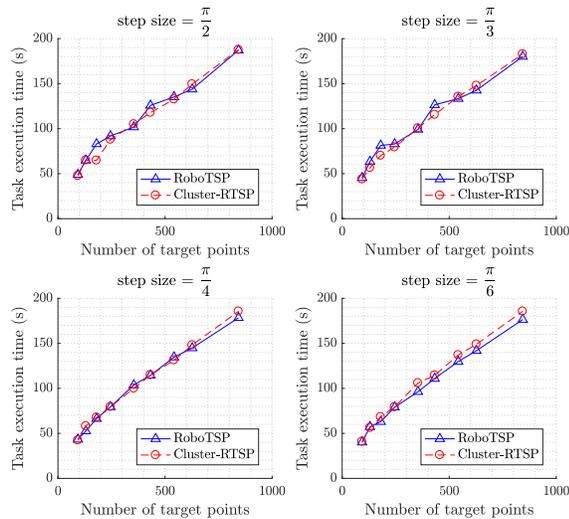


Fig. 4. Task execution time for RoboTSP and Cluster-RTSP algorithms applied to the Airbus Shopfloor Challenge.

implemented using the open-source package developed by the original authors. Parameter values used for each algorithm are shown in Table I. While both algorithms utilise the Bi-RRT for motion planning, the random effects due to the stochastic nature of the algorithm were reduced by resetting the random seed for sample generation during each motion planning query. Furthermore, trajectory smoothing was applied to the Bi-RRT path to increase the likelihood of returning an optimal solution.

In many industrial applications such as inspection and drilling, the yaw angle of the end effector at each target point is typically unspecified. Hence the DoF corresponding to the end effector yaw rotation is often treated as a free DoF. Indeed there can be infinitely many configurations for any given target point as a result. To address this redundancy in a consistent way, we adopt the strategy of discretizing the 6<sup>th</sup> DoF,  $q_6$ , to generate a finite set of IK solutions as used in the RoboTSP algorithm. This was achieved by assigning fixed values to  $q_6$  and, for each value, the IK solutions were computed for the remaining 5 DOFs as explained in [15]. The set of fixed values were generated between  $[0, 2\pi]$  for a specified discretization step size. OpenRAVE’s IKFast module [17] was used to generate this discretised set of IK solutions.

We conduct a number of trials consisting of {93, 130, 180, 245, 354, 432, 542, 627, 843} points for discretization

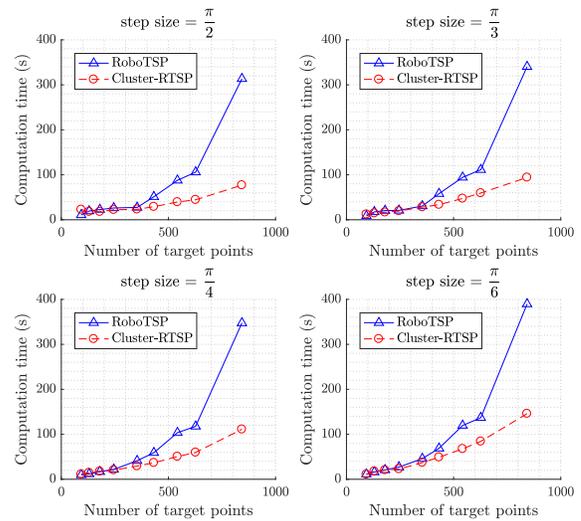


Fig. 5. Computation time for RoboTSP and Cluster-RTSP algorithms applied to the Airbus Shopfloor Challenge.

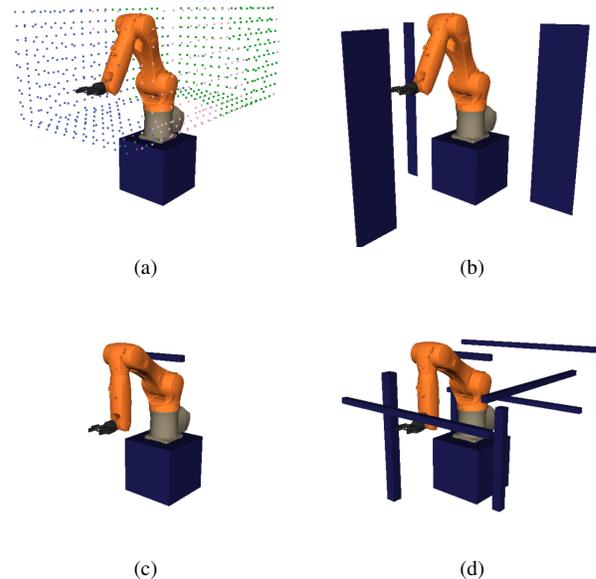


Fig. 6. Environments used for evaluating the performance of RTSP algorithms. (a) Environment A - clutter-free and shown with sample input target points, (b) Environment B - contains three planar obstacles, (c) Environment C - contains a single bar obstacle limiting motion of joint two, (d) Environment D - cluttered environment imposing significant spatial constraints on robot.

step sizes of  $\frac{\pi}{2}$ ,  $\frac{\pi}{3}$ ,  $\frac{\pi}{4}$  and  $\frac{\pi}{6}$  using both the RoboTSP and Cluster-RTSP algorithms. The resulting task execution times for all trials are shown in Fig. 4. In all trials we observe that our proposed algorithm provides comparable performance with RoboTSP, which has already been benchmarked against other existing methods for solving RTSP. However, as the discretization step size increases, the quality of the solution returned by Cluster-RTSP marginally deviates from the baseline, which is representative of the algorithm’s lack of optimality guarantee. We discuss this further in Section VI. Nevertheless, we conclude that for planning problems involving relaxed spatial constraints, the Cluster-RTSP is able to find plans of comparable quality to existing methods in

literature irrespective of the number of input target points.

To benchmark the speed of the solvers, the computation time required for solving each problem instance is shown in Fig. 5. For small problems (<300 points) both algorithms required approximately the same computation time. However, as the number of target points increased, RoboTSP began to consume significantly greater computation time compared to Cluster-RTSP. For example, for 843 points with a discretization step size of  $\frac{\pi}{2}$ , Cluster-RTSP solved the problem 75.5% faster. As RoboTSP had already been shown to outperform existing approaches in terms of planning speed, we conclude that the Cluster-RTSP is a competitive solver for RTSPs.

### B. Benchmarking on Environments A-D

While Section IV-A shows that our proposed algorithm produces solutions of comparable quality in relaxed problems, here we consider the algorithm's performance under different types of spatial constraints. Using the environments shown in Fig. 6, we test the behaviour of the algorithm for solving RTSPs that involve a combination of the following: (i) target points distributed across multiple planes, (ii) planar obstacles that do not invalidate many IK solutions but obstruct linear motion between neighbouring points, (iii) obstacles located close to the robot that substantially reduce  $C_{free}$ , and (iv) highly cluttered environments posing significant spatial constraints.

We run both algorithms on problems consisting of {220, 326, 554, 747, 907, 1379, 1816} input target points for environments A-D with the discretization step size set to  $\frac{\pi}{3}$ . Fig. 7 shows the task execution times for all trials conducted. In environments A and B, the algorithms produced solutions of comparable quality, which is in agreement with the results in Section IV-A that considered relatively relaxed problems. In contrast, Cluster-RTSP outperformed RoboTSP for all trials conducted in environment C, with an improvement of up to 14.5% shorter solutions. Likewise for environment D, Cluster-RTSP consistently achieved better solutions than RoboTSP, with a maximum reduction of 22.2% in task execution time for 1095 visited target points. These results show that RoboTSP can indeed miss optimal solutions for problems that involve spatial constraints. Conversely, the Cluster-RTSP algorithm is able to find better solutions as it makes informed selections of configurations relative to the entire task. By solving the TSP in C-space, the algorithm considers the actual robot states required to reach each target point, which provides a better approximation to the cost of required motions between target points for sequencing compared to the task space counterpart.

Fig. 8 reports the computation time required by each algorithm to solve the RTSPs. Clearly, the Cluster-RTSP achieved comparatively faster planning times for all environments, particularly when the number of target points increased. For example, our proposed algorithm was able to reduce the computation time by 58.5% in environment A when  $n = 1816$ . In Table II, the motion planning success rate and average computation time for a single motion planning query are reported for both algorithms. While the average computation times for motion planning were comparable between the two algorithms for Environments A and B (varying by 10-20 ms), we observe that the computational cost of motion

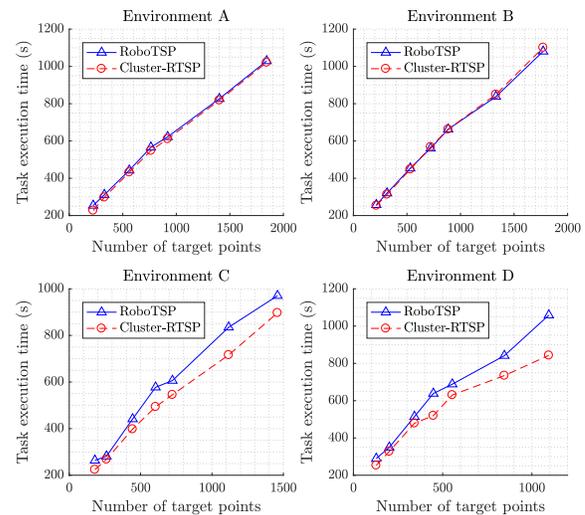


Fig. 7. Task execution time for problem instances shown in Fig. 6.

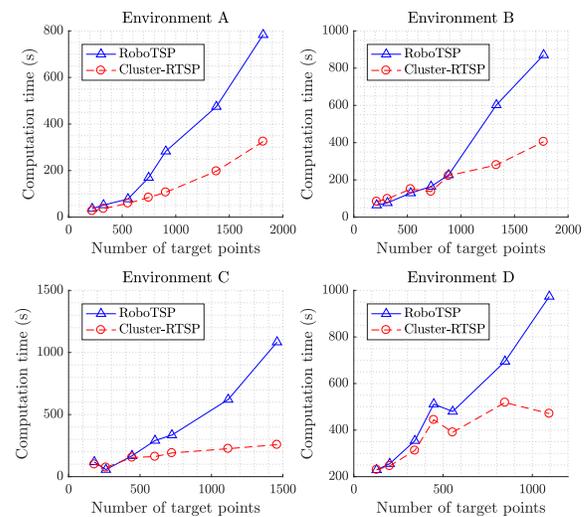


Fig. 8. Computation time for problem instances shown in Fig. 6.

planning is approximately 100 ms less for Cluster-RTSP in Environments C and D. This can be explained by the higher quality task sequence obtained by Cluster-RTSP, which subsequently reduces the number of complex motions required to move between successive configurations. These are generally more costly to compute. Looking at the success rate, both algorithms had encountered a failure in motion planning for Environment D. Indeed, a limitation of both algorithms is the absence of reachability analysis to evaluate whether assigned configurations can be reached prior to motion planning. This is discussed in more detail in Section VI.

Fig. 9 gives a breakdown of the computation time required for each step of the Cluster-RTSP algorithm across all trials. Importantly, the computation times required by the configuration selection and CTSP procedures vary according to the number of reachable targets points (and IK solutions), while the computation time for motion planning varies according to the complexity of the environment. In all trials, the time required for clustering was negligible in comparison.

TABLE II  
MOTION PLANNING PERFORMANCE FOR 1816 TARGET POINTS

Environment:		A	B	C	D
Success rate (%)	RoboTSP	100.0	100.0	100.0	99.9
	Cluster-RTSP	100.0	100.0	100.0	99.9
Mean CPU time per query (ms)	RoboTSP	53	86	175	433
	Cluster-RTSP	33	96	74	339

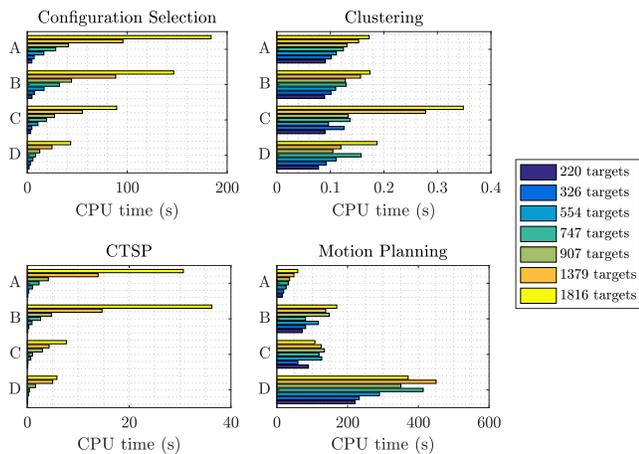


Fig. 9. Breakdown of computation time for the Cluster-RTSP algorithm.

Finally, we observe that the computation time for Environment D does not monotonically increase with the number of target points. We speculate that this is caused by the varying distribution of target points across trials. Each set of target points were generated by adjusting the uniform spacing between points within defined task space boundaries. In a cluttered environment, this can produce numerous points that lie very close to obstacles, which leads to added complexity in motion planning (as reported in Fig. 9).

### V. CASE STUDY: SURFACE INSPECTION OF PIPES

Following the experiments conducted in simulation, we now present a case study consisting of a mock-up of a physical pipe surface scanning task. The inspection task consists of four hollow pipes whose outer surfaces should be inspected by a tool mounted on the end effector of a KUKA KR 6 R900 sixx manipulator. These pipes were placed on a level surface within the workspace of the robot as shown in Fig. 10. In addition to avoiding collision with these pipes, the robot was mounted within an enclosed cell that introduced additional spatial constraints. Several sets of inspection points were generated by equally distributing points across the surfaces of the pipes with defined separation distances and an approach direction normal to the surface. These correspond to  $n = \{425, 645, 948, 1499\}$  reachable inspection points, respectively. We formulate an RTSP comprising of the inspection points as target points to be visited and solved for a configuration sequence using both the Cluster-RTSP and RoboTSP algorithms as before.

We deploy the obtained solutions on the physical robot using the *ITRA toolbox* developed for KUKA KR C4 controllers described in [27]. Specifically, we use the computer approach

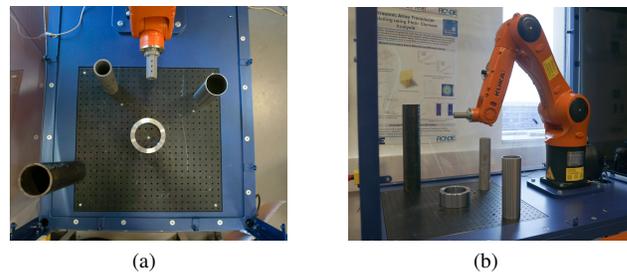


Fig. 10. Physical setup for surface inspection of pipes task. (a) Top-down view, (b) side view.

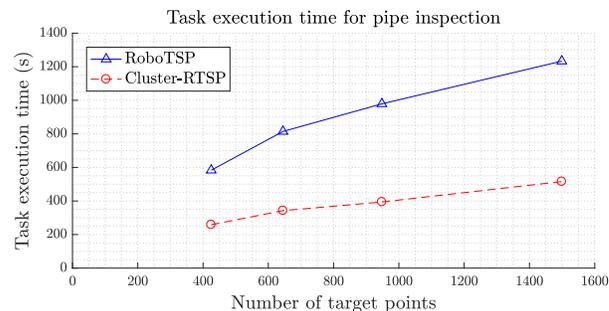


Fig. 11. Task execution times for the pipe surface inspection task.

for external control, which updates the robot command position (specified in joint space) along the trajectory at 12 ms interpolation cycles. When computing robot trajectories in the motion planning stage, we limit the robot’s joint velocity and acceleration limits to 50% of its rated maximum according to the manufacturer’s specification.

Fig. 11 plots the execution times obtained from both algorithms. As this shows, the Cluster-RTSP was able to reduce the execution time by up to 60% across the four trials conducted in comparison to RoboTSP. This level of performance was achieved by identifying the minority of points with assigned configurations that were drastically different from the default “elbow-up” configurations seen in Fig. 10b. Our algorithm clustered these points together and visited them contiguously while RoboTSP found itself alternating frequently between these two groups of points.<sup>6</sup>

Fig. 12 shows a breakdown of the computation time required by each major component of the respective algorithms. This includes solving the TSP, motion planning for pose-to-pose motions along the sequence, and miscellaneous operations unique to each algorithm. Here the computation time for solving the TSP was almost negligible for Cluster-RTSP, while the computation time for RoboTSP was dominated by this step for large values of  $n$ . Additionally, as Table III further shows, the average computation time for a single motion planning query was substantially longer for RoboTSP. Like before, this was due to the increased number of complex motion planning queries that arose from a sub-optimal sequence in a cluttered environment. A further consequence of this was an increase in motion planning failures due to time-out of the Bi-RRT

<sup>6</sup>A video showing the respective solutions of each algorithm executed on the physical setup is available at: <https://youtu.be/3PolyxXkWPk>.

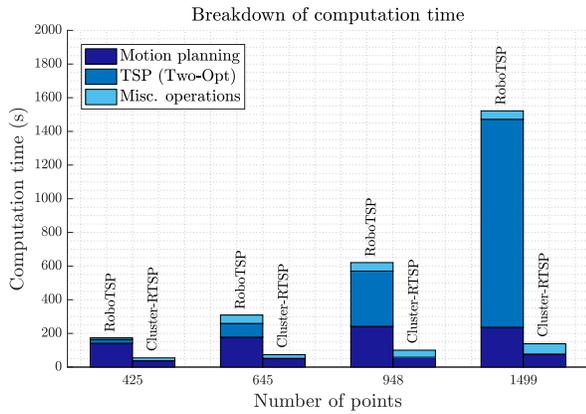


Fig. 12. Breakdown of computation time for the pipe surface inspection task.

TABLE III  
MOTION PLANNING PERFORMANCE FOR PIPE INSPECTION

		# of targets:	425	645	948	1499
Success rate (%)	RoboTSP		100.0	98.6	99.4	99.8
	Cluster-RTSP		100.0	<b>100.0</b>	<b>99.9</b>	<b>99.9</b>
Mean CPU time per query (s)	RoboTSP		0.332	0.279	0.257	0.158
	Cluster-RTSP		<b>0.087</b>	<b>0.080</b>	<b>0.060</b>	<b>0.049</b>

planner. Overall we observed reductions of up to 90.9% in total computation time for Cluster-RTSP. Combining these improvements with the reduction in task execution time show that the proposed algorithm is capable of reducing the absolute time required to complete a task by up to several folds (e.g. for  $n = 1499$  inspection points, the Cluster-RTSP method saved 2101.3 seconds from planning through to execution). This is particularly important for one-off applications where extensive offline planning is costly.

In the remainder of this section, we briefly discuss a number of real-world implementation considerations relevant to practitioners. To give context to this discussion, we present trajectory tracking results that were obtained by deploying the solution from Cluster-RTSP to the robot via the ITRA toolbox. To assess the tracking performance of the test system across its full working range, we vary the maximum joint velocity and acceleration limits of the robot from 10% through to 100% for  $n = 425$  inspection points. During these trials, we considered three aspects of error:

- Max joint error - the joint error in degrees given as the maximum difference between any individual joint
- Total joint error - the joint error in degrees given by the Euclidean metric in C space
- Total position error - the position error of the end effector given as the Euclidean distance in task space.

Fig. 13 plots the maximum tracking error recorded between the executed trajectory and the sent trajectory as the velocity and acceleration limits were varied across its full range. The largest tracking errors were recorded for velocity and acceleration limits below 50%, where the total position error reached approximately 15.2 millimetres, while the max joint error and total joint error reached  $2.8^\circ$  and  $3.7^\circ$ , respectively.

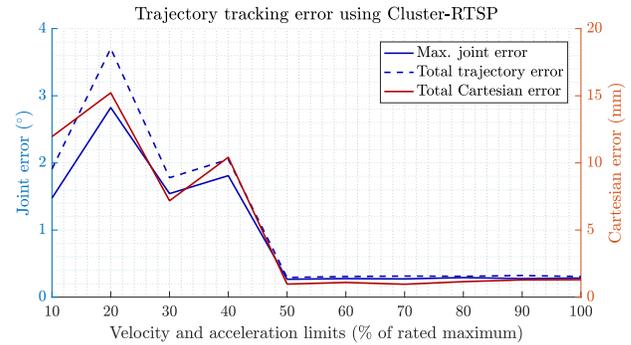


Fig. 13. Trajectory tracking error for pipe surface inspection experiments.

First of all, these errors were unrelated to the solution of the Cluster-RTSP. Unlike existing methods of solving RTSP in task space, where manipulability and singularity occurrences must be explicitly accounted for [8], the Cluster-RTSP avoids reaching singularities by planning entirely in the C-space. Evidently then, these positioning errors arose from the limitations of hardware used in our test system. While efforts to advance control systems can mitigate these errors to some extent, considerations at the planning level can further reduce the risk of unexpected events occurring (such as collision) for applications that cannot tolerate these degrees of errors.

In applications that demand a minimum clearance from obstacles, a common practice used in the robotics community is the inflation of obstacles, either by preserving its 3D form during enlargement or by applying a bounding box representation. An example can be found in [28], where the use of voxels to represent obstacles in the 3D environment enabled inflation to take place by appending additional voxels in the horizontal and vertical directions to enhance navigational safety. In contrast, Chen et al. [29] used an axis-aligned bounding box (AABB) tree to approximate obstacles represented by a point cloud, where nodes higher up in the tree level corresponded to AABBs with greater levels of obstacle clearance. Indeed such approaches are considered conservative and could introduce narrow passages in the C-space, leading to increased computational cost for motion planning. As we have shown, the Cluster-RTSP algorithm is able to handle cluttered environments comparatively well as it is able to efficiently maintain a high-quality task sequence that reduces the number of complex motion planning queries. This is not equally true for existing solvers that are not informed by spatial constraints when developing task sequences, while others have not been proven to do so efficiently. However, inflating obstacles in this way can lead to an increased likelihood of failure in finding a feasible trajectory during motion planning (see Section VI).

When considering the interaction of the robotic end effector with a workpiece, it may be inappropriate to virtually inflate the workpiece to avoid minor collisions resulting from tracking error (the robot may need to move close to the surface to accomplish its task). To address this, a sufficient standoff distance (determined by the maximum Cartesian error of the system) may be applied to all target points such that the robot does not immediately make contact with the surface of the

workpiece. Where contact is then required for a given task (e.g. drilling or contact-based inspection), force sensing could be used for fine precision in achieving contact with the workpiece.

## VI. DISCUSSION

The proposed Cluster-RTSP algorithm possesses several limitations in its current implementation. Firstly, distance metrics are used to estimate the closeness of configurations in the C-space, which provide an estimate of the true cost of motion between any two configurations. However, these metrics only represent the likelihood of two configurations being connected in the C-space [30]. In cluttered environments comprising of many obstacles, the C-space may be composed of disconnected regions for which no feasible path exists between two configurations. The current implementation of the configuration selection procedure (Section III-B1) fails to identify such configurations. This remains a challenge in RTSP as explicit planning is generally required to determine the connectivity of configurations. Yet the sheer number of configurations considered in a typical RTSP<sup>7</sup> renders this a highly inefficient solution. Mapping the free regions of the C-space (e.g. [31]) prior to planning could provide a viable approach to evaluating the connectivity of configurations in configuration selection, but this inadvertently introduces a computationally costly preprocessing step.

Due to the varying topology and complexity of C-spaces across different robotic applications, the adoption of the weighted squared Euclidean distance for computing the similarity measure in step 2 of the algorithm (Section III-B1) may not necessarily be the best selection criteria. As we saw in Fig. 4, the algorithm consistently found a marginally poorer solution than the baseline as the number of discrete values for the free DOF increased, suggesting that the chosen distance metric does not effectively differentiate the optimal configuration from among multiple ‘good’ candidates. The authors in [32] studied the effects of different metrics on the behaviour of a 7-DoF robot and found that some metrics such as the Euclidean distance were more effective for contraction tasks while others performed well for expansion tasks. Similarly, the authors of [30] found that different distance metrics (when used in conjunction with different local planners) provided different connectivity in the construction of probabilistic roadmaps for path planning. These findings suggest that no metric is universally superior across all robot tasks. Further study into the effects of different metrics for specific RTSP applications could provide greater insight into the effectiveness of certain metrics for specific tasks. For example, evaluating the shortest distance of a robot to nearby obstacles at each configuration may provide a selection of configurations that maximises the likelihood of finding feasible paths in highly cluttered environments (but this would likely incur additional computation time).

In the current implementation of the algorithm, any failures in computing a feasible trajectory during motion planning are handled by simply skipping the configuration and moving to

<sup>7</sup>Up to 96 configurations were found per target point in the experiments conducted in this work.

the next target point in the sequence.<sup>8</sup> Indeed by addressing the limitations discussed above, the probability of failure in motion planning could be drastically reduced or eliminated entirely. Alternatively, an ad-hoc solution could involve iterating through each of the originally dismissed configurations to determine a new candidate configuration (if it exists) when an instance of motion planning fails.

Lastly, this work has so far demonstrated the effectiveness of formulating the RTSP as a two-layer CTSP. In the future, an in-depth study on how additional layers of clustering might further improve computational performance (and whether this would impact solution quality) could provide greater insight into the use of clustering techniques within this domain.

## VII. CONCLUSION

In this paper we have presented a novel clustering-based algorithm for solving spatially-constrained robotic task sequencing problems. Through detailed benchmarking, we have evaluated the competitiveness of our algorithm in terms of computation speed and the quality of solutions obtained. We have addressed challenging problems consisting of the combination of large sets of target points and substantial spatial constraints, which, to our knowledge, no other work in RTSP to date has considered before. In particular, we have shown that our algorithm significantly outperforms a state-of-the-art approach when applied to these problems, with up to 60% reduction in task execution time and 91% reduction in computation time observed in our experiments.

The capabilities of our algorithm are relevant to many industrial tasks such as free-form surface inspection, surface profiling, drilling and screw fastening. Indeed for those applications where extensive offline planning is too costly, our approach offers significant benefit by providing near-optimal solutions within minutes. Through a case study involving the deployment of our algorithm on a common industrial robot, we have further discussed some implementation considerations for applications where precision is an important factor.

## REFERENCES

- [1] D. F. Elkott, H. A. Elmaraghy and W. H. Elmaraghy, "Automatic sampling for CMM inspection planning of free-form surfaces", *Int. Journal of Production Research*, 2002, vol. 40, no. 11, pp. 2653-2676, doi: <https://doi.org/10.1080/00207540210133435>.
- [2] M. P. Mali, and K. H. Inamdar, "Optimization of spot welding process using digital manufacturing", *International Archive of Applied Sciences and Technology*, Jun. 2013, vol. 4, no. 2, pp. 27-35.
- [3] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms", *European Journal of Operational Research*, 1992, vol. 59, pp. 231-247, doi: [https://doi.org/10.1016/0377-2217\(92\)90138-Y](https://doi.org/10.1016/0377-2217(92)90138-Y).
- [4] S. Dubowsky and T. Blubaugh, "Planning time-optimal robotic manipulator motions and work places for point-to-point tasks". *IEEE Trans. Robotics and Automation*, 1989 vol. 5, pp. 377-381, doi: <https://doi.org/10.1109/70.34775>.
- [5] L. L. Abdel-Malek and Z. Li, "The application of inverse kinematics in the optimum sequencing of robot tasks", *International Journal of Production Research*, 1990, vol. 28, no. 1, pp. 75-90, doi: <https://doi.org/10.1080/00207549008942685>.
- [6] E. Kolakowska, S. F. Smith, and M. Kristiansen, "Constraint optimization model of a scheduling problem for a robotic arm in automatic systems", *Robotics and Autonomous Systems*, 2014, vol. 62, no. 2, pp. 267-280, doi: <https://doi.org/10.1016/j.robot.2013.09.005>.

<sup>8</sup>These failures are reported to the user on the host PC.

- [7] P. Zacharia and N. Aspragathos, "Optimal robot task scheduling based on genetic algorithms", *Robotics and Computer-Integrated Manufacturing*, 2005, vol. 21, no. 1, pp. 67-79, doi: <https://doi.org/10.1016/j.rcim.2004.04.003>.
- [8] A. Kovács, "Integrated task sequencing and path planning for robotic remote laser welding", *International Journal of Production Research*, June 2015, vol. 54, no. 4, pp. 1210-1224, doi: <https://doi.org/10.1080/00207543.2015.1057626>.
- [9] W. Jing et al., "Model-based coverage motion planning for industrial 3D shape inspection applications", in *Proc. 2017 13th IEEE Conf. Automation Science and Engineering (CASE)*, Xi'an, China, pp. 1293-1300, doi: <https://doi.org/10.1109/COASE.2017.8256278>.
- [10] W. Jing et al., "Sampling-based coverage motion planning for industrial inspection application with redundant robotic system", in *2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vancouver, Canada, pp. 5211-5218, doi: <https://doi.org/10.1109/IROS.2017.8206411>.
- [11] S. N. Spitz and A. A. G. Requicha, "Multiple-goals path planning for coordinate measuring machines", in *Proc. 2000 IEEE Int. Conf. Robotics and Automation (ICRA)*, San Francisco, CA, USA, pp. 2322-2327, doi: <https://doi.org/10.1109/ROBOT.2000.846373>.
- [12] E. K. Xidias, P. T. Zacharia, and N. A. Aspragathos, "Time-optimal task scheduling for articulated manipulators in environments cluttered with obstacles", *Robotica*, May 2010, vol. 28, no. 3, pp. 427-440, doi: <https://doi.org/10.1017/S0263574709005748>.
- [13] P. T. Zacharia, E. K. Xidias, and N. A. Aspragathos, "Task scheduling and motion planning for an industrial manipulator", *Robotics and Computer-Integrated Manufacturing*, Dec 2013, vol. 29, no. 6, pp. 449-462, doi: <https://doi.org/10.1016/j.rcim.2013.05.002>.
- [14] L. B. Gueta, R. Chiba, J. Ota, T. Ueyama and T. Arai, "Coordinated motion control of a robot arm and a positioning table with arrangement of multiple goals", in *Proc. 2008 IEEE Int. Conf. Robotics and Automation (ICRA)*, Pasadena, CA, USA, pp. 2252-2258, doi: <https://doi.org/10.1109/ROBOT.2008.4543549>.
- [15] F. Suárez-Ruiz, T. S. Lembono, Q-C. Pham, "RoboTSP - A fast solution to the robotic task sequencing problem", in *Proc. 2018 IEEE Int. Conf. Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, pp. 1611-1616, doi: <https://doi.org/10.1109/ICRA.2018.8460581>.
- [16] S. L. Villumsen and M. Kristiansen, "A framework for task sequencing for redundant robotic remote laser processing equipment based on redundancy space sampling", *Procedia Manufacturing*, 2017, vol. 11, pp. 1826-1836, doi: <https://doi.org/10.1016/j.promfg.2017.07.320>.
- [17] R. Diankov, J. Kuffner, "Openrave: A planning architecture for autonomous robotics", *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008, vol. 79.
- [18] J. J. Kuffner and J. M. Steven LaValle, "RRT-Connect: An efficient approach to single-query path planning", in *Proc. 2000 IEEE Int. Conf. Robotics and Automation (ICRA)*, San Francisco, CA, USA, pp. 995-1001, doi: <https://doi.org/10.1109/ROBOT.2000.844730>.
- [19] M. Englert, H. Röglin and B. Vöcking, "Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP", *Algorithmica*, 2014, vol. 68, no. 1, pp. 190-264, doi: <https://doi.org/10.1007/s00453-013-9801-4>.
- [20] J. J. A. Slootbeek, "Average-case analysis of the 2-opt heuristic for the TSP", M.S. Thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, Univ. of Twente, Enschede, Netherlands, 2017. [Online]. Available: [https://essay.utwente.nl/72060/1/Slootbeek\\_MA\\_EEMCS.pdf](https://essay.utwente.nl/72060/1/Slootbeek_MA_EEMCS.pdf)
- [21] D. Pelleg, A. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters", in *Proc. 17th Int. Conf. Machine Learning*, Stanford, CA, USA, Jun./Jul. 2000, pp. 727-734.
- [22] R. E. Kass and L. Wasserman, "A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion", *Journal of the American Statistical Association*, 1995, vol. 90, no. 431, pp. 928-934, doi: <https://doi.org/10.1080/01621459.1995.10476592>.
- [23] B. Hancock, D. Frank, A. Foglia, and R. Yozzo, "Notes on Bayesian information criterion calculation for X-means clustering", 2012. Accessed: Jul. 28, 2019. [Online]. Available: [https://github.com/bobhancock/goxmeans/blob/master/doc/BIC\\_notes.pdf](https://github.com/bobhancock/goxmeans/blob/master/doc/BIC_notes.pdf)
- [24] J. A. Chisman, "The clustered traveling salesman problem", *Computers & Operations Research*, Sep. 1975, vol. 2, no. 2, pp. 115-119, doi: [https://doi.org/10.1016/0305-0548\(75\)90015-5](https://doi.org/10.1016/0305-0548(75)90015-5).
- [25] G. A. Croes, "A method for solving traveling-salesman problems", *Operations Research*, Dec. 1958, vol. 6, no. 6, pp. 791-908, doi: <https://doi.org/10.1287/opre.6.6.791>.
- [26] D. Aloise, A. Deshpande, P. Hansen and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering", *Machine Learning*, Jan. 2009, vol. 75, no. 2, pp. 245-248, doi: <https://doi.org/10.1007/s10994-009-5103-0>.
- [27] C. Mineo, C. Wong, M. Vasilev, B. Cowan, C. Macleod, S. G. Pierce and E. Yang, "Interfacing Toolbox for Robotic Arms with Real-Time Adaptive Behavior Capabilities", *University of Strathclyde*, 2019, doi: <https://doi.org/10.17868/70008>.
- [28] L. v. Stumberg, V. Usenko, K. Engel, K. Stuckler and D. Cremers, "From monocular SLAM to autonomous drone exploration", in *2017 European Conf. on Mobile Robots (ECMR)*, Paris, France, doi: <https://doi.org/10.1109/ECMR.2017.8098709>.
- [29] G. Chen, D. Liu, Y. Wang, Q. Jia and X. Zhang, "Path planning method with obstacle avoidance for manipulators in dynamic environment", *Int. Journal of Advanced Robotic Systems*, 2018, doi: <https://doi.org/10.1177/1729881418820223>.
- [30] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones and D. Vallejo, "Choosing Good Distance Metrics and Local Planners for Probabilistic Roadmap Methods", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1998, pp. 630-637, doi: <https://doi.org/10.1109/ROBOT.1998.677043>.
- [31] J. Pan and D. Manocha, "Efficient configuration space construction and optimization for motion planning", in *Engineering*, March 2015, vol. 1, no. 1, pp. 46-57, doi: <https://doi.org/10.15302/J-ENG-2015009>.
- [32] H. J. Jeon, A. D. Dragan, "Configuration space metrics", in *Proc. 2018 IEEE/RSJ Int. Conf. Robotics and Systems (IROS)*, Madrid, Spain, pp. 5101-5108, doi: <https://doi.org/10.1109/IROS.2018.8593564>.



**Cuebong Wong** is a Robotics Research Technologist at National Nuclear Laboratory Ltd. He obtained a Masters with Distinction in Electrical and Mechanical Engineering at the University of Strathclyde (UK) in 2016 and subsequently joined the Department of Design, Manufacture and Engineering Management (DMEM) to undertake a PhD in Robotics and Autonomous Systems for Harsh Environments, which was funded by the EPSRC Doctoral Training Partnership (DTP) 2016-2017 programme. His current research interests lie in adaptive task planning,

motion planning and manipulation for robotic manipulators and mobile robots that interact with challenging environments.



**Carmelo Mineo** is a Member ASME and a Member of British Institute of Non-Destructive Testing. He received a B.Eng and M.Eng in Mechanical Engineering at the University of Palermo in 2009 and 2011, respectively. In 2012, he joined the Centre for Ultrasonic Engineering at the University of Strathclyde to undertake a PhD in Automated Non-Destructive Inspection of Large and Complex Geometries of Composite Materials. He became a Research Associate of the University of Strathclyde in 2015 and a Research Fellow in 2018. In 2020,

Carmelo was awarded a two-year Marie-Curie Fellowship from the European Commission to lead research on Robotic Adaptive Behaviours for NDT Inspections in Dynamic Contexts with the University of Palermo.



**Erfu Yang** received the B.Eng. and M.Eng. degrees both in Aerospace Propulsion Theory and Engineering, from Beijing University of Aeronautics and Astronautics, China, and the Ph.D. degree in Robotics from the University of Essex, UK, in 2008. He is currently a Lecturer under Strathclyde Chancellor's Fellowship Scheme at the University of Strathclyde, Glasgow, UK. His main research interests include robotics, autonomous systems, mechatronics, manufacturing automation, computer vision and applications of machine learning and artificial intelligence,

etc. He is the Fellow of the UK Higher Education Academy, committee member of the Chinese Automation and Computing Society in the UK (CACSU), and the IET SCOTLAND Manufacturing Technical Network.



**Xiu-Tian Yan** is a Professor in Mechatronic Systems Technology and the Director of Space Mechatronic Systems Technology Laboratory (SMeSTech), of the University of Strathclyde. He received his PhD from Loughborough University of Technology. He is a Fellow of Institution of Engineering and Technology and a Fellow of IMechE. He is a Technical Editor of IEEE/ASME Transactions on Mechatronics. Prof Yans research interests include mechatronic systems design, robotics, computer support mechatronic systems design, knowledge intensive product modelling and simulation, the proactive computer support of product life-cycle synthesis, and modular design. He has published over 240 technical papers in major journals and conferences and edited/wrote 6 books in the fields.



**Dongbing Gu** (Senior Member of IEEE) is a Professor in the School of Computer Science and Electronic Engineering, University of Essex. He received the B.Sc. and M.Sc. degrees in automatic control from Beijing Institute of Technology, Beijing, China and the Ph.D. degree in robotics from University of Essex, UK. His research expertise is in the field of robotics, autonomous systems, navigation and control, mapping and localisation, cooperative control, and machine learning. His research has been supported by UK Research Councils (EPSRC), InnovateUK, Royal Academy of Science, European Commission, and industry. He is an Editorial Board Member of Cognitive Computation, Frontiers in Robotics and AI: Multi-robot systems.