2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Cycle Time based Multi-Goal Path Optimization for Redundant Robotic Systems

Iacopo Gentilini, Kenji Nagamatsu, and Kenji Shimada

Abstract— **Finding an optimal path for a redundant robotic system to visit a sequence of several goal placements poses two technical challenges. First, while searching for an optimal sequence, infinitely many feasible configurations can be used to reach each goal placement. Second, obstacle avoidance has to be considered while optimizing the path from one goal placement to the next. Previous works focused on solving a discrete formulation of this optimization problem where only few configurations are used to represent each goal placement. We instead model it as a Traveling Salesman Problem with Neighborhoods (TSPN), where each neighborhood is defined as the set of the infinitely many configurations corresponding to the same goal placement. A solution procedure based on a Hybrid Random-key Genetic Algorithm (HRKGA) and bidirectional Rapidly-exploring Random Trees (biRRTs) is then proposed. Finally, experimental tests performed on a 7-Degree Of Freedom (DOF) industrial vision inspection system show that the proposed method is able to drastically reduce the cycle time currently required by the system.**

I. Introduction

Industrial manipulators can be used to perform a sequence of multiple tasks during an operating cycle. If the robotic system is redundant there is an infinite number of possible configurations that can be used to place the manipulator end-effector at the specific position and orientation, i.e., *goal placement*, required to perform each task. In order to optimize the overall cycle time, not only an optimal sequence of the goals has to be defined, but also, for each goal, an optimal configuration has to be chosen among infinitely many possibilities. Moreover, the actual cost for the system to move from one configuration to the next depends on many factors, such as obstacle avoidance or joint limitations, and can be calculated only through the employment of specific path planning techniques.

A formulation of this optimization problem based on the classic Traveling Salesman Problem (TSP) can not fully capture its mixed combinatorial and continuous nature. Indeed, due to the redundancy in the system each vertex of the graph, i.e., the configuration corresponding to each goal placement, is not fixed but can move within a continuous domain, called *neighborhood*. Therefore, not only an optimal sequence has to be found that visits each neighborhood once, but also the optimal position of each vertex in its neighborhood has to be defined, as shown in Figure 1. This problem is commonly referred to as the TSP with Neighborhoods (TSPN) [1], and the combination of an optimal sequence and optimal vertices is called *optimal tour*.

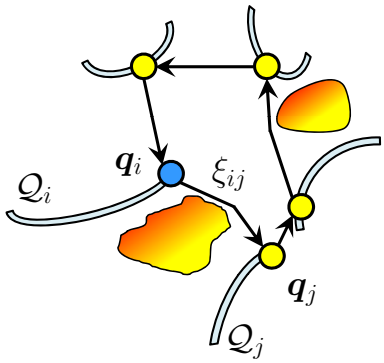


Fig. 1. Collision-free near optimal tour for a TSPN instance. Obstacles are shaded in red/yellow and one-dimensional neighborhoods are shaded in bright blue.

Heuristic approaches for solving a discretized version of this optimization problem are proposed in the robotics literature. In [2], a small set of discrete samples for each neighborhood is first extracted. A near optimal solution for the resulting Generalized TSP (GTSP) is then calculated by using a minimum group spanning tree as a special case of the Steiner tree problem and by performing a preorder tree walk. In [3], the authors propose to find first a near optimal sequence in three steps: (1) cluster the representative placements in the workspace, (2) solve the resulting TSP in each cluster, and (3) concatenate the resulting paths. Then each neighborhood is sampled, and a configuration is chosen for each neighborhood by combining a greedy nearest neighbor method and the Dijkstra algorithm using a rough-to-smooth procedure. Finally, the problem of finding a solution to the redundant task-constrained path planning is discussed in [4], where the sequence of goals is not considered in the optimization problem but is predefined as an equispaced sampling of the trajectory in the task space.

The cited approaches are limited by the fact that the procedures used to search for a near optimal sequence (if not fixed) and for near optimal configurations are solved separately and not integrated [3], [4]. Moreover, the continuous neighborhoods are replaced with small clusters of nodes [2], [3], resulting in drastic reduction of the opti-

I. Gentilini is with the Department of Aerospace and Mechanical Engineering, Embry-Riddle Aeronautical University, Prescott, AZ 86301 gentilii@erau.edu.

K. Nagamatsu is is with DENSO WAVE, Incorporated, 1, Yoshiike, Agui-cho, Chita-gun, Aichi, Japan, 470-2297 kenji.nagamatsu@denso-wave.co.jp.

K. Shimada is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 shimada@cmu.edu.

mization potential. We instead model the TSPN using a Mixed Integer Non-Linear Programming (MINLP) formulation [5], where the neighborhoods are defined as continuous domains. We then propose a solution procedure based on a Hybrid Random-Key Genetic Algorithm (HRKGA) [6]. Furthermore, bidirectional Rapidly-exploring Random Trees (biRRTs) are embedded in the HRKGA using an ad-hoc single/multiple query planning scheme to ensure the near optimal tour to be collision-free.

Finally the proposed approach is tested on 7-Degree of Freedom (DOF) robotic system used to perform vision inspection of assembled industrial components. The system consists of a 6-DOF robotic manipulator and 1-DOF turntable. Results show that the proposed method is able to improve the traveling time currently required to complete a 32-goal cycle up to 30%.

The rest of the paper is organized as follows. The proposed TSPN formulation and the HRKGA are presented in Section II. The procedure to evaluate the objective function and to calculate a collision-free near optimal tour in Section III. The procedure to define the continuous neighborhoods and the experimental results in Section IV. Finally, Section V contains conclusions and discusses potential future work.

II. TSPN Formulation and Solution Procedure

If m is the dimension of the configuration space, $\mathcal{Q}$, of the robotic system, a Symmetric TSPN (STSPN) instance is given by a set $\mathrm{V} = \{1, 2, \ldots, n\}$ of the indices of the goal placements, by a set $\mathcal{Q}_i \subseteq \mathbb{R}^m$ for $i \in \mathrm{V}$ of neighborhoods, and by a symmetric nonnegative cost function $\mathrm{d}(\boldsymbol{u}, \boldsymbol{v})$ for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^m$, called *edge weighting function*.

As illustrated in [5], the STSPN can be formulated using n configurations $\boldsymbol{q}_i \in \mathbb{R}^m$ for all $i \in V$ and $n(n-1)/2$ binary variables ξ_{ij} for all $i, j \in V$ with $j > i$ such that $\xi_{ij} = 1$ only if neighborhood j is visited just after neighborhood i or viceversa in the tour, otherwise it is zero. Figure 1 illustrates a STSPN instance, and the complete MINLP formulation is given by:

$$\text{minimize}: \quad \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j>i}}^{n} \xi_{ij}\, \mathrm{d}\left(\boldsymbol{q}_i, \boldsymbol{q}_j\right) \;, \tag{1}$$

$$\text{subject to}: \quad \sum_{j=1}^{i-1} \xi_{ji} + \sum_{j=i+1}^{n} \xi_{ij} = 2 \qquad \forall\, i \in \mathrm{V} \;, \tag{2}$$

$$\sum_{i \in \mathrm{S}} \left(\sum_{\substack{j \in \mathrm{V} \setminus \mathrm{S} \\ j<i}} \xi_{ji} + \sum_{\substack{j \in \mathrm{V} \setminus \mathrm{S} \\ j>i}} \xi_{ij} \right) \geq 2$$

$$\forall\, \mathrm{S} \subset \mathrm{V} \setminus \{1\},\, |\mathrm{S}| \geq 3 \;, \tag{3}$$

$$\boldsymbol{q}_i \in \mathcal{Q}_i \subseteq \mathbb{R}^m \qquad \forall\, i \in \mathrm{V} \;, \tag{4}$$

$$\xi_{ij} \in \{0, 1\} \qquad \forall\, i, j \in \mathrm{V},\, j > i \;, \tag{5}$$

$$\boldsymbol{q}_i \in \mathbb{R}^m \qquad \forall\, i \in \mathrm{V} \;. \tag{6}$$

The assignment problem constraints (2) ensure that each vertex is visited exactly once. The subtour elimination constraints (3) ensure that no subtour is present in a solution. Constraints (4) define each neighborhood, i.e., the set of infinitely many configurations corresponding to the same goal placement. It is worth noting that the actual formulation of these constraints depends only on the kinematic characteristics of the redundant robotic system , not on the edge weighting function or metric used in the formulation. Finally, constraints (5) and (6) define the domain of the instance.

The objective function (1) is a non-convex function of both the binary and continuous variables. Therefore, an exact solution procedure is computationally very expensive and only small instances with convex neighborhoods and a norm based edge weighting functions can be efficiently solved to optimality [5]. Some heuristic approaches have been proposed in the literature to deal with larger scale instances but only simple neighborhoods have been used, such as balls in $\mathbb{R}^2$ or $\mathbb{R}^3$ [7]. To overcome these limitations we adapt the HRKGA proposed in [6] to the path planning problem discussed in this work. The HRKGA can handle large scale STSPN instances without any limitation on the nature of the edge weighting function or on the definition of the neighborhoods.

As for standard genetic algorithms, the HRKGA maintains a large pool (*population*) of near optimal tours (*chromosomes*), and low cost operations are performed to improve each chromosome in the pool. Random-key coding is chosen for the chromosomes to guarantee feasibility during crossover operations. Moreover, the convergence rate of the algorithm is drastically improved by replacing mutation operators with two ad-hoc heuristics as illustrated in [6]. First, the position of each vertex is fixed, and their sequence is improved by using the Lin-Kernighan heuristic. Then the touring heuristic optimizes the position of each vertex by solving the Non Linear Programming (NLP) instance resulting from fixing their sequence, i.e. the binary variables, in the original MINLP formulation (1) to (6). In the next Section we show how this standard scheme of the HRKGA can be adapted to the multi-goal path planning problem discussed in this work.

III. Objective function evaluation

A. Traveling time

The edge weighting function, $\mathrm{d}\left(\boldsymbol{q}_i, \boldsymbol{q}_j\right)$, i.e., the traveling time for the robotic system to move from configuration $\boldsymbol{q}_i$ to configuration $\boldsymbol{q}_j$ needs to be calculated to evaluate the objective function (1). Given the simultaneous motion of the joints, a fast estimate can be obtained using the weighted maximum norm [3]:

$$\mathrm{d}\left(\boldsymbol{q}_i, \boldsymbol{q}_j\right) = \max_{k=1,\ldots,m} \left\{ \frac{|q_{i,k} - q_{j,k}|}{\omega_k} \right\} \;, \tag{7}$$

where ω_k is an average velocity set by the controller for joint k. Alternatively, the quadratic norm can be employed:

$$\mathrm{d}\left(\boldsymbol{q}_i, \boldsymbol{q}_j\right) = \sqrt{\left(\boldsymbol{q}_i - \boldsymbol{q}_j\right)^T \mathbf{Q} \left(\boldsymbol{q}_i - \boldsymbol{q}_j\right)}, \tag{8}$$

where $\mathbf{Q}$ is a diagonal matrix with elements $[1/\omega_1^2, \ldots, 1/\omega_m^2]$.

Since the heuristic procedure proposed in this work does not aim to prove optimality, the above two norms are used as surrogate edge weighting functions to efficiently perform crossover and heuristic operations within the HRKGA. However, since the path connecting $\boldsymbol{q}_i$ to $\boldsymbol{q}_j$ might not be a straight line in the configuration space due to obstacle avoidance and to better match the behaviour of the actual system, the final value of objective function (1) is obtained using a more precise model. If we consider a point to point motion from a generic start configuration $\boldsymbol{q}_s$ to a generic goal configuration $\boldsymbol{q}_g$ along the path from $\boldsymbol{q}_i$ to $\boldsymbol{q}_j$, the traveling time can be estimated as:

$$\mathrm{d}\left(\boldsymbol{q}_s, \boldsymbol{q}_g\right) = \max_{k=1,\ldots,m} \begin{cases} \frac{2\alpha_k |q_{s,k} - q_{g,k}| - 2\omega_k^2 + \dot{q}_{s,k}^2 + \dot{q}_{g,k}^2}{2\alpha_k \omega_k} \\ \quad + \frac{2\omega_k - \dot{q}_{s,k} - \dot{q}_{g,k}}{\alpha_k} \\ \qquad \text{if } |q_{s,k} - q_{g,k}| > \frac{2\omega_k^2 - \dot{q}_{s,k}^2 - \dot{q}_{g,k}^2}{2\alpha_k} \\ \frac{\sqrt{4\alpha_k |q_{s,k} - q_{g,k}| + 2\dot{q}_{s,k}^2 + 2\dot{q}_{g,k}^2} - \dot{q}_{s,k} - \dot{q}_{g,k}}{\alpha_k} \\ \qquad \text{otherwise} \end{cases} \tag{9}$$

where α_k is an average acceleration set by the controller for joint k, and $\dot{q}_{s,k}$ and $\dot{q}_{g,k}$ are the initial and final joint velocities, which are considered to be positive in Equation (9). If an inversion in the joint velocity has to be modeled, a middle configuration with zero joint velocity is introduced.

B. Obstacle avoidance

To obtain a realistic evaluation of the path followed by the robotic system to move from configuration $\boldsymbol{q}_i$ to configuration $\boldsymbol{q}_j$ collision avoidance has to be considered. This additional requirement drastically increases the complexity of the optimization problem and requires the employment of an ad-hoc path planning technique.

Probabilistic sampling-based planners have been intensively used to solve single query or multiple query motion planning problems in high dimensional configuration spaces [8]. Single query motion planners are used when the planning procedure has to be performed only once, and the configuration space is explored using a single or a bi-directional tree. Rapidly-exploring Random Trees (RRTs) are an example of this planning approach [9], [10]. When the planning procedure has to be repeated more than once possibly with different start and goal configurations, multiple query motion planners are used. An example of such a planner is the Probabilistic Roadmap Method [11]. Recent attempts have been proposed to integrate these two methods for large scale motion planning. An example is the Sampling-based Roadmaps of Trees (SRT) [12].

In the HRKGA a path planning step has to be executed n times for each chromosome in the population since in each tour there are n distinct edges. Thus, a multiple query planner seems to be the best choice to efficiently handle this large number of requests. However, a preprocessing step to fully explore the collision-free configuration space, $\mathcal{Q}_{free}$, would be extremely expensive in terms of computational cost. Similarly to the SRT approach, single query planners are instead used to simultaneously answer the initial queries and incrementally build a roadmap $G = (V_G, E_G)$ over $\mathcal{Q}_{free}$, which afterwards is used to answer multiple queries. A high-level description of the proposed approach is provided in Algorithm 1.

Algorithm 1 BiRRT based Single and Multiple Query Path Planner for the STSPN.

In: tour $\boldsymbol{q}_{\pi(i)}, i = 1, \ldots, n$, edge weighting function $\mathrm{d}(\cdot)$, indicator matrix M_{conn}, sampling threshold l_{samp}, distance matrix D, and roadmap $G = (V_G, E_G)$
Out: collision-free tour length O, updated D and G

```
1.  O ← 0
2.  for i = 1 to n do
3.    if |V_G| > l_samp and M_conn[π(i), π(i+1)] = 1 then
4.      if q_π(i) ∉ V_G then connect(G, q_π(i))
5.      if q_π(i+1) ∉ V_G then connect(G, q_π(i+1))
6.      v_s ← index(V_G, q_π(i))
7.      v_g ← index(V_G, q_π(i+1))
8.      if v_s ≠ NIL and v_g ≠ NIL then
9.        if D(v_s, v_g) = −1 then
10.         D(v_s, v_g) ← short_path(G, q_π(i), q_π(i+1))
11.       O ← O + D[v_S, v_G]
12.     else
13.       O ← O + ∞
14.   else
15.     V_P ← {q_π(i), q_π(i+1)}
16.     E_P ← ∅
17.     P = (V_P, E_P)
18.     if local_planner(q_π(i), q_π(i+1)) then
19.       E_P ← {(q_π(i), q_π(i+1))}
20.     else
21.       biRRT_planner(P, G, q_π(i), q_π(i+1))
22.     if |E_P| > 0 then
23.       if connect(G, P) then
24.         M_conn[π(i), π(i+1)] ← 1
25.       O ← O + length(P)
26.     else
27.       O ← O + ∞
28. return O
```

First, the neighborhoods sequence is decoded from the chromosome into a permutation $\pi(i)$, with $\pi(n+1) = \pi(1)$. Afterwards, for each edge in each tour, $(\boldsymbol{q}_{\pi(i)}, \boldsymbol{q}_{\pi(i+1)})$, the algorithm checks if the number of configurations in the roadmap, $|V_G|$, is larger than a given threshold, l_{samp}, and if the corresponding neighborhoods $\mathcal{Q}_{\pi(i)}$ and $\mathcal{Q}_{\pi(i+1)}$ have been previously connected at least once by using an indicator matrix M_{conn}. During the initial calls these two conditions are not verified, and thus a single query planner is invoked adding the extracted collision-free path to the roadmap G. Once the roadmap is completed according to the given criteria, then a multi query planner is used and edge

Algorithm 2 Function `biRRT_planner`$(P, G, \boldsymbol{q}_s, \boldsymbol{q}_g)$.

In: collision-free path $P = (V_P, E_P)$,
roadmap $G = (V_G, E_G)$ in $\mathcal{Q}_{free}$,
start configuration $\boldsymbol{q}_s$, goal configuration $\boldsymbol{q}_g$,
first tree $T_1 = (V_{T_1}, E_{T_1})$,
second tree $T_2 = (V_{T_2}, E_{T_2})$,
number of attempts to merge the trees l_{merge}, and
resolution for collision avoidance q_{res}
Out: updated P and G

$V_{T_1} \leftarrow \{\boldsymbol{q}_s\}$; $E_{T_1} \leftarrow \emptyset$
$V_{T_2} \leftarrow \{\boldsymbol{q}_g\}$; $E_{T_2} \leftarrow \emptyset$
for $l = 1$ to l_{merge} **do**
  let $\boldsymbol{q}_{rand}$ be a randomly chosen configuration in $\mathcal{Q}$
  $\boldsymbol{q}_{new,1} \leftarrow$ `extend`$(T_1, \boldsymbol{q}_{rand})$
    if $\boldsymbol{q}_{new,1} \neq$ NIL **then**
      $\boldsymbol{q}_{new,2} \leftarrow$ `extend`$(T_2, \boldsymbol{q}_{new,1})$
        if $\boldsymbol{q}_{new,2} \neq$ NIL and
          $\|(\boldsymbol{q}_{new,1}, \boldsymbol{q}_{new,2})\|_2 < q_{res}$ **then**
          if `local_planner`$(\boldsymbol{q}_{new,1}, \boldsymbol{q}_{new,2})$ **then**
            $P \leftarrow$ `extract_path`$(T_1, \boldsymbol{q}_{new,1},$
              $T_2, \boldsymbol{q}_{new,2})$
            `connect`(G, T_1); `connect`(G, T_2)
            $E_G \leftarrow E_G \cup \{(\boldsymbol{q}_{new,1}, \boldsymbol{q}_{new,2})\}$
            return TRUE
        `swap`(T_1, T_2)
return FALSE

costs calculated using Equation (9) are stored in a sparse distance matrix, D.

1) Single Query Planner and Roadmap Construction: The function `local_planner`, is used to verify if the point to point motion along the line segment defined in the configuration space by the two configurations $\boldsymbol{q}_{\pi(i)}$ and $\boldsymbol{q}_{\pi(i+1)}$ is collision-free.

If the line segment between the two configurations $\boldsymbol{q}_{\pi(i)}$ and $\boldsymbol{q}_{\pi(i+1)}$ is not collision free, then the function `biRRT_planner`, described in Algorithm 2, is used to find a collision-free path between these two configurations. Two RRTs, T_1 rooted at a start configuration $\boldsymbol{q}_s = \boldsymbol{q}_{\pi(i)}$ and T_2 rooted at a goal configuration $\boldsymbol{q}_g = \boldsymbol{q}_{\pi(i+1)}$, are grown towards each other to build the collision-free path. Initially a random configuration, $\boldsymbol{q}_{rand}$, is generated as follows:

$$\boldsymbol{q}_{rand} = \begin{cases} \boldsymbol{q}_l + ((1+2\eta)\,U - \eta)|\boldsymbol{q}_g - \boldsymbol{q}_s| \\ \quad \text{if } l < l_{loc} \text{ or } u < p_{loc} \\ \boldsymbol{q}_{\text{MIN}} + U(\boldsymbol{q}_{\text{MAX}} - \boldsymbol{q}_{\text{MIN}}) \\ \quad \text{otherwise} \end{cases} \tag{10}$$

where $q_{l,k} = \min\{q_{s,k}, q_{g,k}\}$, $\boldsymbol{q}_{\text{MIN}}$ and $\boldsymbol{q}_{\text{MAX}}$ are the joint limits vectors, u is a uniformly distributed number in $[0, 1]$, U is a diagonal $(m \times m)$ matrix with diagonal entries uniformly distributed in $[0, 1]$, the parameter η defines a reduced sampling domain, l_{loc} is the minimum number of configurations sampled from the reduced domain, and p_{loc} is the probability to sample from the reduced domain.

Afterwards, the procedure attempts to merge the two trees using the functions `extend` and `local_planner` until the maximum number of attempts, l_{merge}, is reached.

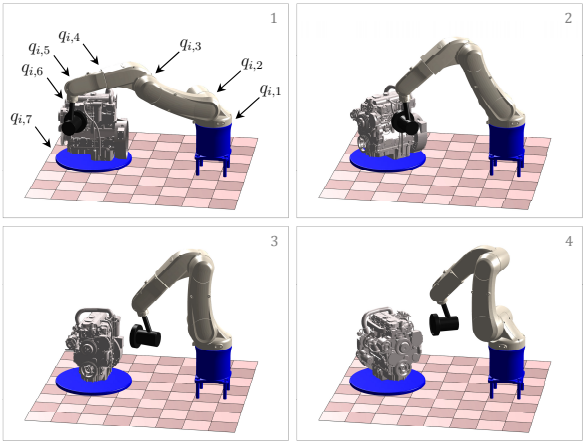


Fig. 2. Robotic vision inspection system: four different configurations for neighborhood $i = 14$ that correspond to the same relative placement of the camera with respect to the component.

If the trees can be merged a path $P = (V_P, E_P)$ is extracted and smoothed by the function `extract_path`. Moreover, the trees are added to the roadmap G by the function `connect`: the root nodes, the last added leaf nodes, and the internal nodes are connected to the corresponding closest node in the roadmap G by using again either the function `local_planner` or, if it fails, the function `biRRT_planner`. Finally, the function `length` returns the path cost calculated using Equation (9).

2) Multiple Query Planner: If the neighborhoods corresponding to the two configurations $\boldsymbol{q}_{\pi(i)}$ and $\boldsymbol{q}_{\pi(i+1)}$ have been connected at least once by the single query planner, and if the number of configurations, $|V_G|$, added to the roadmap, G, is larger than the threshold l_{samp}, then G is used to find a collision-free path between the two configurations.

First, if the two configurations are not in G the function `connect` is used to find a collision-free path between them and the corresponding nearest configurations in G. Then, using the function `index`, which returns the index of a configuration in G or NIL if the configuration is not in G, the corresponding entry in the sparse distance matrix D is checked. If the entry has not been assigned yet, the function `short_path` is used to calculate the shortest path in G between the two configurations. For fast convergence the nearest neighbor and shortest path operations are implemented using kd-trees [13] and the surrogate edge weighting function (7) or (8). Finally, the distance matrix D is updated using the path cost calculated with Equation (9)

IV. Experimental Results

The application analyzed in this work is a robotic system by DENSO WAVE, Inc., used for inspecting assembled industrial components, such as engine blocks or evaporators. The system consists of a 6-DOF robotic manipulator and 1-DOF turntable. The component that has to be inspected is placed on the turntable, and a camera is mounted at the end-effector of the manipulator. The goal is to rapidly

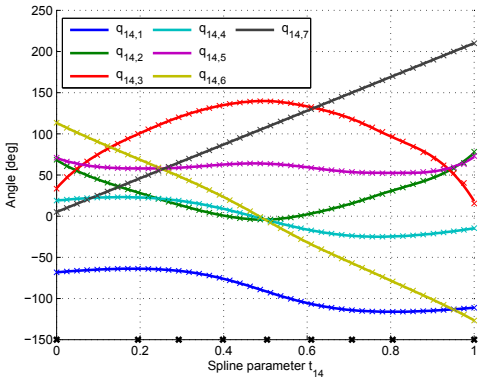
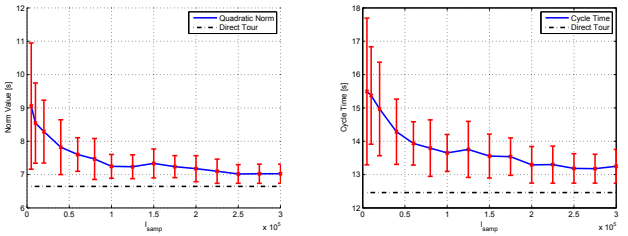


Fig. 3. Piecewise cubic least-square approximation for neighborhood $i = 14$. The actual configurations sampled at an interval of 5 deg are indicated with x-marks in the same color of the corresponding curve. The hyperspline consists of 8 polynomial pieces ($n_p = 8$) with breakpoints indicated by black x-marks.



(a) quadratic norm (b) Cycle Time

Fig. 4. Multiple query planner convergence as function of the parameter l_{samp}. Direct tour is the lower bound for the optimal value of the objective function.

inspect a predefined set of features on the component surface using the camera and a real time image processing software. To perform this task, the component is rotated and the manipulator is simultaneously actuated to locate the camera at all the relative placements with respect to the component surface where the required images have to be acquired from.

Since the sequence of the inspection locations is not fixed, the overall cycle time required to acquire all the images can be optimized by searching for an optimal sequence. Moreover, since the configuration space is seven-dimensional ($m = 7$), this system has one degree of redundancy. Figure 2 shows four among the infinitely many configurations that correspond to the same relative placement between the camera and the component, and thus to identical images. By exploiting this redundancy in the system, the cycle time can be further improved by searching not only for an optimal sequence, but also for an optimal sequence of optimal configurations.

A. Neighborhood definition

For each image i, i.e., for each relative placement between the camera and the component, the position and orientation of the manipulator end-effector can be represented in the turntable frame by a fixed homogenous transformation ${}^{(i)}\mathbf{T}_e^t$. If the rotation axis of the turntable is aligned with the z-axis of the manipulator base frame, if $\mathbf{p}_t^b$ is the origin of the turntable frame with respect to the manipulator base frame, if $q_{i,7}$ is the turntable rotation angle, and if $\mathrm{rot}(\boldsymbol{e}_z, q_{i,7})$ is the elementary rotation matrix about the z-axis then each neighborhood, i.e., each constraint (4) in the original formulation, can be defined as:

$$\mathrm{IK}\left(\begin{bmatrix}\mathrm{rot}(\boldsymbol{e}_z, q_{i,7}) & \mathbf{p}_t^b \\ 0\ \ 0\ \ 0 & 1\end{bmatrix} \cdot {}^{(i)}\mathbf{T}_e^t\right) = [q_{i,1}, \ldots, q_{i,6}]^T \ , \quad (11)$$

where $\mathrm{IK}(\cdot)$ is the inverse kinematic function of the manipulator. This definition does not depend on the edge weighting function used to define the objective function.

The inverse kinematic of a 6-DOF manipulator can have up to 8 different solutions, called *figures*. Therefore, constraint (11) represents a set of possibly non-connected curves in a seven-dimensional configuration space. However, since the focus of the present work is investigating how to exploit redundancy due to the additional DOF in the system, not due to the inverse kinematic of the manipulator, only one manipulator figure is considered. In particular, the left-handed shoulder, over-handed elbow, and normal wrist figure is used [14], leaving at most only one solution possible to the inverse kinematic for a given value of $q_{i,7}$. Using this figure all the required relative placements for the considered inspection problem can be reached. Moreover, by varying the value of $q_{i,7}$ each neighborhood is represented now by a unique curve in the seven-dimensional configuration space.

To preserve the continuous nature of the TSPN formulation while reducing the computational cost of the optimization procedure, piecewise cubic hypersplines are used to obtain a closed form definition of each neighborhoods. The inverse kinematic is solved for a set of values of $q_{i,7}$ with a 5 degree resolution, and the corresponding values of $q_{i,1}$ to $q_{i,6}$ are used to fit piecewise cubic splines, as illustrated in Figure 3. If $4 \times n \times n_p$ spline coefficients $\mathbf{s}_{k,i,p} \in \mathbb{R}^m$, where n_p is the number of polynomial pieces, and n spline parameter $t_i \in [0, 1]$ are employed for the parametrization, constraints (11) can be redefined as:

$$\sum_{p=1}^{n_p} \mathbf{1}_p(t_i)\big(\mathbf{s}_{0,i,p} + \mathbf{s}_{1,i,p}\,(t_i - t_{i,p}) + \mathbf{s}_{2,i,p}\,(t_i - t_{i,p})^2 + \mathbf{s}_{3,i,p}\,(t_i - t_{i,p})^3\big) - \boldsymbol{q}_i = 0 \qquad \forall\, i \in \mathrm{V}\ ,$$

where $t_{i,p}$ for $p = 1, \ldots, n_p + 1$ are the spline breakpoints, and $\mathbf{1}_p(t_i) = 1$ if $t_i \in [t_{i,p}, t_{i,p+1})$, 0 otherwise. For the considered experimental setup 8 polynomial pieces guarantee enough approximation accuracy while joint limitations, unfeasible configurations, and obstacle avoidance are used to define the limits of each hyperspline.

B. Planner Parameters

The parameter l_{samp}, i.e. the minimum number of configurations in the roadmap G, strongly influences the performance of the proposed path planner, and an ad-hoc method is

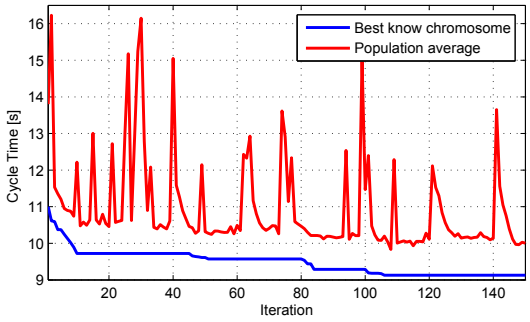


Fig. 5. Convergence rate of the prosed algorithm with high turntable speed and quadratic norm.

TABLE I

SIMULATION AND EXPERIMENTAL RESULTS FOR THE 7-DOF VISION INSPECTION SYSTEM.

Norm		Orig.	TSP	TSPN LS	TSPN HS
Weight. Max.	norm [s]	5.640	4.440	3.135	2.515
	impr.	-	21.3%	44.4%	55.4%
	sim. time [s]	13.86	11.80	10.14	9.40
	impr.	-	14.9%	26.8%	32.2%
	sys. time [s]	10.62	8.92	8.35	7.88
	impr.	-	16.0%	21.4%	25.7%
Quadr.	norm [s]	7.390	5.911	4.728	3.561
	impr.	-	20.0%	36.0%	51.8%
	sim. time [s]	13.86	11.74	9.98	9.13
	impr.	-	15.3%	28.0%	34.1%
	sys. time [s]	10.62	9.10	7.87	7.42
	impr.	-	14.3%	25.9%	30.1%

proposed to optimize its value. A roadmap is built repeating Algorithm 2 on randomly generated tours of random configurations until the threshold l_{samp} is reached. Afterwards, a predefined fixed tour of fixed configurations is analyzed using the multiple query planner. This procedure is repeated 10 times for each value of l_{samp} to provide information about average and standard deviation of the results. Figures 4.a and 4.b illustrate how quadratic norm and cycle time of the fixed tour evolve as function of l_{samp}. Based on these numerical tests the parameter is fixed to $l_{samp} = 100,000$ hereafter, which seems to guarantee a sufficient quality in the retrieved solution without resulting in excessive computational cost.

The other parameters used in the algorithm are: $\eta = 0.3$, $p_{loc} = 0.1$, $l_{loc} = 10$, $q_{res} = 0.08$, and $l_{merge} = 1,000$. Discussion on the influence of these parameters on the performance of single query planners can be found in the literature [8]. Finally, the indicator matrix, M_{conn}, is re-initialized every time the best known chromosome in the HRKGA does not improve for a certain number of consecutive iterations.

C. Tested scenario

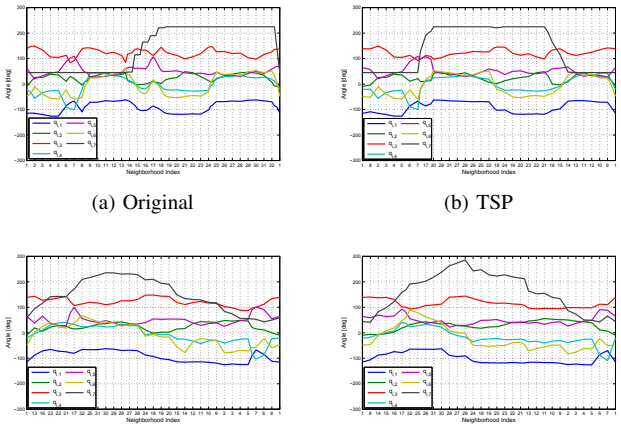
The proposed procedure is tested using a set of 31 relative placements ${}^{(i)}\mathbf{T}_e^t$ and one depot placement. $\boldsymbol{q}_{\text{MIN}}$, $\boldsymbol{q}_{\text{MAX}}$, ω_k, and α_k are obtained from the DENSO VS 6577E-B manipulator specification sheet [15]. For the turntable, i.e., joint 7, two sets of parameters are tested to simulate high and low rotational inertia scenarios: low and high angular speed, labeled "LS" and "HS", with values 2.88 rad/s and 6.98 rad/s, respectively. Collision evaluation is performed in a hierarchical fashion, first checking the bounding capsules containing the components of the system for intersection [16], and then using bounding-volume trees defined using triangular meshes [17], [18].

In the adapted HRKGA 60 chromosomes are used for each population, and the algorithm is terminated after 150 iterations. Figure 5 illustrates the convergence rate of the algorithm. The best known chromosome is usually improved after large immigration cycles, i.e., when the population average increases due to the introduction of new genetic material. On average, an improvement of the best known chromosome is observed in 51 iterations, 197,500 configurations are added to the roadmap, 18,100 calls are made to the Single Query Planner and 162,400 to the Multiple Query Planner.

Table I illustrates the results of four different solution procedures. The column with label "Orig." reports the results obtained with the original sequence, which was extracted by DENSO WAVE using a proprietary operator-guided procedure. These are used afterwards as reference values. The results reported in the column with label "TSP" are obtained by finding an optimal sequence of the original configurations. First, a distance matrix is calculated by using the planning procedure illustrated in the previous Section on all the possible edges of the full graph defined by the 32 original configurations. Then, the resulting TSP is solved to optimality using the exact TSP solver CONCORDE [19]. These results constitute an important benchmark for the proposed approach since they represent the best cycle time attainable with available optimization methods for this problem. Finally, the results reported in the columns with labels "TSPN LS" and "TSPN HS" are obtained using the proposed optimization procedure for low and high turntable speed, respectively.

In Sections III-A and III-B.2 it was illustrated that the weighted maximum norm or the quadratic norm is used to drastically reduce the computational cost of the algorithm internal operations, and Equation (9) is used to evaluate the cost of the chromosomes only at the end of each iteration. To investigate the influence of the selected norm on the overall performance of the algorithm, for each one of the four solution procedures and for each one of the two norms three different objective function values are reported: the norm-based objective function value, i.e., the cost of the extracted near optimal tour calculated using the two norms, which is labeled "norm", the corresponding cycle time calculated using Equation (9), which is labeled "sim. time", and the experimental cycle time, which is labeled "sys. time".

If the quadratic norm is used the actual cycle time can be reduced by 26% for low turntable speed or by 30% for high turntable speed. In this scenario the optimization achieved searching only for an optimal sequence of the original configurations is about 14%. Figure 6(a) shows the original tour, Figure 6(b) shows the tour of the original configurations



(a) Original (b) TSP

(c) TSPN LS (d) TSPN HS

Fig. 6. Optimization results obtained using the quadratic norm. The black line corresponds to the turntable joint angle.

optimized using the TSP solver, and Figures 6(c) and 6(d) show the near optimal tours obtained with the proposed procedure for the two turntable speeds.

If the weighted maximum norm is used the actual cycle time can be reduced by 21% for low turntable speed or by 26% for high turntable speed. In this case the optimization achieved using only the TSP solver is about 16%. These results confirm that both norms well represent the behavior of the actual system for the purpose of cycle time minimization, whereas the quadratic norm seems to lead to better optimization results.

Finally, we observe that the simulated cycle time is usually larger than the one measured on the actual system. The values calculated with the kinematic model defined in Equation (9) are larger than the actual ones on average by 20%, if the weighted maximum norm is used, and by 25%, if the quadratic norm is used. The actual dynamic performance achieved by the manipulator with a light camera mounted on its end-effector is probably higher than the one predicted using the standard kinematic parameters. However, the improvement trends of simulation and experimental results are consistent, i.e., high turntable speed always leads to shorter cycle time than the one obtained using low turntable speed or the TSP based optimization.

V. Conclusion and Future Work

In this work the Traveling Salesman Problem with Neighborhood (TSPN) is used to model the multi-goal path planning problem for redundant robotic systems. A Hybrid Random-Key Genetic Algorithm (HRKGA) is integrated with a probabilistic single/multiple query path planning technique based on bidirectional Rapidly-exploring Random Trees (RRTs) to better estimate the cost of each edge in the tour while generating collision-free paths. The proposed procedure is then tested on a 7-DOF robotic vision inspection system, where the neighborhoods are approximated using piecewise cubic splines in a seven-dimensional configuration space. Experimental results show that cycle time can be drastically reduced.

In future work we want to investigate the possibility not only to optimize the cycle time while exploiting the redundancy in the system, but also to minimize the total energy used while allowing a longer cycle time. Moreover, the proposed approach should be tested on robotic systems with higher degree of redundancy to understand how the algorithm scales in larger configuration spaces. Finally, it might be worth studying how a definition of the neighborhoods based on the forward kinematics could affect the computational cost and broaden the applicability of the proposed approach.

References

[1] E. Arkin and R. Hassin, "Approximation Algorithms for the Geometric Covering Salesman Problem," *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 197–218, 1994.

[2] M. Saha, T. Roughgarden, J. Latombe, and G. Sánchez-Ante, "Planning Tours of Robotic Arms Among Partitioned Goals," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 207–223, 2006.

[3] L. 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 *IEEE International Conference on Robotics and Automation, 2008 (ICRA 2008)*, 2008, pp. 2252–2258.

[4] G. Oriolo and M. Vendittelli, "A control-based approach to task-constrained motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, 2009, pp. 297–302.

[5] I. Gentilini, F. Margot, and K. Shimada, "The Traveling Salesman Problem with Neighborhoods: MINLP Solution," *Optimization Methods and Software*, vol. 28, no. 2, pp. 364–378, 2013.

[6] I. Gentilini, "Multi-Goal Path Optimization for Robotic Systems with Redundancy based on the Traveling Salesman Problem with Neighborhoods," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, 2012.

[7] W. Mennell, "Heuristics for Solving Three Routing Problems: Close-Enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, Sequence-Dependent Team Orienteering Problem," Ph.D. dissertation, University of Maryland, College Park, 2009.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] S. LaValle and J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[10] D. Berenson, S. Srinivasa, D. Ferguson, A. Collet, and J. Kuffner, "Manipulation planning with workspace goal regions," in *IEEE International Conference on Robotics and Automation, 2009 (ICRA 2009)*, 2009, pp. 618–624.

[11] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] E. Plaku, K. Bekris, B. Chen, A. Ladd, and L. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 597–608, 2005.

[13] J. Bentley, "Multidimensional divide-and-conquer," *Communications of the ACM*, vol. 23, no. 4, pp. 214–229, 1980.

[14] Denso Robotics, *Vertical articulated VS Series - Setting-up Manual*. Denso Wave, Inc., Japan, 2009.

[15] ——, *VS Series - Specification Sheet*. Denso Wave, Inc., Japan, 2009.

[16] M. Allen, G. Evans, D. Frenkel, and B. Mulder, "Hard convex body fluids," *Advances in chemical physics*, pp. 1–166, 1993.

[17] S. Gottschalk, M. Lin, and D. Manocha, "Obbtree: a hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 171–180.

[18] C. Ericson, *Real-time collision detection*. Morgan Kaufmann, 2005, vol. 1.

[19] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, "Concorde tsp solver," 2006. [Online]. Available: http://www.tsp.gatech.edu/concorde