Task-Based Motion Planning Using Optimal Redundancy for a Minimally Actuated Robotic Arm

Yanai Gal and David Zarrouk *

Department of Mechanical Engineering, Ben Gurion University of the Negev, Beer Seva 84105, Israel
* Correspondence: zadavid@bgu.ac.il

Abstract: In planning robotic manipulations, heuristic searches are commonly considered impractical due to the high dimensionality of the problem caused by redundancy in the kinematic chain. In this paper, we present an optimal motion planning algorithm for an overly redundant minimally actuated serial robot (MASR) using the manipulator workspace as a foundation for the heuristic search. By utilizing optimized numerical probability methods, a novel sub-workspace search was developed. The sub-workspace allows the search to quickly and accurately find the minimal subset of joints to be actuated and ensures the existence of a path to a given target. Further on, the search result is used as a search graph for the heuristic planning problem which guarantees an optimal solution within the problem boundaries. Using this approach, optimal heuristic search can become practical for various types of manipulators, tasks, and environments. We describe our workspace minimization and heuristic search using the example of a general robotic arm and then implement the approach on a MASR model, a robotic arm with five passive joints and a single mobile actuator that is free to travel along the arm and rotate each joint separately. A series of simulations show how our minimal redundancy approach can benefit from path planning in the case of traditional hyper-redundant manipulators, and its greater effectiveness when addressing the specific design of the MASR.

Keywords: serial robot; minimal actuation; workspace analysis; heuristic search; path planning; minimal redundancy

1. Introduction

Kinematic redundancy refers to the difference between the number of degrees of freedom (DOF) of a robotic manipulator and the DOF needed to execute a given task [1]. In confined spaces where traditional robots are generally unable to operate, the most common solution is to use hyper-redundant robots (with more than 6 DOF and 3 DOF for 3D and 2D working spaces, respectively). Their large number of DOF enables them to easily maneuver around obstacles, avoid singularities and joint limits, and minimize the energy consumption [2,3]. However, this comes at the cost of a notoriously complex design for control and motion planning due to the high dimensionality of the problem [4]. Analytical methods usually require an extremely complicated characterization of the motion planning problem [5], which is impractical to apply, whereas numerical approaches struggle with expensive computational efforts stemming from the complexity level of the planning problem [6]. Although planning optimization for hyper redundant robots has been extensively researched in the last two decades, there have been no major breakthroughs in the field of motion planning for modular serial robots. Modular robotic arms are composed of a varying number of modules (links, joints, etc.) assembled together to achieve a given purpose [7,8]. The ability to change the assembly of the modules provides control over the level of redundancy of the robot so it can be adapted to a specific environment or task [8]. Although most algorithms have been developed to optimize the motion planning...
problem of hyper-redundant manipulators, the optimization of the redundancy itself remains an open challenge. Minimizing a manipulator’s redundancy level to fit a task enables easier and more efficient motion planning. Moreover, as the redundancy level increases, the motion plan is subjected to larger mechanical uncertainties, due to the accumulating backlash and the mechanical freedom associated with the joints. Thus, “minimal redundancy” clearly has both mechanical and computational advantages. We define minimal redundancy as the minimal number of joints that need to be actuated to perform a certain task. More specifically, when given a manipulation task such as pick-and-place, and the robot’s mechanical properties are known beforehand (such as joint limits, module dimensions, grabbing tolerance, etc.), the goal is to find the minimal subset of joints that ensures the existence of a path to the target. Then, based on this particular subset, we deploy a heuristic search algorithm to find the path itself.

In this paper, we use a planning algorithm on a planar modular manipulator to find a least-cost trajectory with mechanical constraints and obstacle avoidance, while optimizing the number and order of the modules needed for robot assembly. Although the algorithm can be easily applied to various manipulators consisting of \( N \) revolute joints (NR robots), as shown below, it was applied here on the recently developed minimally actuated serial robot (MASR) [9,10], a reconfigurable robotic arm with passive (non-motorized) rotary joints, and a single mobile actuator that travels along the arm to rotate the passive joints. In addition to the modularity of the MASR, it differs from other NR manipulators in two respects, both derived from its unique design. Since the passive joints are externally actuated by a single actuator, only one joint at a time can be rotated. However, the mobility of the actuator allows the gripper, which is not fixed to the end of the manipulator, to travel to any point along the arm. This design makes it possible to manufacture a low-cost light-weight robotic arm that can be adapted for a variety of purposes, such as agricultural automation, surgical operations, or space tasks.

This paper is organized as follows. Section 2 presents the key mechanical characteristics of the MASR that the algorithm uses as a constraint to achieve an optimal solution. In Section 3, we present a novel approach for the fast and accurate numerical computation of the manipulator workspace, which we use later on to find the optimal module assembly. Once the assembly is complete, we find a least-cost trajectory in Section 4, using the combination of a heuristic search and our workspace analysis. In Section 5, we test the algorithm in several simulations to assess its performance.

2. MASR Mechanical Characteristics

The solution to our motion planning problem depends, as is the case for many other problems, on boundary conditions, which in a manipulation task are composed of the geometric and mechanical constraints of the robotic arm. Thus, before elaborating on the motion plan itself, the key mechanical characteristics of the MASR are outlined. Figure 1 presents the design of the MASR robot fitted with five passive joints (the number of joints can be increased or decreased based on the task). For more a comprehensive description of its design and operation, as can be seen in [9,10].

In this example, the manipulator is composed of a 1 m-long planar arm comprised of two to five links (20, 40, 60, or 80 cm long, 4 cm wide, 5 cm high). The links are connected by passive revolute joints (2R–5R robot) whereas a single mobile actuator travels over the links to reach designated joints to rotate them (Figure 1). After being rotated by the mobile actuator, the joints remain locked, using a worm gear setup. A gripper is attached to the mobile actuator, thus enabling it to transport an object along the links to decrease the actuation of the joints and the work time. The relative angle \( \theta \) between two adjacent links (at 1 and \( j \)) can be varied in the range of \([-60°, 60°]\). In the following sections, we describe the algorithm and present simulations based on the mechanical and geometric properties of the MASR. Note that although the algorithm solves the problem of a minimal actuated modular arm, it can also be used in simpler cases, such as a fully actuated manipulator (all-motorized joints), and at any level of redundancy, with or without modularity, as shown below.
Figure 1. Mechanical design of the MASR robot. In this example, the robot consists of a planar serial arm with five passive joints actuated by a mobile actuator. The mobile actuator, which can travel over the links, is fitted with a gripper to carry objects along its path. The vertical motion is actuated by a linear lift unit, based on a stepper motor with a lead screw [9].

3. Workspace Analysis

The primary workspace of a robotic arm is defined as the set of points that can be reached by its end-effector [11] in a given environment when all of its joints can be actuated. Activating only a subset of joints results in a smaller or equal sub-workspace than in the primary workspace. Therefore, if a target lies within a sub-workspace’s boundaries, the existence of a path to reach it, using the appropriate subset of joints, is guaranteed. This premise constitutes the guiding principle for the minimal redundancy motion plan algorithm: instead of applying a direct and exhaustive search to find the least-redundant path (i.e., the path that requires actuating the minimal number of joints), we divide the problem into two steps.

First, we search through the sub-workspaces to find the arm’s minimal redundancy, i.e., the minimal subset of joints that simply ensures the existence of a path to the target. Then, based on the minimal subset of joints, we apply a heuristic search algorithm to find the path itself (Section 4).

In this section, we introduce an enhanced approach to determine the manipulator’s workspace that uses a superposition of sub-regions based on a beta distribution function. This approach makes it possible to depict the manipulator workspace boundaries quickly and accurately. This not only serves to determine the manipulator’s minimal redundancy, but also to use it as a search graph, as will be shown in the next section.

In this example, the manipulator is composed of a 1 m-long planar arm comprised of two to five links (20, 40, 60, or 80 cm long, 4 cm wide, 5 cm high, see Figure 2).
Figure 2. Number of assemblies to form a one-meter arm using different combinations of links with lengths that vary from 20 cm to 80 cm. In the case of a pick-and-place task, the optimal combination will use the minimal number of links.

3.1. Workspace Computation

The exact and rapid computation of the shape and size of the robot manipulator workspace is crucial for finding its minimal redundancy. When analyzing hyper-redundant robots with joint constraints, the analytic computation of the workspace becomes impractical [12] and numerical approaches based on the Monte Carlo random sampling method become more effective [13]. Although it is simple to apply since it does not involve inverse Jacobian calculations or complex analytical representation, the Monte Carlo yields non-uniform solutions which may omit parts of the workspace.

For planar manipulators consisting of \( N \) revolute joints (NR robot), the \((x_j,y_j)\) position of joint \( j \in \{1, \ldots, N\} \) is given by:

\[
X_j = \begin{bmatrix} x_j \\ y_j \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{j-1} L_n \cos \left( \sum_{m=1}^{j} \theta_m \right) \\ \sum_{n=1}^{j-1} L_n \sin \left( \sum_{m=1}^{j} \theta_m \right) \end{bmatrix}
\]  

(1)

We define the base link as \( j = 1 \), where \((x_1,y_1) = (0,0)\). \( L_i \) is the length of link \( j \), which appends to joint \( j \), and where \( \theta_j \) is the relative angle between link \( j \) and \( j - 1 \). Note that for \( j = N + 1 \), we obtain the position of the end-effector \( p \) with respect to the world frame.

Rastegar et al. used the Monte Carlo method [14] while relying on uniformly distributed random values for every joint angle \( \theta_j \). For every set of joint angles, the end-effector position is calculated using Equation (1), thus forming a cloud of “Monte Carlo points” which represents the workspace plane. Although the angle values are generated uniformly, the non-linearity in Equation (1) leads to a non-uniform distribution of the cloud points within the workspace, as shown in Figure 3a, where most of the points are within the upper outer part of the cloud.

To improve the accuracy of the workspace boundaries, and reduce the number of points needed to fully describe the workspace, ref. [11] suggested generating random values using the beta density function:

\[
\text{Beta}(\alpha, \beta): \quad \text{prob}(x \mid \alpha, \beta) = \frac{x^{\alpha-1} (1-x)^{\beta-1}}{B(\alpha, \beta)}
\]  

(2)

where \( B \) is the beta function:
The major advantage of the beta distribution is the various shapes it can take, depending on the values of the parameters α and β. According to the simulations conducted in [11], in a 3R model, the optimal value for (α, β) ∈ (0, 1) is solely determined by the motion range of the joints, regardless of the number of joints. For example, each joint of the MASR has an equal motion range of 120°, so that according to [11], α and β are determined by:

\[ \alpha = \beta = \frac{\bar{\theta}}{5\pi} + 0.3, \quad \frac{\pi}{2} \leq \bar{\theta} \leq \pi \]  

where \( \bar{\theta} \) stands for the motion range in radians. Substituting \( \bar{\theta} = \frac{2\pi}{3} \) radians in Equation (4) yields the optimal parameter value of \( \alpha = \beta = 0.43 \). However, while still the most effective as a single parameter value, in the simulations, we found that for larger numbers of joints (\( N > 3 \)), this number (0.43) loses its effectiveness and a larger number of simulated points \( N_p \) is required. A comparison of the examples shown in Figure 3a,b, which were both based on the 5R MASR model, makes it clear that for the same number of points (20,000), the cloud generated with the beta distribution (b) has a much more homogenous spread, but the upper-outer boundary has been left out and the bottom and inner boundaries are vague. To be able to more accurately describe the workspace boundary, for various numbers of links, without having to increase the number of generated cloud points, a more generalized approach must be considered.

By analyzing the alteration of the workspace’s cloud as a function of change in the \( (\alpha, \beta) \) parameters, we found that for all NR models for the MASR (i.e., where the highest was \( N = 5 \)), low parameter values \( (\alpha, \beta \to 0) \) accurately depicted the bottom and inner boundaries of the cloud, and high values \( (\alpha, \beta \to 1) \) described the upper-outer boundary. This shows that no single parameter value can be used to accurately find all of the workspace boundaries. Instead, we suggest using two pairs of parameters where the work volume is the union of each cloud:

\[
U_{\text{low}} = \text{Beta}(\alpha_1, \beta_1, \frac{1}{2} N_p) \cup \text{Beta}(\alpha_2, \beta_2, \frac{1}{2} N_p)
\]  

where \( U_{\text{low}} \) is the union beta function, \((\alpha_1, \beta_1)\) and \((\alpha_2, \beta_2)\) are, respectively, the low and high parameter values, and \( N_p \) is the total number of cloud points generated, i.e., each sub-region is created with half of the total number of cloud points. For example, instead of creating a uni-parameter value cloud, as shown in Figure 3b, half of the points (10,000) can be generated with \( \alpha_1 = \beta_1 = 0.1 \) and the other half with \( \alpha_2 = \beta_2 = 0.9 \). The union of the low value region (blue in Figure 3c) with the high value region (green in Figure 3c) creates a cloud with much clearer boundaries, without changing the total number of generated points. The accuracy of the boundaries in this new approach is illustrated in Figure 3c and when increasing \( N_p \) dramatically to one million points (Figure 3d). The comparison shows that although the inner cloud was completely covered with cloud points, the shape and size of the boundaries remain practically unchanged.
Figure 3. Workspace computation of the 5R manipulator, with a uniform link length of 200 mm and a relative joint angle range $\theta_j \in [-60^\circ, 60^\circ]$, using 20,000 Monte Carlo points with uniform and beta distribution functions. (a) Generating a workspace cloud with uniformly distributed points (green); (b) Using a beta distribution with $(\alpha, \beta) = 0.43$; (c) and (d) combining the two regions produced with $(\alpha, \beta) = 0.1$ (blue) and $(\alpha, \beta) = 0.9$ (green) for 20,000 and 1,000,000 points, respectively.

3.2. Contour Coverage

As the number of joints comprising the arm increases, the workspace area grows as well, and more cloud points are needed for an accurate boundary description. Since more cloud points result in more extensive computational effort, the goal was to find the minimal number of points per number of joints that can form clear boundaries. The optimal superposition parameters ($((\alpha_1, \beta_1)$ and $((\alpha_2, \beta_2)$ in Equation (4)) provide full contour coverage, i.e., the area enclosed by the workspace boundary, with the minimal number of cloud points $N_p$. To compute the contour coverage, we used MATLAB’s boundary function, which creates a polygon shape around the cloud, and then calculated the area within it using the polyarea function. The determination of the optimal parameters was performed by examining different high-medium-low combinations of values within the range $(\alpha, \beta) \in (0,1)$ to find the best $(\alpha, \beta)$ pairing, while comparing the simulation results from the sub-regions method to those obtained when using a single optimal beta value (i.e., $\alpha=\beta=0.43$) and a uniform distribution (i.e., $\alpha=\beta=1$). The principle guiding the simulations involved computing the contour coverage for different combinations of values of the parameters, with increasing numbers of cloud points.Statistically, for a sufficiently large number of points, the contour coverage of all combinations should converge to the same constant value. Since this number could be infinitely large, we took $N_p = 1,000,000$ to be the upper bound, whereas the maximal area calculated across all combinations was taken as the true contour coverage value. To determine which values converged the fastest, we set a criterion of 99% of the full contour coverage, i.e., the minimal number of cloud points needed to reach 99% of the true coverage value.

The simulation results presented in Figure 4 show that the low–high combination ($(\alpha, \beta) = 0.1\cup0.9$) achieved full contour coverage the fastest, since it was the first (210,000 cloud points) to enter and remain within the 99% zone (grey in Figure 4). Compared to
previous methods, the single optimal beta value (0.43) ranked third for 99% coverage using 610,000 cloud points, compared to the low–medium combination ($\alpha \beta = 0.1 \cup 0.5$) at 260,000 points, but behind the medium–high ($\alpha \beta = 0.5 \cup 0.9$) combination at 1,000,000 points, whereas the uniform distribution did not meet the criteria at all.

![Contour coverage simulation results](image)

**Figure 4.** Contour coverage simulation results. Top—different sets of ($\alpha \beta$) were combined with low–medium–high values of $0.1–0.5–0.9$ and compared to single beta values of $\alpha = \beta = 0.43$ and to $\alpha = \beta = 1$ (uniform distribution). Bottom—pairs of ($\alpha \beta$) values between 0.1 and 0.9 combined with a constant pair with value of 0.9. All simulations were based on the 5R MASR model.

These simulations tested the minimal number of cloud points needed to achieve full coverage in case of a 1 m arm, composed of five links. In subcases where the arm has fewer links (or fewer joints that are active), the workspace area will naturally decrease, and a smaller number of cloud points will be needed. To set a criterion for the number of points per number of links, we ran similar convergence simulations for a 1 m arm composed of two, three, and four links of equal length. The results confirm that the low–high combination ($0.1$ and $0.9$) produced the fastest convergence, and in addition (as indicated in Table 1), the superposition method dramatically decreases (up to 90.1%) the number of points needed to reach full workspace coverage—compared to the optimal single value of ($\alpha \beta$) = 0.43.
Table 1. Comparison of superposition and no superposition for the minimal number of points needed to achieve 99% coverage.

<table>
<thead>
<tr>
<th>Number of Links</th>
<th>Points to Full Contour Coverage</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Superposition $(\alpha, \beta) = 0.1\times0.9$</td>
<td>No Superposition $(\alpha, \beta) = 0.43$</td>
</tr>
<tr>
<td>2</td>
<td>1600</td>
<td>4950</td>
</tr>
<tr>
<td>3</td>
<td>11,100</td>
<td>112,350</td>
</tr>
<tr>
<td>4</td>
<td>56,950</td>
<td>267,200</td>
</tr>
<tr>
<td>5</td>
<td>210,000</td>
<td>610,000</td>
</tr>
</tbody>
</table>

3.3. Optimal Link-Joint Assembly

Our aim here was to determine which link-joint combination best executed a given pick-and-place task with minimal redundancy, i.e., with the smallest number of joints possible. In other words, the goal was to find the smallest workspace area that contains the set of targets to reach. To do so, we created an arm with the largest number of possible joints, and computed the workspace boundaries for each subset of joints in ascending order until all the targets were within the boundary. Initially, a preliminary search was performed with the maximal joint-combination (i.e., five active joints in this case) to verify whether the targets were within the arm’s reach or not. If they could be reached, an iterative breadth-first search was launched. Initially, all two-joint combinations were examined; then, if no related workspace enclosed the targets, all three-joint combinations were examined and so on until a five-joint-combination was found. To determine whether a target was in the workspace, we used MATLAB’s inpolygon function. The maximal number of workspace computations $C_{max}$ for an arm with $N$ possible joints is given by:

$$C_{max} = \sum_{i=1}^{N} \binom{N}{k} \binom{N}{k}$$

For example, take the case in which we were given two targets to reach with the following $(x,y)$ coordinates: $(500,800)$ and $(-700,-300)$. We tested the MASR arm with its maximal five-joint assembly configuration. $C_{max}$ is 26 computing iterations, but it only took 11 to find the minimum combination (Figure 5f). Since each iteration consists of creating a cloud, calculating the boundary, and determining whether the targets were in it, it is highly important to compute the workspace with the minimal number of points (Table 1) to save on computational efforts. Using the superposition cloud points (Table 1) in this case was 44.3% faster than without the superposition (0.176 s and 0.316 s, respectively). As the maximal number of joints increases (i.e., as the redundancy increases), more iterations are needed for the subset scans, so that the superposition method becomes increasingly effective. Once the minimal subset of joints has been found, the modular arm is rebuilt with the associated links to create the minimally redundant manipulator for the task. In the current example, the fourth and fifth joints were the redundant ones, so that the optimal assembly (or minimal redundancy) was an arm consisting of three links with lengths of [20,20,60] cm. In the case of a non-modular arm, rebuilding would not be possible, and the motion plan would only consider the first three joints as the active ones.
Figure 5. (a–d): Last link (LL) workspaces scanned by joint combinations in ascending order. The numbers in the bottom right corners indicate the number of iterations. The highlighted joints were used to compute the workspace. To reach the targets, only the first, second, and third joints are needed; (e–f): any link (AL) workspaces, which serve to create a workspace using a single joint (e) and cover a larger area (f) for the same number of joints as used in the LL method (d).

3.4. Last Link and Any Link Workspace

The workspace computation typically involves the targets that can be reached by the end of the last link (LL) of the arm, since this is where the end-effector is located. However, the MASR differs in that it can move the end-effector along the arm because it is attached to the mobile actuator. This means that the workspace can be calculated as the set of points that can be reached by any link of the arm (AL). The area of the AL workspace is obviously larger (or at least equal) to that of the LL workspace when given the same set of joints. In other words, the AL workspace covers a larger area with fewer joints. Therefore, the use of the AL workspace makes it possible to find a less redundant solution. Nevertheless, when computing the AL workspace, the super-position beta method remains the same with the exception of one modification that involves creating link-lines instead of points for the last link. In the example in Section 2, the LL workspace method produced a minimal assembly consisting of 3 out of 5 possible joints, in 11 iterations. Using the AL workspace for the same set of targets produced the same results (Figure 5f bottom). However, when considering a simpler case such as reaching the top right target, the AL method resulted in 1 out of 5 joints (Figure 5e), compared to 2 out of 5 in the LL method (Figure 5a,b).
The results shown in Figure 5 show that the AL method not only covered a larger area compared to the LL, but also created a workspace using a single joint (whereas the LL creates a one-dimensional line). Since the AL workspace contains its associated LL workspace, the minimal assembly found by the AL method will always be less or equally redundant compared to the LL method.

4. Motion Planning

Once a minimal arm assembly was found, a motion plan is needed to execute the pick-and-place task. In this section, we present a new approach for finding the optimal path by implementing the popular heuristic search A* [15], using the cloud points of the workspace as the search graph. Although heuristic searches have often been applied to find least-cost plans because of their generality, robust guarantees on completeness and optimality, and simplicity in implementation [16,17], in path planning for robotic arms, these methods are commonly considered impractical for high-dimensional problems [18-20]. By achieving minimal redundancy and representing the search problem as the cloud points graph, the dimensionality is kept at a minimum. In addition, the path search by graph serves to avoid inverse kinematic calculations, which become complicated in high-dimensional problems.

4.1. Areal Coverage

An accurate graph-based search is only guaranteed if the graph represents the entire feasible workspace. While full contour coverage is sufficient for finding the optimal arm assembly, it mostly describes the workspace’s boundaries, and not the surface. Therefore, creating a workspace-based graph requires the formation of a cloud with full areal coverage.

We define $r$ as the grabbing tolerance, i.e., that the end-effector has reached a specific point if the distance between the two is $r$ or less. In other words, each cloud point has an area cover radius of $r$. Similar to our full contour coverage definition, we define full areal coverage as the minimal number of cloud points with a cover radius $r$ needed to reach 95% or 98% (respectively, for low- and high-accuracy tasks) of the true workspace area; i.e., the true contour coverage value (Section 3.2). To find a criterion for the number of points per link, a series of simulations were conducted for several cover radii. In each simulation, an increasing number of cloud points were generated by the superposition method presented in the previous section. In each cloud, the points were connected to form triangles, using MATLAB’s AlphaShape (based on the alpha shape method [21] and the cover radius $r$). The total area of the alpha shapes was calculated with MATLAB’s area. An example of these simulation results is shown in Figure 6. The number of cloud points needed for full areal coverage increases as the cover radius decreases and/or the cover percentage increases. Comparing the results for the contour coverage (Table 1) and the areal coverage (Table 2) simulations shows that the latter required up to 51.25 times more points to achieve 95% coverage with $r = 5$ mm, and up to 67.5 times more for 98% coverage with $r = 5$ mm. These results underscore the efficiency of these new contour and areal coverage approaches: the contour coverage required a relatively small number of points to accurately describe the workspace boundaries, thus making it well-suited for running a quick and efficient iterative search along the sub-workspaces of the arm, but less suitable as a search graph.
Table 2. Criteria for the number of cloud points to generate to achieve 95% and 98% area coverage for a cover radius of 5, 10, and 15 mm.

<table>
<thead>
<tr>
<th>Number of Links</th>
<th>98% Coverage</th>
<th>95% Coverage</th>
<th>Number of Links</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r = 15 mm</td>
<td>r = 10 mm</td>
<td>r = 15 mm</td>
</tr>
<tr>
<td>12K</td>
<td>27K</td>
<td>108K</td>
<td>20K</td>
</tr>
<tr>
<td>67K</td>
<td>148K</td>
<td>574K</td>
<td>49K</td>
</tr>
<tr>
<td>181K</td>
<td>384K</td>
<td>1406K</td>
<td>117K</td>
</tr>
<tr>
<td>282K</td>
<td>587K</td>
<td>2061K</td>
<td>165K</td>
</tr>
</tbody>
</table>

Figure 6. Areal coverage simulation example results for 95% coverage with two cover radii. The top figures present areal coverages of the 5R MASR workspace, produced from $10^5$ cloud points, with cover radii of 10 mm (left) and 5 mm (right). The bottom figures show differences in the convergence of the coverage percentage as the number of active joints decrease and the cover radius increases.

In contrast, the areal coverage requires a significantly larger number of points. Thus, although it is impractical to use in an iterative search, it is well suited to use as a search graph. The minimal redundancy motion plan algorithm exploits the advantages of both the contour and the areal coverages. It uses the former to consecutively calculate sub-workspaces until minimal redundancy has been achieved (Section 3). The latter is then used exactly once on the chosen subset of joints to create an informative search graph for the heuristic search. Table 2 shows how the number of cloud points for full coverage rises as the cover radius decreases and the number of joints increases. The size of the cover radius and percentage is derived from the end-effector properties, type of task, the environment, etc. For high accuracy tasks, lower radius value and higher cover percentages will be selected, unlike simpler tasks, where larger radius values and smaller percentages typically suffice. When using a graph search, the best strategy is to search for the sparsest graph possible that still accurately describes the environment.
Since the search graph is based on the cloud points, it is crucial for the efficiency of the search to generate the minimum number of cloud points. By adjusting the cover radius and percentage to the task, we can maintain the number of cloud points at a minimum, without losing significant data for the motion plan.

4.2. Heuristic Search

Any effective heuristic search relies on two major components: the search graph and the cost-to-go estimation. The level of accuracy at which these two components describe a specific problem enables obtaining a more efficient solution [18,22]. Despite the popularity of heuristic searches in robotic navigation, devising a search graph for manipulators is considered impractical due to the high-dimensionality of the state-space problem [18]. Our new approach for manipulator path planning relies on using the cloud points with full areal coverage and suitable heuristics as the search graph. The cloud contains all the arm’s kinematics (configurations and constraints), as well as the data on unachievable configurations as a result of the presence of obstacles within the workspace. By searching the cloud points, we can find an optimal path without needing inverse kinematics, Jacobian calculations, or obstacle sensing. Since we are generating a minimal number of points, we also guarantee the minimal possible dimensionality of the problem.

To perform the search, we need to know the start and goal nodes for the graph. In the case of manipulators, the nodes can be defined by the end-effector position or by the joint configuration. Since all configurations are known from the cloud, and our aim is to find the path with the fewest joint rotations, we define the goal node as the configuration which is both within the cover radius and has the least angular change with respect to the starting configuration. We then apply the traditional A* search, with a modification of a neighbor search: instead of searching for the eight Cartesian neighbors, we identify all the cloud points within a search radius which is predefined from the geometry of the cloud. The optimal path is defined as the path with minimal angular change between the start and goal configurations; hence, the cost at each node \( s \) is given by:

\[
\begin{align*}
    f(s) &= g(s) + h(s) \\
    g(s) &= \sum_{i=\text{start}+1}^{s} \| \bar{\theta}(s_i) - \bar{\theta}(s_{i-1}) \| \\
    h(s) &= \| \bar{\theta}(s) - \bar{\theta}(s_{\text{goal}}) \|
\end{align*}
\]

where \( f(s) \) is the total estimated cost from start to goal, \( g(s) \) is the true cost from the start node to the current node \( s_n \), calculated as the 2nd norm sum of change in the relative joint angles \( \bar{\theta} \), between adjacent nodes \( s \) and \( s_{i-1} \). \( h(s) \) is an optimal estimation of the cost to go, calculated as the norm of angular change between the current node and the goal. To guarantee optimality in the solution, the heuristic function \( h \) was constructed so it is admissible and consistent, i.e., it meets the requirements: \( h(s_{\text{goal}}) = 0 \) and \( h(s_i) \leq h(s_{i-1}) + c(s_i, s_{i-1}) \), where \( c(s_i, s_{i-1}) \) is the true cost between the adjacent nodes. Another advantage to the minimization of angles is the ability to use the same heuristic function and the A* search in general, both on last link and any link searches. The optimal path found by this search is based on the assumption of a fully actuated arm; in other words, that all the joints are rotating simultaneously. In the case of an MASR with only one actuator, the solution is split into its components in a depth-first approach, thus obtaining a minimally actuated path for the minimally redundant arm.
5. Simulation Results

This section presents two simulations for optimal motion planning based on applying the optimal assembly with the heuristic search. In each simulation, the same robotic model was tested (the MASR model), in the same environment. The first simulation demonstrates a full run of the algorithm over a classic pick and place task, while emphasizing the advantages of finding minimal redundancy before planning the path. The second simulation tests the entire algorithm on random targets, to show its efficiency in general, and for the AL approach in particular. All simulations were carried out on an 11th Gen Intel Core i7-1165G7 (2.80 GHz) with 15.7 GB of RAM running on MATLAB 2019b on a Windows 10 OS.

5.1. Pick and Place

In this simulation, the entire algorithm was tested over a typical pick and place task while maneuvering around an obstacle. The positions of the targets are given by their \((x,y)\) coordinates in mm, where \((500,800)\) is the pick, and \((-700,-300)\) is the place, when \((0,0)\) represents the first (and fixed) joint of the arm. The obstacle was a vertical wall between the points \((200,700)\) and \((200,1100)\), and the arm’s initial position corresponded to the case where all links are vertically aligned. First, an optimal assembly was generated after 21 workspace iterations (0.47 s), which considered joints 1–4 as active (magenta in Figure 7a) and left the last joint redundant. Secondly, a search graph was created with 935K nodes (Table 2) to achieve 95% full areal coverage with a 5 mm cover radius, when all configurations less than 50 mm from the obstacle were eliminated (safety offset). The search from the initial position to the target to pick (Figure 7b) required 127,267 node expansions, and the second search from the pick to place targets took 324,460 expansions. The number of expansions is a conventional criterion in heuristic searches, since it solely depends on the algorithmics, as opposed to runtime, which for the same search, can differ across different programs and environments. Thus, performance was tested on the number of expansions. For comparison, other heuristic searches that were executed on similar models such as in [18] without the minimal workspace as the search graph required expansions in the scale of millions, and often ran out of memory. This indicates that optimal path planning by using the minimal redundancy approach is at least ten times more effective than previous heuristic methods. To keep the search as efficient as possible, the path found by the A* search was adapted for a fully actuated manipulator. To do so, the last step in the algorithm was to separate the simultaneous movement into its component (Figure 7c, steps downsized by 20) to achieve a minimal redundant motion plan for a minimally actuated arm.

![Figure 7. Simulation results for the pick and place task. (a) For the given targets, minimal redundancy only involves 4 out of 5 joints (in magenta). In (b) and (d), an optimal path was found for the target to pick (b) and place (d), using full areal coverage as the search graph for the A* search. The steps shown in black were downsized by half. (c) Shows the path from pick to place after modification for a minimally actuated arm.](image-url)
5.2. Last Link vs. Any Link

One of the main advantages of the MASR over other manipulators is its ability to move the end-effector along the arm, i.e., any link (AL) and not just by the last link (LL) can grab an object. In the previous simulation, we showed that in the LL case, which is applicable to any other traditional manipulator, our algorithm exhibited a significant improvement in terms of computational efforts and the feasibility of the solution. The next simulation was conducted to test the AL advantage in terms of numerical performance. The arm’s model, initial position, and environment were the same as in the previous simulation. Twenty targets to pick were randomly generated in the arm’s vicinity, and the algorithm was executed twice for each target, once as LL, and once as AL. Naturally, for the same set of joints, the AL workspace is larger than the LL, so we expected that for a given target, the AL approach would require fewer active joints than the LL. Fewer active joints means fewer configurations for full areal coverage, which translates into a smaller search graph, and hence a more effective path planning. Out of the 20 targets, the algorithm found that 10 were out of reach for both AL and LL (marked in red, Figure 8). This emphasizes another advantage of the preliminary search for minimal redundancy; namely, that it can predetermine whether a path does not exist, and thus economize on a significant amount of unnecessary computing efforts. As for the target within the workspace boundary (marked in green, Figure 8), once the optimal assembly was found for LL and AL separately, the A* search was executed for each.

![Figure 8. Last link vs. any link simulation. The algorithm indicated that the targets in red were out of reach. The targets in green were within the workspace boundary and were carried on to the path planning stage.](image)

The results presented in Table 3 show that in 8 out of 10 searches, the AL approach was more efficient (up to 12.2 times improvement) than LL. The two targets for which the LL outperformed the AL indicate that in cases where the target is near the tip of the arm (target No. 16, Figure 8), the LL did better, although the difference was negligible in terms of the number of expansions (4 as compared to 9 expansions, Table 3). The case where the target lies near the outer boundary of the maximal joint assembled arm (5 active joints in our case, target No. 7, Figure 8) is also instructive. If both AL and LL use the same joints to create a fully covered workspace, the number of nodes in the search graph will be the same, which directly leads to a similar number of expansions. Thus, these results make it clear that while the minimal redundancy through the LL approach can make optimal path planning possible, the AL approach can make it practical.
Table 3. Expansion within the A* search for last link and any link. The target list corresponds to the numbering in Figure 8.

<table>
<thead>
<tr>
<th>Target No.</th>
<th>Last Link Expansions</th>
<th>Any Link Expansions</th>
<th>LL/AL Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>152,332</td>
<td>27,738</td>
<td>5.5</td>
</tr>
<tr>
<td>7</td>
<td>10,448</td>
<td>11,044</td>
<td>0.9</td>
</tr>
<tr>
<td>9</td>
<td>34,548</td>
<td>11,044</td>
<td>3.1</td>
</tr>
<tr>
<td>10</td>
<td>74,128</td>
<td>10,530</td>
<td>7.0</td>
</tr>
<tr>
<td>12</td>
<td>4495</td>
<td>447</td>
<td>10.1</td>
</tr>
<tr>
<td>13</td>
<td>1211</td>
<td>99</td>
<td>12.2</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>9</td>
<td>0.4</td>
</tr>
<tr>
<td>17</td>
<td>1851</td>
<td>276</td>
<td>6.7</td>
</tr>
<tr>
<td>18</td>
<td>13,021</td>
<td>12,712</td>
<td>1.0</td>
</tr>
<tr>
<td>20</td>
<td>32,299</td>
<td>9693</td>
<td>3.3</td>
</tr>
</tbody>
</table>

6. Conclusions

In this paper, we presented a novel approach to motion planning for a minimally actuated serial robot (MASR) that minimizes not only the trajectory cost, but also the number of joints that need to be actuated. Using the Monte Carlo method, we integrate the beta distribution with the non-linear kinematics of the manipulator to achieve new numerical workspace analysis method, which makes it possible to find the minimal subset of joints that need to be activated to reach a given target, before finding the path itself. The preliminary search for the minimal workspace is also proven to be beneficial in creating a heuristic search graph, establishing start and goal nodes, and forming a consistent cost function. This method emerged as highly effective when planning a path for hyper redundant robots such as the MASR. The ability to quickly and accurately describe a hyper redundant manipulator’s workspace boundary (up to 44.3% faster than previous methods) makes it possible to run a preliminary search to find the minimal subset of joints to be actuated (defined as “minimal redundancy”) to reach a given target. The use of minimal redundancy guarantees that the dimensionality of the problem is kept at a minimum. Together with taking our numerical workspace as a search graph, path planning with an optimal heuristic search such as A* becomes practical and efficient (reduces expansions number by a factor of 10). We showed that, in cases when the arm’s end-effector is not fixed to the last link, and can move along the arm, the workspace volume grows substantially for the same set of joints, which translates into even more efficient planning (by reducing expansions up to a factor of 12.2). We believe that our minimal redundancy approach will enable heuristic searches for robotic manipulation to become more popular, and less impractical.

Our future work will focus on expanding the workspace analysis to 3D space, while taking additional types of prismatic or spherical joints into consideration. We also plan to incorporate machine learning tools into our motion planning algorithm to improve the effectiveness of the search.

Author Contributions: Conceptualization, Y.G. and D.Z.; methodology, Y.G. and D.Z.; software, Y.G.; validation, Y.G.; investigation, Y.G.; writing—original draft preparation, Y.G.; writing—review and editing, Y.G. and D.Z.; supervision, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the Ministry of Science and Technology (MOST), grant number is 315654, the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Initiative, and by the Marcus Endowment Fund at Ben-Gurion University of the Negev.

Conflicts of Interest: The authors declare no conflict of interest.
References