Letter

Finding Optimal Manipulator Arm Shapes to Avoid Collisions in a Static Environment

Tomáš Kot\textsuperscript{1,*}, Zdenko Bobovský\textsuperscript{1}, Mathias Brandstötter\textsuperscript{2}, Václav Krys\textsuperscript{1}, Ivan Virgala\textsuperscript{3} and Petr Novák\textsuperscript{1}

\textsuperscript{1} Department of Robotics, Faculty of Mechanical Engineering, VSB-Technical University of Ostrava, 17. listopadu 2172/15, 708 00 Ostrava-Poruba, Czech Republic; zdenko.bobovsky@vsb.cz (Z.B.); vaclav.krys@vsb.cz (V.K.); petr.novak@vsb.cz (P.N.)
\textsuperscript{2} Institute for Robotics and Mechatronics, Joanneum Research, Lakeside B13b, 9020 Klagenfurt am Wörthersee, Austria; mathias.brandstoetter@joanneum.at
\textsuperscript{3} Department of Mechatronics, Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic; Ivan.Virgala@tuke.sk

* Correspondence: tomas.kot@vsb.cz

Featured Application: The proposed method is useful in applications where it is not possible to use a standard industrial robot because of space restrictions and it is necessary to design a custom robotic manipulator.

Abstract: In situations of a confined workplace with a lot of obstacles and a complicated required trajectory of the endpoint of an industrial or collaborative robot, it may be impossible to find a suitable robot and its position within the workplace to fulfill the given task. In some cases, it could be favorable to design a custom manipulator arm with an unusual kinematic structure or shapes of some of its links. This article presents a novel way of finding the optimal lengths and shapes of two crucial links of a manipulator arm, where the target lengths are as short as possible to reduce mass, and the shape in the form of a Bézier curve is chosen to avoid collisions. The chosen type of kinematic structure of the manipulator arm is fixed and is based on the most typical structure of existing industrial robots, with six degrees of freedom. Two algorithm variants were proposed; one method uses iterations to find the solution based on in-depth collision analysis, and the second method uses the particle swarm optimization algorithm. Both methods were implemented in a simulation system and verified in several testing workplaces.

Keywords: robot; manipulator; collision avoidance; Bézier curve; PSO

1. Introduction

The number of industrial robots applied in the industry has been steadily increasing for many years. Nowadays, robots are also considered in situations where the application of an industrial robot was unimaginable in the past. A big role in this is played by the introduction of collaborative robots that can operate in close proximity to human workers without the need for expensive and space-demanding safety measures and can even directly work together with a human on a particular task.

However, there are still situations where the presence of a human operator in the workplace is not the main issue, and the workplace itself is confined, with inadequate space for a standard industrial or collaborative robot—a small robot is not able to reach the whole trajectory, but a large robot collides with some obstacles in the workplace. In cases of a given complicated workplace, it may be hard to plan a trajectory for the industrial or collaborative robot to avoid collisions and keep the desired cycle time. Sometimes, it may even be impossible, making it necessary to change the layout of the workplace.

In cases when the workplace cannot be rearranged, it may be necessary to design a custom robot with a specific kinematic structure or with unusually shaped arms to
avoid collisions while keeping the large reach of the robot. The research described in this paper focuses particularly on changing the shape of the arms while keeping the same type of kinematic structure, which is inspired by the most common structure of existing industrial robots.

1.1. Overview of Related Work

The following overview focuses on articles related to the topic of this paper—manipulator arm collision avoidance and manipulators with general shapes of arms. The most relevant articles with use cases that were as close to our research as possible were selected.

Collision avoidance for industrial robotic manipulators is typically associated with the selection of a proper trajectory. This can be done either manually during robot programming (teaching), which requires a lot of time and experience, or, quite effectively, by autonomous path planning in the configuration space [1–6]. Path planning can also be used in dynamically changing environments [7], where it can be beneficial to utilize some innovative methods, such as vision-based planning using neural networks [8].

As far as manipulators with a general shape of the links are concerned, there has been some related research, mostly focusing on theoretical and some practical issues of such manipulators—for example, real-time collision-free trajectory planning for manipulators with arbitrary shapes in point-to-point tasks [9], where the authors proposed a novel obstacle avoidance algorithm using a signed distance function to deal with collisions on the whole body of the manipulator.

Changing the shape and kinematic structure of a robot has been used to allow adaptation of the same robot for different tasks, effectively replacing the need for additional degrees of freedom with the possibility to manually change the kinematic structure as needed. This principle is applied to a low-cost home service robot with four degrees of freedom composed of rigid joints and two deformable links [10–12]. The deformable manipulator can create a relatively dexterous and extended workspace with fewer joints by bending its deformable links. A kinematic chain that allows mechanical modifications to match the desired Denavit–Hartenberg parameter set in a wide range is introduced in [13].

Another common topic of research is continuum robots; for example, the authors in [14] try to deal with the problem that, despite extensive research in kinematic and dynamic modeling, the inaccuracies and shape deformation of the robot due to unknown loads and collisions with the anatomy make shape sensing important for intraoperative navigation. To address this issue, vision-based solutions have been explored to find the actual shape of the robot. A similar issue regarding finding the actual shape of a surgical robot during a medical procedure using a small number of X-ray projection images has been addressed in [15,16].

General key issues involved in the design and implementation of continuum robots and the potential of these types of robots for enhanced productivity in novel applications are discussed in [17]. The challenges related to the control of continuum robots (stability and performance) emerging from the structural flexibility of such robots are addressed in [18] by presenting a robust tracking control approach using interval arithmetic and in [19] by a robust control for continuum robots that are modeled by Cosserat Rod theory.

So-called soft robots are, to some extent, also related to this research. The design and modeling of joints for inflatable robotic arms for long-range inspection are investigated in [20]. The authors in [21] propose a novel soft inflatable arm for telepresence robots using a common and low-cost inflatable material with a low weight. The focus of this research is on joint compressed models, joint kinematic models, and kinematic models for the whole inflatable robot arm.

As far as the possible mechanical construction of an arm with a changeable general shape is concerned, some research projects have focused on finding a method of making an object that can transform from a flexible into a rigid state and vice versa at will. Brandstötter et al. [13] present an arm containing a fusible alloy with a heating sleeve inside a rubber skin; the arm becomes soft when the alloy is heated up by means of an
electrical current, and it becomes rigid when it cools down. A different principle with similar results was proposed by Clark and Rojas [22], who designed a malleable robot structure using layers of mylar sheets cut into flaps, where the transition between the soft and rigid states is done by changing the pressure inside the arm.

1.2. New Approach

Although industrial robots are universal systems, in real applications, they usually operate as single-purpose machines and repeat the same task in an enclosed work cell that represents a completely static environment (except for well-defined material transfer). In some cases, using a commercially available universal industrial robot can be very problematic, and could, for example, involve a much larger robot than theoretically necessary.

As can be seen from the presented overview of related work, collision avoidance of industrial robots is commonly done by changing the layout of the workplace, changing the placement of the robot, changing the trajectory during programming, or path-planning, but not by changing the shape of the robot itself.

The biggest fields of related research are soft robots and continuum robots. This is, however, a different topic—these systems can change the kinematic structure and shape drastically, even during operation, and the shape changing can be the source of the technological movement (typical for some inflatable robots). These robots are not rigid enough for tasks meant for an industrial robot.

Manual modification of the shape of some parts of a robotic arm, as described in [10–12], is close to our research, but these arms have some stiffness issues and can bend uncontrollably at higher loads. Each shape modification also changes the kinematic structure, which makes accurate control quite complicated—as can be seen from the articles focusing on control of continuum robots [14–16,18,19]. The advantage of such robots is that the shape can be changed as needed without having to manufacture a completely new arm. This advantage can be useful in some situations, especially in service robotics, and can outweigh the disadvantages. In industrial applications, however, the ease of precise and reliable control of the robot is crucial.

Our novel approach to collision avoidance in a static environment involves the use of a specific type of kinematic structure with six degrees of freedom (typical for commercial robots, as will be explained further) and changing the length and shape of the two main links of the manipulator (the lower and upper arm) while keeping the relative orientation of joints the same—thus also keeping the same type of kinematic structure. As can be seen from the overview of related work, this is an innovative approach, not just a modified method. The aim of this work is to present some possible methods for this approach and to demonstrate its possible use, because a custom-made manipulator could provide better performance parameters than a universal manipulator.

The custom-shaped arms are supposed to be designed and manufactured in advance for particular tasks, without the ability to be changed later (they can, however, be replaced by differently shaped parts when needed). It is also possible to utilize some of the above-mentioned methods [13,22] that allow changing of the shape at will.

2. Kinematic Structure of the Robot

A survey on the types of kinematic structures typically used in collaborative and industrial robots by the selected well-known manufacturers (namely ABB, Fanuc, Yaskawa, Kawasaki, Kuka, Stäubli, and Universal Robots) shows [23] that approximately 76% of the robot series use some minor variations of the same kinematic structure (see Figure 1). This structure has six degrees of freedom with six rotational joints in a specific sequence that allow simple calculation of inverse kinematics and a good maneuverability, while also providing an easy mechanical design of the arm. The Denavit–Hartenberg parameters [24,25] uniquely describing this kinematic structure are listed in Table 1.
Figure 1. Kinematic structure of the robot with local coordinate systems based on the Denavit–Hartenberg principle.

Table 1. Denavit–Hartenberg parameters of the chosen kinematic structure ($q_1$...$q_6$ are the joint variables).

| $i$ | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|-----|------------|------|------|--------|
| 0   | 0          | $d_0$ | 0    | 0      |
| 1   | $q_1$      | 0    | 0    | $\pi/2$|
| 2   | $q_2$      | 0    | $a_2$| 0      |
| 3   | $q_3$      | 0    | 0    | $\pi/2$|
| 4   | $q_4$      | $d_4$| 0    | $-\pi/2$|
| 5   | $q_5$      | 0    | 0    | $\pi/2$|
| 6   | $q_6$      | $d_6$| 0    | 0      |

The length of the vertical link $|P_gP_0| = d_0$ (see Figure 1 and Table 1) does not have to be changed to make some meaningful modifications to the properties of the robot because the same effect can be achieved by mounting the robot higher or lower. The last three degrees of freedom ($q_4, q_5, q_6, |P_5P_6|$) are responsible mainly for the orientation of the TCP (tool center point) and are considered as an integrated mechanical unit.

Important are the lower arm $|P_1P_2| = a_2$ and the upper arm $|P_3P_4| = d_4$, which are mainly responsible for the position of the TCP. Making these lengths larger extends the workspace of the robot. However, making these lengths smaller reduces the torques and makes the robot lighter, so in theory, the arms should be as short as possible. These two links are also the links that are most commonly responsible for collisions of the robot with some obstacles in the workplace (this will be discussed later), so these links are the best candidates when shape changing is considered for collision avoidance.

Arm Shape Geometry

The idea of this work is to replace the straight manipulator arm shape (line) with a curve that allows better collision avoidance in some situations. For the implementation
of the optimization algorithm, it is necessary to choose a curve that can be fully and un-
ambiguously described by a mathematical function—for instance, a part of a conic section
(circular of elliptic arc, hyperbola, parabola), a general polynomial, or a complex spline
(B-spline, T-spline, Bézier curve, etc.). Examples of some possible shapes are displayed
in Figure 2. The curves have different properties, but it can be anticipated that none of
the shapes will represent a generally better solution as far as collision avoidance is con-
cerned; the best shape can be selected only when dealing with one concrete situation. For a
universal solution, we will select the shape under other conditions.

![Figure 2. Examples of several possible simple shapes for an arm link (blue and cyan show cubic
Bézier curves with different pointedness near the middle point).](image)

Although this stage of the research does not include physical verification of the
results, it is important to ensure that it will be possible to make a physical prototype later.
The circular arc and quadratic Bézier curve (see Figure 2) present a significant complication
related to the mechanical construction of the joints located at the ends of the curved arm
(represented by circles)—the angle of approach (tangent) differs with varying amounts
of bending of the arm, which means that for each version of the arm, a different flange
connecting the arm to the joint would have to be designed and manufactured. This problem
is eliminated by using, for instance, a combination of two cubic Bézier curves, as in this
case, the tangents at the outer control points are always pointing in the direction of the
straight line connecting the joints.

The chosen spline consists of two cubic Bézier curves connected in the middle
(see Figure 3). The first curve (green) is described by the control points \( A, Q_1, Q_2, D, \) and the
second one (blue) by control points \( D, Q_3, Q_4, B. \) The points \( A \) and \( B \) theoretically lie at
the corresponding joints of the robot—\( A = P_1, B = P_2 \) for the lower arm, and \( A = P_3, \)
\( B = P_4 \) for the upper arm (see also Figure 1), but there will have to be some offset given
by the construction of the joint. However, in this article, we consider this offset to be
zero for simplicity. Vector \( \vec{e} \) is the result of the optimization algorithm (see further) and
represents the required bending amount and direction in a plane perpendicular to \( AB. \)
The control points for the Bézier curves are calculated as follows:

\[
\begin{align*}
C &= \frac{B + A}{2}, \\
D &= C + \vec{e}, \\
\vec{p} &= \frac{B - A}{|AB|}, \\
Q_1 &= A + m \cdot \vec{p}, \\
Q_2 &= D - n \cdot \vec{p}, \\
Q_3 &= D + n \cdot \vec{p}, \\
Q_4 &= B - m \cdot \vec{p},
\end{align*}
\]

where \( m \) and \( n \) are scalars describing the pointedness of the curve near the particular
boundary control point. The values of \( m \) and \( n \) can be constant or can be changed by
the optimization algorithm. Finally, the curves are described by the following parametric
equations for \( 0 \leq t \leq 1: \)
\[ Z(t) = (1-t)^3 A + 3(1-t)^2 t Q_1 + 3(1-t) t^2 Q_2 + t^3 D, \] (8)
\[ Z'(t) = (1-t)^3 D + 3(1-t)^2 t Q_3 + 3(1-t) t^2 Q_4 + t^3 B. \] (9)

Figure 3. Geometry of the chosen shape for an arm link consisting of two cubic Bézier curves; \(A\) and \(B\) represent locations of the two joints.

3. Simulation Model

To simulate and verify the idea of using curved robot arms to avoid collisions, a simulation model was designed and implemented as an application in C++ with 3D visual feedback based on the Direct3D graphics API (Application programming interface). The use of our custom-built simulation system allowed a huge degree of flexibility and speed compared to existing simulation systems.

3.1. Workplace and Trajectory

A trajectory is a 3D curve (path) that defines the required robot TCP movement. The trajectory is given by the actual task and is considered immutable in this research. The trajectory is defined by key points, where each point has the position and orientation (3 + 3 values), the velocity of the TCP, and the type of segment that connects this point with the previous one (linear or arc). This type of trajectory definition follows the common principle used in industrial robot teaching and programming.

For the simulation, the trajectory definition in the form of key points is replaced by a set of points in a constant time distance (0.005 s), where each point has only position and orientation (transformation matrix of the TCP). During the simulation, the robot TCP is placed into each of these points and inverse kinematics are calculated to get the joint angles of the robot (see Section 3.2).

In addition to the trajectory, the workplace is mainly defined by the objects that act as obstacles for the robot. Obstacles are used for collision checking and can be defined by various methods, for example, as bounding boxes, convex hulls, general meshes, or point clouds.

For simplicity (while also maintaining high flexibility), this research uses oriented bounding boxes to model the obstacles, which is mostly suitable for simple obstacles, forbidden areas, and borders like walls, ceilings, and floors.

3.2. Robot

The robot follows a trajectory given by its TCP, so it is necessary to calculate the inverse kinematics (IK) to get joint variables from the TCP position and orientation. As the kinematic structure of the robot is given and known, an analytical method of IK can be used. The calculation of IK with an analytical method is extremely fast and stable; the solution is calculated in one pass (it is not an iterative method). Changing the shape of the arms does not affect the IK calculation because the joints stay in the same relative positions. Changing the length of the arms affects just two constants in the algorithm—two Denavit–Hartenberg (D-H) parameters, namely \(a_2\) and \(d_4\) (see Table 1).

The first step in the IK calculation is getting the \(q_1\) value. Based on the known TCP position \(P_6\), we calculate point \(P_4\) (the origin of LCS4—local coordinate system of the fourth joint) as

\[ P_4 = P_6 - z_6 \cdot d_6, \] (10)
where \( d_6 = |P_4P_6| \) is a D-H parameter. Using the components of point \( P_4 = [x_4, y_4, z_4] \), we can write
\[
q_1 = \text{atan2}(y_4, x_4). \tag{11}
\]

Next, it is possible to calculate
\[
z'_4 = z_4 - d_0, \tag{12}
\]
\[
r_4 = \sqrt{x_4^2 + y_4^2}, \tag{13}
\]
where \( d_0 = |P_2P_0| \) is a D-H parameter. Further calculations are done according to Figure 4:
\[
p = \sqrt{r_4^2 + z'_4^2}, \quad p \leq a_2 + d_4, \tag{14}
\]
where \( a_2 = |P_1P_2| \) and \( d_4 = |P_2P_4| \) are D-H parameters. The mentioned condition is necessary for verification of whether the arm is able to reach the desired target—if not, the calculation can be aborted here with an error.

![Figure 4](image.png)

**Figure 4.** Inverse kinematics of the chosen robot structure—diagram for the calculation of \( q_2 \) and \( q_3 \) (two different solutions).

As can be seen in Figure 4, for the given points \( P_1 \) and \( P_4 \), it is possible to find two different solutions for the positions of \( P_2 \). The calculations of the following auxiliary angles are identical for both cases:
\[
\delta = \text{atan2}(z'_4, r_4), \tag{15}
\]
\[
\alpha = \text{acos} \left( -\frac{p^2 + a_2^2 + d_4^2}{2a_2d_4} \right), \tag{16}
\]
\[
\beta = \text{acos} \left( -\frac{d_4^2 + a_2^2 + p^2}{2a_2p} \right). \tag{17}
\]

However, the way of expressing \( q_2 \) and \( q_3 \) is different—for the first solution, it is
\[
q_2 = \delta + \beta, \quad q_3 = \alpha - \frac{\pi}{2}, \tag{18}
\]
and for the second solution, it is
\[
q_2 = \delta - \beta, \quad q_3 = \frac{3\pi}{2} - \alpha. \tag{19}
\]

At this stage, it is possible to calculate the transformation matrices \( T_{x}^0, T_{y}^0(q_1), T_{x}^1(q_2), \) and \( T_{x}^2(q_3) \) from the Denavit–Hartenberg formula:
\[
T_{i-1}^i = \begin{bmatrix}
\cos \theta_i & -\sin \theta_i & 0 & a_i \\
\sin \theta_i & \cos \theta_i & 0 & -a_i \sin \theta_i \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}. \tag{20}
\]
and subsequently, to apply forward kinematics and get the positions and orientations of the coordinate systems LCS₀, LCS₁, LCS₂, and LCS₃.

The remaining joint variables can be found using vector algebra. The value of \( q₅ \) is given simply as the angle between unit vectors \( \vec{z}_₆ \) (known as the IK input) and \( \vec{z}_₃ \) (already known as an axis of LCS₃):

\[
q₅ = \cos(\vec{z}_₃ \cdot \vec{z}_₆).
\]  

(21)

Afterward, we find the projection of the vector \( \vec{z}_₆ \) on the vector \( \vec{z}_₃ \) as

\[
\vec{m} = (\vec{z}_₆ \cdot \vec{z}_₃)\vec{z}_₃,
\]  

(22)

and the normalized perpendicular vector from the vector \( \vec{z}_₆ \) to the vector \( \vec{z}_₃ \):

\[
\vec{n} = \frac{\vec{z}_₆ - \vec{m}}{|\vec{z}_₆ - \vec{m}|}.
\]  

(23)

The angle between \( \vec{n} \) and the axis \( x₃ \) represents the value \( q₄ \) (the reference vector \( \vec{z}_₃ \) is used to distinguish between an obtuse and acute angle):

\[
q₄ = \begin{cases} 
\cos(\vec{x}_₃ \cdot \vec{n}), & \text{if } (\vec{x}_₃ \times \vec{n}) \cdot \vec{z}_₃ \geq 0 \\
-\cos(\vec{x}_₃ \cdot \vec{n}), & \text{if } (\vec{x}_₃ \times \vec{n}) \cdot \vec{z}_₃ < 0 
\end{cases}
\]  

(24)

Now, it is possible to calculate the transformation matrix \( T₆^₃(q₄) \) and the position and orientation of LCS₄, and to find the last missing value \( q₆ \) as the angle between vectors \( \vec{y}_₅ \) and \( \vec{y}_₆ \):

\[
q₆ = \begin{cases} 
\cos(\vec{y}_₅ \cdot \vec{y}_₆), & \text{if } (\vec{y}_₅ \times \vec{y}_₆) \cdot \vec{z}_₆ \geq 0 \\
-\cos(\vec{y}_₅ \cdot \vec{y}_₆), & \text{if } (\vec{y}_₅ \times \vec{y}_₆) \cdot \vec{z}_₆ < 0 
\end{cases}
\]  

(25)

The last step is a check of whether all values \( q₁, q₂, \ldots, q₆ \) lie in the allowed ranges given by mechanical limits of the joints.

The condition that has to be verified to find if the robot in its current form can fulfill the given task is that the robot is able to reach all parts of the trajectory. To do this, the TCP is successively placed into each point of the trajectory, and all of the following are tested:

- IK calculation is done to verify that it has a valid solution (if it does not, the robot is not able to reach the point with its end effector).
- Joint variables are checked against the limits (if at least one is out, the robot is not able to reach the point with its end effector).
- Joint velocities are calculated as symmetric derivatives of the joint angles around this point and checked against the corresponding velocity limits (if at least one value is over the limit, the robot is not able to achieve the required end-point velocity).
- Collisions of the robot parts are checked against the robot itself and also against the workspace obstacles (see the Section 3.3).

It is also necessary to verify that all points of the trajectory are reachable using the same solution variant of the IK (see Figure 4) because the robot cannot change the configuration abruptly during movement. Only two of the eight solutions of the IK are taken into account because the other solutions do not lead to a different geometry of the manipulator links, and, thus, cannot provide any advantage when collision avoidance is considered. The omitted solutions are related to calculation of \( q₁ \) (11) and \( q₅ \) (21).

### 3.3. Collision Detection

The actual mechanical structure of the robot is considered unknown in this stage, so the robot has to be replaced with simplified bounding volumes. A combination of bounding boxes and bounding spheres was chosen—boxes are used for the parts that do not change shape during the simulation, while spheres cover the lower and upper arm, which can bend (see Figure 5). The positions of the centers of all bounding spheres are
calculated using the Equations (8) and (9), where the parameter $t \in (0; 1)$ for each sphere is given by the order and the total number of spheres. The number of spheres depends on the effective length of the curve so that the spheres cover the whole shape without big gaps or overlaps.

Figure 5. Example of a simple workplace and a robot with the selected kinematic structure, where both the lower and upper arms bend according to the cubic Bézier curve. (a) Schematic model of the robot. (b) Visualization of collision bounding volumes.

In each simulation step (each point of the trajectory), the robot’s bounding boxes and spheres are transformed according to the corresponding robot link. Intersections of box–box or sphere–box pairs are checked using the standard separating axis theorem.

4. Optimization Algorithms

Given the robot’s kinematic structure and simulation model described above, the goal is to find the shape of the lower and upper arms of the robot with the following conditions:

- The trajectory and the layout of the workplace, including the location of the robot base, are fixed;
- The robot must be able to reach all points of the trajectory without collisions;
- The dimensions of the robot should be as small as possible.

The shape of each the lower and upper arm is described by the following parameters:

- The linear distance between the boundary joints ($|AB|$ in Figure 3);
- The amount of bending ($|CD|$ in Figure 3);
- The direction of bending (vector $\vec{e}$ in Figure 3).

Two methods were proposed, implemented, and tested: an iterative algorithm that relies on an analysis of the character of the collisions to choose the next step of shape modification, and an algorithm based on the particle swarm optimization (PSO) method that does not require any additional information about the collisions.

4.1. Method 1—Iterative Algorithm with Collision Analysis

Collision checking requires special care in this case because the algorithm of this method relies on a simple analysis and classification of collisions.

4.1.1. Types of Collisions

Collisions with an obstacle in the workplace can occur at any part of the robot. The significance of the collision and possible ways of dealing with it are different, so we need to classify the collisions into five categories: C1, C2, ..., C5 (see Figure 6 and Table 2).
C1: Collisions of the base of the robot, the first link, and the first two joints (see an example in Figure 6b) are not related to the lengths and shapes of the arms higher on the kinematic structure, and can be eliminated only by changing the location of the robot, which is not a subject of this paper.

C2, C4: Collisions within the lower or upper arm (Figure 6c,e) can be potentially fixed by changing the length or shape of the arm—making the arm elude the obstacle. This is the main focus of this paper.

C3: Collisions that occur near the joint (elbow) between the lower and upper arm (see an example on Figure 6d) can potentially be fixed by changing the robot’s location or by changing the lengths of the lower and upper arms. In this work, the robot’s location is fixed; changing the arms’ lengths is therefore the option to be considered.

C5: Collisions at or near the end effector of the robot or the wrist (the highest three degrees of freedom; see Figure 6f) cannot be fixed in most cases because the robot must go to that place to fulfill the given task. An option is to change the required orientation of the TCP, unless it is forced by the technology or by other constraints. This is not a subject of this work.

Figure 6. Simplified representation of the robot’s kinematic structure with zones corresponding to the collision categories, and examples of categories C1–C5.
Table 2. Overview of five categories of collisions with possible ways of removing them; only cases marked by an asterisk (*) are considered in this research.

| Possibly Fixable by            | C1 | C2 | C3 | C4 | C5 |
|--------------------------------|----|----|----|----|----|
| Modify trajectory              | no | yes| yes| yes| yes|
| Change robot position          | yes| yes| yes| yes| no |
| Modify arms lengths            | no | yes*| yes*| yes*| no |
| Modify arms shapes             | no | yes*| no | yes*| no |

As already mentioned in Section 3.3, the lower and upper arms use collision spheres that cover the length of the arm. The type of a particular collision is determined as follows:

- The first 10% of the spheres of the lower arm (counted from the bottom; rounded up) are considered too close to the shoulder joint and create C1 collisions.
- The last 10% of the spheres of the lower arm and the first 10% of the spheres of the upper arm (all counted from the bottom; rounded up) are considered too close to the elbow joint and create C3 collisions.
- The last 10% of the spheres of the upper arm (counted from the bottom; rounded up) are considered too close to the wrist joint and create C5 collisions.
- All other spheres create C2 or C4 collisions.

4.1.2. Collision Events

In addition to the collision type, the algorithm also analyzes the required direction of the bend “out of the collision zone.” To achieve this, collisions of the spheres of the upper and lower arms are stored as collision events. A collision event is a continuous intersection of a particular sphere and a particular workspace box. During each event, the direction vector \( \vec{p}(t_i) \) from the center of the box (node) \( B \) to the center of the sphere \( C(t_i) \) is recorded in each time step \( t_i \) (see Figure 7) as

\[
\vec{p}(t_i) = C(t_i) - B, \tag{26}
\]

Afterward, the vectors \( \vec{p}(t_i) \) are averaged throughout the event, which gives an imaginary spring force that will later be used to bend the arm:

\[
\vec{p}_{\text{avg}} = \frac{1}{n} \sum_{i=t_0}^{n} \vec{p}(t_i), \tag{27}
\]

where \( t_0 \) denotes the start of the particular collision event (not the start of the simulation) and \( n \) represents the length of the collision event measured in time steps of the simulation. If the intersection ends and starts again later during the trajectory, a new event is created.

Figure 8 shows the result of a simple testing trajectory verification. Green points of the trajectory represent sections that the robot was able to reach with no collisions and yellow points mean that a collision occurred. The arm bounding spheres rendered yellow mean that there exists at least one collision event for the sphere. A red arrow is drawn for each collision event of a sphere; it represents the vector \( \vec{p}_{\text{avg}} \) or the imaginary force pushing the sphere out of the collision. Multiple arrows originating from the same sphere mean that the sphere had several collision events during the trajectory.
Figure 7. Example of a collision event of one arm bounding sphere and one workplace bounding box; the sphere is drawn in five simulation time steps: $t_{-1}$, $t_0$, ..., $t_3$. The event begins at $t_0$.

Figure 8. Simple result of a simulation showing colliding bounding spheres with collision events (red arrows).

The collision data, including the collision events, are used as input for the robot adaptation algorithm. The algorithm starts by finding the smallest possible arm lengths that can reach all points of the trajectory without taking collisions into account. This can easily be done by finding the largest value of $p$ from (14) and then choosing

$$a_2 = k \cdot p,$$  
$$d_4 = p - a_2,$$  

where $k \in (0; 1)$ is a coefficient describing the constant ratio between $a_2$ and $d_4$ (typically $k \geq 0.5$ for $a_2 \geq d_4$). Note that for simplicity, the lengths of the lower and upper arms are tied together; however, their bending is individual. The whole algorithm can be described by the following sequence of operations:

1. Start with straight arms of the smallest theoretically possible length.
2. Run the simulation for every point of the trajectory.
3. If no collision occurred $\Rightarrow$ solution found, end.
4. If at least one C1 or C5 collision occurred $\Rightarrow$ impossible to solve, end.
5. If at least one C3 collision occurred $\Rightarrow$ remove any existing arm bending, make the lower and upper arms longer by a chosen value, and go to step 8.
6. If some C2 or C4 collisions exist ⇒ use the force vectors $\mathbf{p}_{\text{avg}}$ calculated from collision events to bend the arm by a chosen amount (the bending is added to any already existing bending).

7. If the overall bending of an arm is larger than a set threshold ⇒ remove the bending, and make the lower and upper arms longer by a chosen value.

8. If the arm lengths are larger than a set threshold ⇒ impossible to solve, end.

9. Go to step 2.

As can be seen, the proposed algorithm is iterative. The force vectors $\mathbf{p}_{\text{avg}}$ from collision events are used only to determine the direction in which the bending of the arm is increased at the particular iteration, not the total amount of bend. The bending and/or lengths of arms increase progressively until a solution is found or until boundary conditions are reached.

4.2. Method 2—PSO Algorithm without Collision Analysis

The algorithm based on the particle swarm optimization (PSO) method relies purely on the ability of PSO to find the solution to the given problem; there is no need for the detailed collision classification described above, and collision events are not recorded. Collisions still have to be checked, but the feedback is only in the binary form (no collision occurred; some collision occurred). This algorithm starts directly with the arms in any configuration (unlike the previous algorithm, which required the arms to initially be as short as possible). PSO [26] solves the problem using a population of particles that move in the search space in a way that eventually converges to the optimal solution based on the specified fitness function. This particular optimization method was chosen because it is one of the commonly used methods in various applications and because it can be implemented efficiently. The aim of this work is not to compare various optimization methods, but merely to demonstrate that optimization methods can be used for our purpose.

The search space in our case is six-dimensional, with the following variables, all normalized to the range $\langle -1; 1 \rangle$:

- Distance between the joints of the lower arm ($d_L$),
- Bending direction of the lower arm ($b_{dL}$),
- Bending amount of the lower arm ($b_{aL}$),
- Distance between the joints of the upper arm ($d_U$),
- Bending direction of the upper arm ($b_{dU}$),
- Bending amount of the upper arm ($b_{aU}$).

The fitness function was chosen with respect to the goal of finding the smallest robot:

$$F = 0.2d_L + 0.2d_U + 0.3b_{aL} + 0.3b_{aU},$$

where all the individual factors (as mentioned above), and thus also the resulting value of $F$, are in the range $\langle -1; 1 \rangle$. The chosen weights give slightly more importance to the bending amount (0.3 for each link) than to the distance between the joints (0.2 for each link), which means that even a slightly longer straight arm is considered better than an arm with a curve.

The PSO variation implemented uses the fully connected neighborhood structure (also called the Global Best PSO), which typically provides the fastest convergence speed, but can be worse in exploring outside the locally optimal regions [27,28]. The termination of the iterative process is configured according to the “no improvement” method, with a slight alteration: A particle becomes inactive (not updated anymore) if its personal best fitness function value has not improved by more than 0.0001 over the last 10 eras; the whole swarm calculation stops when no particles are updating. To increase the initial coverage and diversity of the particles, a large number of 1000 particles was used; this simplified approach provides acceptable results, but could preferably be replaced by some of the Multi-Start PSO methods and a lower number of particles [28]. Particles that correspond to
a robot configuration that is not able to fulfill the given task (i.e., the robot is in a collision or the IK failed) are always given the fitness value $F = 10,000$.

5. Results

Three simplified testing workplaces will be presented in this section, in which the optimal solution is discovered using both of the above-described methods. The initial states of all testing workplaces are shown in Figure 9; each successive example has a higher difficulty. A detailed comparison of the results for all three testing workplaces is available in Table 3.

![Figures 9](image_url)

**Figure 9.** Three simplified testing workplaces with the robot in collisions in the initial state. (a) Simple obstacles in front of the robot and above the robot. (b) More obstacles with less space between them. (c) A confined workplace with the robot behind a rectangular opening in the wall.

The results for the first (simplest) testing workplace are presented in Figure 10 with the arm visualized as a curved tetrahedron. Figure 11a shows the time progression of all three optimized parameters for the iterative method: In step 1, the length of both arms is shortened; in step 2, the bending amount of the upper arm is increased by 0.01 m, and then, finally, in step 3, the bending is increased by another 0.01 m, which is the solution. The PSO method found a similar solution, but it was slightly better according to the fitness function $F$, as the bending amount of the upper arm is smaller. The lower arm stays straight in both cases. The time progression for the best particle is shown on Figure 11b.

The second example (see Figures 9b and 12) shows a more complicated situation, with both arms having multiple collision events at the initial configuration. The arms were made longer by the algorithms, and both the lower and upper arms have a significant curve in a different orientation. The iterative method required 48 iterations in this case (see the graph in Figure 13a)—first, the arms were shortened, which caused C3 collisions, until the lengths were increased to the final value; then, the bending of the arms was increased until there were no C2 and C4 collisions. The PSO method required 65 generations (see the graph in Figure 13b).
Figure 10. Results for the first testing workplace.

Figure 11. Cont.
Figure 11. Graphs for the first testing workplace. (a) Modifications made to the robot structure in individual iteration steps during the optimization algorithm (the sums of arm lengths and individual lower and upper arm bending amounts are shown). (b) Selected parameters of the best particle with the particle swarm optimization (PSO) algorithm (the lower and upper arm lengths and bending amounts and the fitness function value are shown).

Figure 12. Results for the second testing workplace.
Figure 13. Graphs for the second testing workplace. (a) Modifications made to the robot structure in individual iteration steps during the optimization algorithm (the sums of arm lengths and individual lower and upper arm bending amounts are shown). (b) Selected parameters of the best particle with the PSO algorithm (the lower and upper arm lengths and bending amounts and the fitness function value are shown).
The results for the third example (Figure 9c) are shown in Figures 14 and 15a,b. This example has a very tight solution space, which turned out to be problematic for the PSO method. The particles start at random positions in the six-dimensional space, and if none of the particles in the starting position have a valid solution, the particles do not converge to the optimal solution. This required several executions of the algorithm to find a solution. It is also important to mention that the PSO method can converge to a local minimum instead of a global minimum, which can be seen in Figure 14c—although this solution is valid, it is clearly not optimal, and it is possible only because of some simplifications in the workspace model (in a real situation, it would probably be unacceptable to reach above the obstacle).

Figure 14. Results for the third testing workplace.
Figure 15. Graphs for the third testing workplace. (a) Modifications made to the robot structure in individual iteration steps during the optimization algorithm (the sums of arm lengths and individual lower and upper arm bending amounts are shown). (b) Selected parameters of the best particle with the PSO algorithm (the lower and upper arm lengths and bending amounts and the fitness function value are shown).
The important values for all three examples are listed in Table 3. The fitness function was calculated with (30) from the values of joint distances and bending amounts. Smaller values are considered better; a zero for the bending amount represents a straight arm. The time was averaged for five executions on an Intel® Core™ i7-6700K (4.00 GHz) with 16 GB RAM on Windows 10.

Table 3. Results for the testing workplaces, including the time required to run the corresponding optimization algorithm (PSO* denotes the non-optimal solution).

| Example 1 | Example 2 | Example 3 |
|-----------|-----------|-----------|
|           | Iterative | PSO       | Iterative | PSO       | Iterative | PSO       | PSO*       |
| Arm 1 dist. [m] | 0.320 | 0.308 | 0.403 | 0.456 | 0.418 | 0.337 | 0.689 |
| Arm 1 bend. [m] | 0.000 | 0.000 | 0.119 | 0.093 | 0.190 | 0.071 | 0.000 |
| Arm 2 dist. [m] | 0.320 | 0.343 | 0.403 | 0.406 | 0.418 | 0.530 | 0.971 |
| Arm 2 bend. [m] | 0.019 | 0.010 | 0.273 | 0.185 | 0.020 | 0.029 | 0.437 |
| Fitness function | 0.134 | 0.133 | 0.279 | 0.256 | 0.230 | 0.203 | 0.463 |
| Steps or eras | 4 | 60 | 48 | 65 | 20 | 65 | 23 |
| Particles | – | 1000 | – | 1000 | – | 1000 | 1000 |
| Time [s] | 0.2 | 49.1 | 1.4 | 43.5 | 0.6 | 660 | 756 |

6. Discussion

The two proposed methods for finding optimal manipulator arm shapes to avoid collisions were tested on multiple testing workplaces, three of which are presented in this article.

The importance of the results is demonstrated in Figure 16, which shows alternative solutions for the first and second testing workplaces with strictly straight arms (the third workplace does not have any valid solution with straight arms) together with the corresponding solutions acquired using the PSO method. As can be seen, the manipulators are much larger than when using arms with curvature, and the arms must have very specific lengths, so the manipulator would have to be custom made anyway. Moreover, these solutions could be invalid in real conditions, as there can be more obstacles at the top (or even a ceiling), which are not included in the simplified simulation model.

Figure 16. The smallest robots with straight arms able to reach the whole trajectory in the first and second testing workplaces compared to robots with curved arms. (a) First workplace, straight arms. (b) First workplace, curved arms. (c) Second workplace, straight arms. (d) Second workplace, curved arms.

In general, it can be stated that the PSO method proved to be better in finding the optimal solution, as the fitness function value of the result is typically better (lower), but at the expense of much longer simulation times (see Table 3). The PSO method is sometimes prone to finding a non-optimal solution and requires multiple executions to find the global
minimum, which further extends the calculation time. This was demonstrated in the third example (see Figure 14c). The iterative method is limited by using dependent lengths of the lower and upper arm—being able to analyze the situation more deeply and change the lengths individually would require a substantial modification of the algorithm.

PSO was chosen just to demonstrate that it is possible to use a population-based optimization algorithm in this application, and to roughly compare the performance of a general-purpose optimization method with an iterative method designed directly for the particular problem. Thorough analysis and comparison of various PSO variations and configurations, and even different optimization methods, was not the aim of this work, and could be performed in future research. Other optimization algorithms could probably be able to find the solution in a shorter time or find a slightly better solution.

A big advantage of the PSO method is its simplicity—as it is not necessary to perform the in-depth analysis of collisions—and its scalability. In this paper, PSO operated in a six-dimensional space with six parameters of each particle (solution candidate). However, this number can be extended; thus, it is also possible to optimize other parameters of the robot, which will be the focus of further research on this topic by the authors. Although the execution times are much longer for this method, this is typically not a concern, as the times are still in the order of minutes in the worst cases, which will probably be acceptable during the design phase of a robotized workplace. However, it is important to keep in mind that with each added parameter, the simulation times will increase and the chance of not finding any solution (or finding a local minimum) will intensify.

The following possible ways of extending the optimization algorithm are considered:

- Additionally allowing the position of the robot base to be changed, either freely or within some defined limits. This requires the addition of 1, 2, or 3 additional parameters (all or a subset of the x, y, z coordinates).
- Allowing more precise modification of the curve; for example, the tangents near the boundary key points or the middle key point—the parameters m and n in Figure 3. The curve can also be more complex than the combination of two cubic Bézier curves.
- Allowing some minor changes to the trajectory at defined sections that are not strictly given by the technology.
- To better approach reality, it could be also convenient to modify the simulation to only allow certain defined effective lengths of the arm shapes. This would reflect a possible physical implementation using bendable tubes that could be available in specific lengths (for example, 400, 500, 600 mm, etc.)—this restriction binds the values of joint distance and bending amount into a single variable.
- The weight values given to individual factors of the fitness function (30) can be selected differently; this could be the aim of further investigation.

This method has some limitations and drawbacks that must be taken into consideration. Although keeping the same type of kinematic structure of the robot (and, thus, also the IK algorithm) greatly simplifies the control of the real robot, the mass properties of the modified arms will change together with the arm shapes, and dynamic calculations will be affected. The control system of an industrial robot has to take dynamics into account, as the movements are often very quick. Thus, the inertia tensor has to be calculated according to the actual shapes of the arms, and the control system regulator has to be adapted (tuned) appropriately. Another possible problem related to dynamics could be caused by the elasticity of the arm link, especially with a large amount of curvature. This could also be necessary to address by the control system to minimize overshooting and to improve positioning accuracy in general.

The custom-shaped arms are supposed to be designed and manufactured in advance solely for the particular task, so this method is suitable mostly for usage in mass production lines. This process is time-consuming, and the manufacturing of the arm will be more expensive compared to simple straight arms. The links can, however, be replaced by differently shaped parts when needed. The physical construction of a manipulator arm
that allows modifications of the link shapes without the need to manufacture a new piece every time is also the subject of further research.

**Author Contributions:** Conceptualization, T.K. and Z.B.; methodology, T.K. and Z.B.; software, T.K.; validation, T.K., M.B., and Z.B.; investigation, T.K., Z.B., and V.K.; writing—original draft preparation, T.K.; writing—review and editing, Z.B., M.B., V.K., and I.V.; visualization, T.K.; supervision, Z.B. and P.N.; project administration, P.N.; funding acquisition, P.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This article was developed with the support of the project Research Center of Advanced Mechatronic Systems, reg. no. CZ.02.1.01/0.0/0.0/16_019/0000867 in the frame of the Operational Program Research, Development, and Education. This research was also funded by the Slovak Grant Agency VEGA 1/0389/18 “Research on kinematically redundant mechanisms”, and by the specific research project SP2020/141, and was financed by the state budget of the Czech Republic. This research was also funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation, and Technology within the framework of the sponsorship agreement formed for 2019–2021 under the project CredRoS.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

**Abbreviations**

The following abbreviations are used in this manuscript:

- IK Inverse kinematics
- API Application programming interface
- TCP Tool center point
- PSO Particle swarm optimization
- D-H Denavit–Hartenberg

**References**

1. Ting, Y.; Lei, W.; Jar, H. A path planning algorithm for industrial robots. *Comput. Ind. Eng.* 2002, 42, 299–308. [CrossRef]
2. Hsu, D.; Latombe, J.; Motwani, R. Path planning in expansive configuration spaces. In *Proceedings of the International Conference on Robotics and Automation*, Albuquerque, NM, USA, 25 April 1997; Volume 3, pp. 2719–2726.
3. Henrich, D.; Qin, C. Path planning for industrial robot arms—A parallel randomized approach. In *Proceedings of the International Symposium on Intelligent Robotic Systems (SIRS 96)*, Pisa, Italy, 12 October 1996.
4. Van Willigenburg, L.; Hol, C.; van Henten, E. On-line near minimum-time path planning and control of an industrial robot for picking fruits. *Comput. Electron. Agric.* 2004, 44, 223–237. [CrossRef]
5. Fu, B.; Chen, L.; Zhou, Y.; Zheng, D.; Wei, Z.; Dai, J.; Pan, H. An improved A* algorithm for the industrial robot path planning with high success rate and short length. *Robot. Auton. Syst.* 2018, 106, 26–37. [CrossRef]
6. Zhang, H.; Wang, Y.; Zheng, J.; Yu, J. Path planning of industrial robot based on improved RRT algorithm in complex environments. *IEEE Access* 2018, 6, 53296–53306. [CrossRef]
7. Henrich, D.; Wurll, C.; Worn, H. 6 DOF path planning in dynamic environments–a parallel online approach. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation* (Cat. No.98CH36146), Leuven, Belgium, 20–20 May 1998; Volume 1, pp. 330–335.
8. Zeller, M.; Sharma, R.; Schulten, K. Vision-based motion planning of a pneumatic robot using a topology representing neural network. In *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, Dearborn, MI, USA, 15–18 September 1996; pp. 7–12.
9. Xu, S.; Li, G.; Liu, J. Obstacle avoidance for manipulator with arbitrary arm shape using signed distance function. In *Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Kuala Lumpur, Malaysia, 12–15 December 2018; pp. 343–348.
10. Li, G.; Song, D.; Xu, S.; Sun, L.; Liu, J. Kinematic-free orientation control for a deformable manipulator based on the geodesic in rotation group SO(3). *IEEE Robot. Autom. Lett.* 2018, 3, 2432–2438. [CrossRef]
11. Li, G.; Sun, L.; Xu, S.; Song, D.; Liu, J. A hybrid model and kinematic-free control framework for a low-cost deformable manipulator using in home service. In *Proceedings of the 2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Fort Worth, TX, USA, 21–25 August 2016; pp. 1002–1007.
12. Xu, S.; Li, G.; Song, D.; Sun, L.; Liu, J. Real-time shape recognition of a deformable link by using self-organizing map. In Proceedings of the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany, 20–24 August 2018; pp. 586–591.

13. Brandstötter, M.; Gallina, P.; Seriani, S.; Hofbaur, M. Task-dependent structural modifications on reconfigurable general serial manipulators. In Advances in Service and Industrial Robotics; Springer International Publishing: Cham, Switzerland, 2019; pp. 316–324.

14. Vandini, A.; Bergeles, C.; Glockner, B.; Giataganas, P.; Yang, G. Unified tracking and shape estimation for concentric tube robots. IEEE Trans. Robot. 2017, 33, 901–915. [CrossRef]

15. Lobaton, E.J.; Fu, J.; Torres, L.G.; Alterovitz, R. Continuous shape estimation of continuum robots using X-ray images. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 725–732.

16. Vandini, A.; Bergeles, C.; Lin, F.; Yang, G. Vision-based intraoperative shape sensing of concentric tube robots. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 2603–2610.

17. Walker, I.D.; Green, K.E. Continuum Robots. In Encyclopedia of Complexity and Systems Science; Springer: New York, NY, USA, 2009; pp. 1475–1485.

18. Hisch, F.; Giusti, A.; Althoff, M. Robust control of continuum robots using interval arithmetic. IFAC-PapersOnLine 2017, 50, 5660–5665. [CrossRef]

19. Alqumsan, A.A.; Khoo, S.; Norton, M. Robust control of continuum robots using Cosserat rod theory. Mech. Mach. Theory 2019, 131, 48–61. [CrossRef]

20. Li, X.; Sun, K.; Guo, C.; Liu, T.; Liu, H. Design, modeling and characterization of a joint for inflatable robotic arms. Mechatronics 2020, 65, 102311. [CrossRef]

21. Qi, R.; Khajepour, A.; Melek, W.W.; Lam, T.L.; Xu, Y. Design, kinematics, and control of a multijoint soft inflatable arm for human-safe interaction. IEEE Trans. Robot. 2017, 33, 594–609. [CrossRef]

22. Clark, A.B.; Rojas, N. Stiffness-tuneable limb segment with flexible spine for malleable robots. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 3969–3975.

23. Kot, T. Rapid support for integration of collaborative and industrial robots in small and midsized factories. Habilitation Thesis, VŠB-Technická Univerzita Ostrava, Ostrava, Czech Republic, 2019.

24. Craig, J.J. Introduction to Robotics: Mechanics and Control; Pearson Education, Inc.: Upper Saddle River, NJ, USA, 2005.

25. Lynch, K.M.; Park, F.C. Modern robotics: Mechanics, Planning and Control; Cambridge University Press: Cambridge, UK, 2017.

26. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. Swarm Intell. 2007, 1, 33–57. [CrossRef]

27. Muñoz-Zavala, A.; Hernandez-Aguirre, A.; Villa Diharce, E. A New Neighborhood Topology for the Particle Swarm Optimization Algorithm; Association for Computing Machinery: New York, NY, USA, 2011; pp. 227–247.

28. Engelbrecht, A.P. Computational Intelligence: An Introduction, 2nd ed.; Wiley Publishing: Hoboken, NJ, USA, 2007.