Development of Novel Task-Based Configuration Optimization Methodologies for Modular and Reconfigurable Robots Using Multi-Solution Inverse Kinematic Algorithms

by

Saleh Tabandeh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Modular and Reconfigurable Robots (MRRs) are those designed to address the increasing demand for flexible and versatile manipulators in manufacturing facilities. The term, modularity, indicates that they are constructed by using a limited number of interchangeable standardized modules which can be assembled in different kinematic configurations. Thereby, a wide variety of specialized robots can be built from a set of standard components. The term, reconfigurability, implies that the robots can be disassembled and rearranged to accommodate different products or tasks rather than being replaced.

A set of MRR modules may consist of joints, links, and end-effectors. Different kinematic configurations are achieved by using different joint, link, and end-effector modules and by changing their relative orientation. The number of distinct kinematic configurations, attainable by a set of modules, varies with respect to the size of the module set from several tens to several thousands. Although determining the most suitable configuration for a specific task from a predefined set of modules is a highly nonlinear optimization problem in a hybrid continuous and discrete search space, a solution to this problem is crucial to effectively utilize MRRs in manufacturing facilities.

The objective of this thesis is to develop novel optimization methods that can effectively search the Kinematic Configuration (KC) space to identify the most suitable manipulator for any given task. In specific terms, the goal is to develop and synthesize fast and efficient algorithms for a Task-Based Configuration Optimization (TBCO) from a given set of constraints and optimization criteria. To achieve such efficiency, a TBCO solver, based on Memetic Algorithms (MA), is proposed. MAs are hybrids of Genetic Algorithms (GAs) and local search algorithms. MAs benefit from the exploration abilities of GAs and the exploitation abilities of local search methods simultaneously. Consequently, MAs can significantly enhance the search efficiency of a wide range of optimization problems, including the TBCO. To achieve more optimal solutions, the proposed TBCO utilizes all the solutions of the Inverse Kinematics(IK) problem.

Another objective is to develop a method for incorporating the multiple solutions of the IK problem in a trajectory optimization framework. The output of the proposed trajectory optimization method consists of a sequence of desired tasks and a single IK solution to reach each task point. Moreover, the total cost of the optimized trajectory is utilized in the TBCO as a performance measure, providing a means to identify kinematic configurations with more efficient optimized trajectories.

The final objective is to develop novel IK solvers which are both general and complete. Generality means that the solvers are applicable to all the kinematic configurations which can be assembled from the available module inventory. Completeness entails the algorithm can obtain all the possible IK solutions.

Acknowledgements

I thank my supervisors, Dr. Christopher Clark and Dr. William Melek, for their insight, guidance, and encouragement. Chris' hours of brainstorming taught me how to think critically and creatively. His enthusiasm inspired me to see my research as an amazing journey. Also, I am grateful to William's continuous support and trust, which enabled me to take advantage of various industrial, academic, and research interests and opportunities. He provided his timely advice, supervision, and support whenever it was needed. Thank you both.

Many thanks should go to Christian Sterner, and the Sterner Automation for their support at each stage of my research. Branislav Bezak and Ravi Balachandran were there for me with their technical assistance and friendship.

Throughout my time at the University of Waterloo, numerous colleagues and friends contributed to my unforgettable experiences. I would like to particularly thank Dennis Turriff, Orang Vahid, and Nima Eslaminasab. In addition, I want to express my appreciation to Afarin Mansouri, Anahita Ghazanfari, Ali Razavi, Babak Ebrahimi, Ehsan Mohammadi, Golzar Taravati, Kiarash Narimani, Maryam Kafi, Mehrnaz Motiee, Nahal Doroudian, Sara Behjat, Siamak Arzanpour, and Touraj Mohammadi for their friendship.

Most of all, I thank you, my family, namely Vahideh Saidi, Shokrollah Tabandeh, Negar Tabandeh, and Khashayar Hadipour for your unwavering love, support, and understanding.

Dedication

This thesis is dedicated to my parents who have raised me to be who I am today. You have been with me every step of the way, through good times and rough times, and have shared the hardships of this road with me. Thank you for all the unwavering love, guidance, and support, and for instilling in me the confidence that I am capable of doing anything I put my mind to. Thank you for everything.

Cause the drop of knowledge which thou gave us to become united with thy seas,

In my soul there is a drop of knowledge: deliver it from sensuality and from the body's clay,

Before these clays drink it up, before these winds sweep it away.

- Molavi Rumi

Mathnawi, Book I, 1888

Contents

List of Tables

List of Figures

Nomenclature

α_i	Twist angle
β	Expansion ratio of the continuous crossover operator
Δ	Link dimension matrix
$\mathbf{E}_O$	Orientation error
$\mathbf{E}_P$	Positioning error
$\mathbf{m}_n$	n-DOF kinematic configuration consisting only of standard modular joints
$\mathbf{P}_{des}$	Position of the desired end-effector frame in the Cartesian space
$\mathbf{R}_{des}$	Rotation matrix of the desired end-effector frame in the Cartesian space
$\mathbf{T}_{des}$	Homogenous Transformation of the desired end-effector frame
$\mathbb{M}_n$	n-DOF kinematic configuration space
$\mathbb{M}_{\shortparallel,2}$	Set of all $\mathbf{m}_2$ manipulators with parallel joint axes
$\mathbb{M}_{\times,2}$	Set of all $\mathbf{m}_2$ manipulators with intersecting joint axes
$\mathbb{M}_{\shortparallel\!\vdash,2}$	Set of all $\mathbf{m}_2$ manipulators with non-intersecting and non-parallel joint axes
$\mathbb{Q}_o$	Set consisting of all orienting inverse kinematic solutions
$\mathbb{Q}_p$	Set consisting of all positioning inverse kinematic solutions
$\mathbb{Q}_s$	Set consisting of all combined positioning and orienting inverse kinematic solutions
$\mathbb{T}_t$	Task consisted of t task points
ϕ_i	Relative orientation of the i-th joint module
Ψ_{JRO}	Joint Reflection Operator

Σ Kinematic structure matrix

Θ Set of joint angles of a manipulator

Θ^* New inverse kinematic solution extracted from Θ^s

Θ^s Original inverse kinematic solution used to extract Θ^*

θ_i Angle of the i-th joint of the manipulator

θ_{max} Maximum permissible joint angle

θ_{min} Minimum permissible joint angle

a_i Link length

b Size of businessman population

d_i Link offset

$d_{min_{start}}$ Initial minimum allowable distance between the businessman individuals

d_{min} Minimum allowable distance between the businessman individuals

$F_{avg,g}(b)$ Average fitness value of the businessman population in generation g

$F_{fv}(X_i)$ Fitness value of the i-th individual of the GA population

F_{limit} Estimated upper bound of the fitness value

F_{obj} Objective function of the inverse kinematic problem

f_{obj} Efficiency measure in the definition of TBCO

$f_{rch,\mathbb{T}}$ Total reachability error of task $\mathbb{T}$

$f^o_{rch,i}$ Orientation reachability error for the t-th task points

$f^p_{rch,i}$ Position reachability error for the t-th task points

g Genetic Algorithm generation number

g_{max} Maximum allowable Genetic Algorithm generation number

J^{-1} Inverse of matrix J

$J^{\dagger}$ Moore-Penrose pseudoinverse of matrix J

J^T Transpose of matrix J

$J_{\mathbb{T}}$ Task Embedded Jacobian matrix

l_i Length of the i-th link module

l_{max} Maximum allowable length of the link modules

l_{min} Minimum allowable length of the link modules

m_i Type of the i-th joint module

n Degree of freedom of the manipulator

P^i_{des} Translation matrix representing the position of the i-th task point

$P_{ee}(\Theta)$ Translation matrix representing the position of the end-effector at joint angle Θ

$q_{\mathbb{T}}$ Δ search variable vector

R^i_{des} Rotation matrix representing the orientation of the i-th task point

$R_{ee}(\Theta)$ Rotation matrix representing the orientation of the end-effector at joint angle Θ

S^i_j j-th inverse kinematic solution of the i-th task point

T^i_{des} Homogenous Transformation of the i-th task point

$T_{ee}(\Theta)$ Homogenous Transformation of the end-effector at joint angle Θ

W_T Manipulator workspace

X_i i-th individual of the GA population

x_{des} Task points minimal representation

Λ_i Probability of the i-th individual to undergo local search

c_i number of individuals with the same Σ as the i-th individual

C_{pop} Population of children (Offsprings)

e number of elites

E_{pop} Population of elites

f^i_{cons} Optimization constraint of the i-th individual

f_{obj}^{i}	Optimization criteria of the i-th individual
g_{max}	Maximum number of iteration after satisfying the constraints
N_{pop}	Population of the selected individuals to undergo local search
p	Size of the parent pool
P_c	Crossover probability
P_m	Mutation probability
P_{pop}	Population of parents
r	size of population of manipulators
R_{pop}	Population of manipulators

Chapter 1

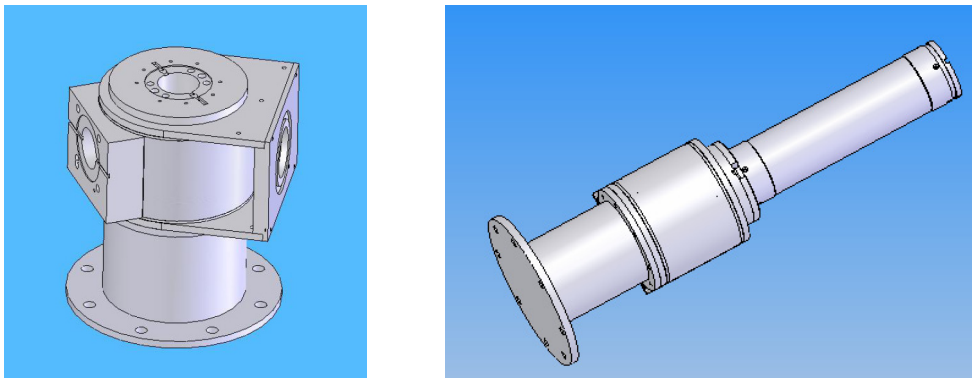
Introduction

1.1 Modular and Reconfigurable Robots

Industrial robot manipulators are the primary building block of many automated manufacturing plants. Most industrial robots consist of a fixed joint and link configuration, and are designed to perform a general set of tasks such as assembly, pick and place, and welding. Although these robots can perform well in many specific workspaces, the robots have a limited adaptability to changes in the environment. In case of a change in the task space or product, numerous problems can occur as a result of the robot singularities, saturation of the torques in the actuators, or mechanical limitations of the joint modules. Consequently, conventional fixed-configuration industrial robots cannot satisfy the increasing demand for Reconfigurable Manufacturing Systems (RMSs).

According to the Visionary Manufacturing Challenges for 2020 by the USA National Research Council, RMSs are the most vital of all priority technology area in manufacturing [1]. One type of robot manipulators whose design is based on this technology is called the Modular and Reconfigurable Robot (MRR) [2, 3, 4]. Modular refers to robots that are constructed by using a limited number of interchangeable standardized modules which can be assembled in different kinematic configurations. Thereby, a wide variety of specialized robots can be built from a set of standard components. Reconfigurable implies that the robot can be disassembled and rearranged to accommodate a different product or task, rather than being replaced. Therefore, an MRR can be used as a tool for designing versatile RMS facilities.

Typically, the set of modules consists of joints, links, and end-effectors. Joint modules are the actuators that provide the degrees of freedom (DOF) of each robot. Each joint module can include embedded actuators, optical encoders, torque sensors, Hall-effect sensors, gear transmission mechanisms, and controllers. Link Modules of varying lengths connect the joints to



(a) (b)

Figure 1.1: WMRR modules: (a) joint module; and (b) link module

each other. The end-effector module consists of the tools required to interact with the robot environment. Fig. 1.1 shows the joint and link modules of the Waterloo Modular and Reconfigurable Robot (WMRR) designed and developed at the University of Waterloo. Fig. 1.2 depicts the connection of the two links and one joint module.

The MRRs, in RMS facilities, allow for a more efficient execution of complex tasks than fixed-configuration manipulators. The advantages of the MRR can be summarized as follows:

- **Fast and cheap conversion of a manufacturing line to produce another product:** An MRR in the manufacturing line can be easily and rapidly reconfigured to perform a new task. Therefore, changes in the product, assembly line, or manufacturing process can be accomplished more conveniently and with less cost.
- **Simplified repair, maintenance, and upgrade:** Since the joint and link modules are standardized, repairs, maintenances and upgrades can be performed by simply replacing a specific module. This translates to lower overall operating costs and faster troubleshooting.
- **Optimal shaping in a greater number of circumstances:** Modularity bestows the MRR with the ability to perform a certain task with a kinematic configuration(KC), which is optimized according to a set of operational performance measures. Instead of using just one robot configuration for all tasks, a robot configuration that is specifically optimized for the task can be realized through the use of MRRs.

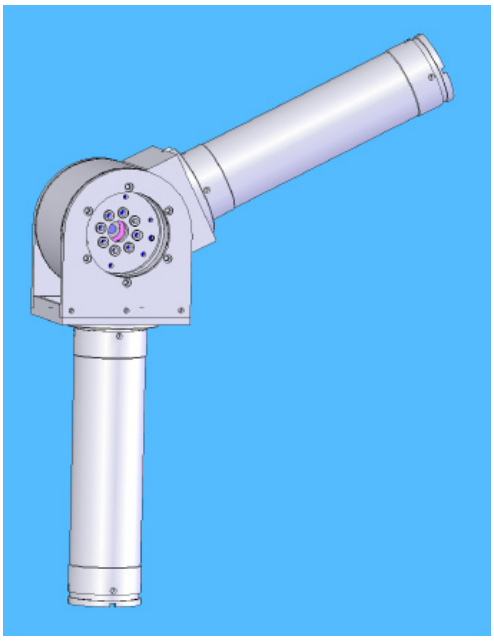


Figure 1.2: Connection of two links to a joint module in WMRR

1.2 Problem Definition

The number of configurations that can be obtained from a set of modules that consists of a joint, capable of generating rotational and pivotal movements, and three types of links, straight, L-shape, and U-shape is expressed as follows

$$x = 2 \times \sum_{k=0}^{n-1} 4^k \times \frac{(n-1)!}{k!(n-1-k)!}\,, \tag{1.1}$$

where x and n are the number of configurations and the number of joints, respectively [5]. According to (1.1) , as many as 6250 configurations can be realized for six serial joint modules. It is evident that by considering more links with different sizes or more joint modules, capable of generating different types of movement, this number increases.

For a specific task, only a subset of all the configurations are capable of reaching all the given set of task points. Moreover, from the set of robot configurations that can actually reach all task points, some perform better than others in terms of meeting a set of specific performance criteria. As a result, for the assessment of the suitability of each KC, numerous constraints and performance criteria for all the desired task points must be examined. This process is demanding in terms of computational power and time. Therefore, development of efficient methods for seeking robotic configurations that are capable of performing a certain task is critical in the utilization of RMSs and their advantages in flexible manufacturing [2, 3].

The large number of possible configurations and the complexities of the KC space necessitates employing highly efficient, intelligent, and automated search methods for determining the most suitable KC in performing a task. Such methods, called Task-Based Configuration Optimization (TBCO), should search, synthesize, and determine an optimum KC which can be assembled from the available MRR modules.

1.3 Objectives

This thesis addresses the development of novel algorithms for the utilization of advanced modular robotic systems. The objectives of this thesis are as follows:

- Developing computationally efficient and fast TBCO algorithms for finding the most suitable MRR configuration for performing a desired task.
- Developing novel algorithms for incorporating multiple solutions of the inverse kinematic (IK) problem into TBCO algorithms. The goal is to identify precise kinematic solutions which can be utilized to efficiently execute a task, while concurrently minimizing a set of performance measures such as shortest collision-free path, minimum required torque, minimum required power, fastest time of performing the task, and the most dexterous path of performing the task.

1.4 Overview of the Approach

In Fig. 1.3, the block diagram of a generic TBCO algorithm is illustrated [6]. The inputs to the algorithm are the module inventory, the optimization criteria, and the task objective. The inputs are encoded before entering the algorithm. The approach to encode these parameters significantly depends on the chosen optimization algorithm.

The first set of inputs, the module inventory, is a set of available joint, link, and end-effector modules. The modules must be modelled and represented mathematically to be used in the TBCO algorithm. In this thesis, a novel representation which includes the joints types, the relative orientation of the assembly of two consecutive joints, and the length of the links, is presented for describing KCs.

The optimization criteria, which consist of the set of considered performance measures are another group of inputs to the algorithm. The adopted optimization criteria are application-dependent.

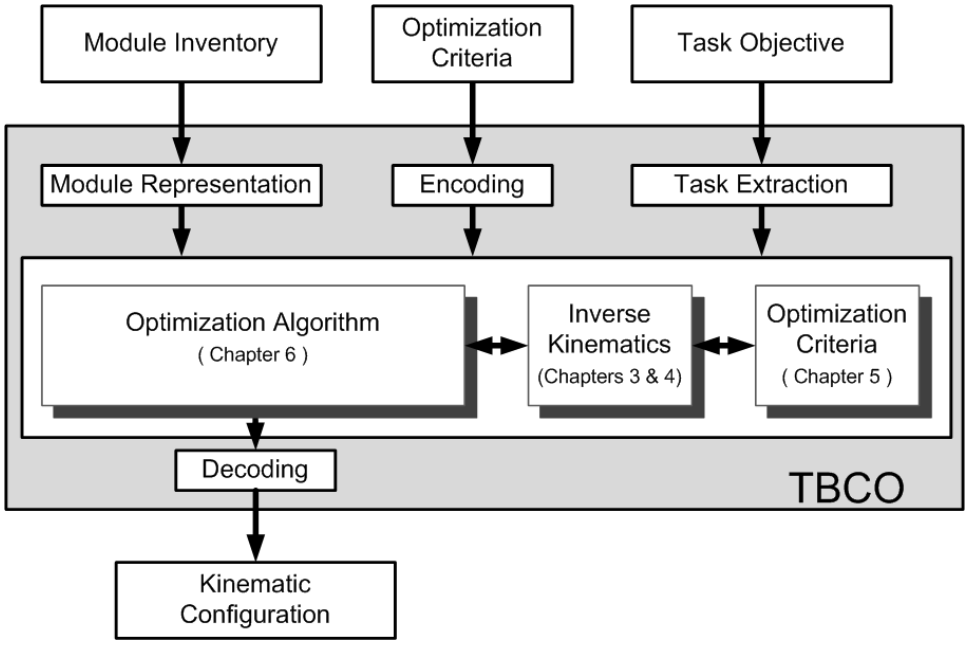


Figure 1.3: Generic block diagram of the Task-based Configuration Optimization Algorithm

The task objective is the last input to the algorithm. The task objective is the operation or the process that the manipulator should perform and is typically based on the product or object which is manipulated by the robot. From the task objective, a task consisting of a set of more dominant task points, is extracted. For example, if the objective is to assemble a car seat, the task of the robot might consist of picking up the bolts and tightening them in a specific place on the product. Thus, the location of the bolts and the holes constitutes the task points. In this thesis, it is assumed that a task consists of a series of task points which are available as the input to a TBCO. The inputs to the TBCO and the corresponding encoding approach are discussed in more detail in Chapter 2.

An optimization algorithm is the core of the TBCO. In this stage, the inputs are analyzed in order to find an optimized KC. In the TBCO literature, Brute-Force, Simulated Annealing (SA), and GAs have been used as optimization algorithms. However, because of the capability of GAs in searching highly nonlinear discrete or continuous spaces, they are the most common optimization methods used in this stage.

The optimization algorithm evaluates the performance of the KCs in executing the desired task according to the considered optimization criteria. It is evident that computing the optimization criteria requires that the IK problem be solved.

Depending on the KC representation, a decoding stage is then needed to convert the output of the algorithm to a KC. The output should be organized such that assembling the manipulator can be achieved easily.

In this thesis, to achieve the objectives the following goals should be established:

- A novel TBCO algorithm which uses the MAs is proposed. MAs are GA and local search hybrids which have been shown in the literature to outperform GAs in terms of the computational speed and the accuracy of the solution reached. The proposed MA based TBCO algorithm is introduced in Chapter 6.
- Novel algorithms for obtaining multiple solutions of the IK problem are proposed. In Chapters 3 and 4, two algorithms are introduced to obtain multiple solutions of the IK problem for MRRs.
- A novel methodology for incorporating multiple solutions of the IK problem in the TBCO algorithm is presented. The proposed methodology enhances the accuracy of the achieved KC solutions. The algorithm is discussed in Chapter 5.

In Fig. 1.3, the research is represented in terms of the chapters of the thesis.

1.5 Contributions

The contributions of this research are grouped into three main category:

1.5.1 Memetic Algorithm Approach for Solving the TBCO Problem

An efficient TBCO algorithm, based on the MAs, is developed. The prominent contributions of the TBCO algorithm are as the following:

- The MA employs the GA to efficiently search the KC space, whereas the numerical local search algorithm seeks the most suitable dimensions of the manipulators. This strategy utilizes the strengths of both methods. To enhance the convergence and global search capabilities of the algorithm, an application-specific elitism and a niching strategy is proposed. It is shown that, compared to the existing GA approaches, the proposed MA increases the efficiency of the TBCO algorithm in terms of convergence speed, and accuracy.
- The length of the links are considered to be adjustable, allowing an optimal length of the links to be determined as a part of the TBCO algorithm. This assumption increases the size and complexity of the search space and necessitates the use of more efficient algorithms than the existing methods.

- A novel numerical local search method for determining the link dimensions of a manipulator is developed. The resultant manipulator is resized such that it can perform a prescribed task. The numerical algorithm constitutes the local search stage of the MA.
- A novel kinematic structure-aware elitism and population restart strategy is proposed to decrease the probability of having the algorithm converge prematurely.

1.5.2 Novel Methods for Obtaining Multiple Solutions of the Inverse Kinematic Problem

In order to incorporate the multiple solutions of the IK problem in the TBCO algorithm, computationally efficient IK solvers, capable of obtaining all the solutions of the problem, are required. Two distinct IK solvers are proposed. The first algorithm utilizes a niching Genetic Algorithm in conjunction with a filtering, a clustering, and a numerical algorithm. The second method is a generalization of numerical techniques for determining the multiple solutions of the IK problem. The advantages of the developed IK solver algorithms are summarized as follows:

- Both algorithms solve for multiple IK solutions of manipulators with rotational type joints.
- Both algorithms are capable of finding multiple IK solutions of a manipulator, even when a closed-form solution does not exist.
- The only requirement of both of the algorithms is the forward kinematic equations of the manipulator. The algorithms do not require knowledge of the number of solutions of the IK problem.

1.5.3 Trajectory Optimization Using the Multiple Solutions of the Inverse Kinematic Problem

The contribution of this part can be listed as follows:

- A novel formulation for considering multiple IK solutions in the trajectory optimization problem is presented. Furthermore, an efficient algorithm for finding the exact solution of the proposed formulation is developed.
- The proposed algorithm can be used for trajectory optimization by using a large number of distinct criteria such as shortest collision-free path, minimum required torque, minimum required power, fastest time of performing the task, or the most dexterous path of performing the task.

- The trajectory optimization algorithm is developed to incorporate the cost of the optimized trajectory as a performance measure in the TBCO.

1.6 Dissertation Organization

The organization of the thesis is described in this section.

Chapter 2 provides the necessary background and the fundamental concepts in this thesis. The existing MRRs in the literature are reviewed and the newly developed Waterloo Modular and Reconfigurable Robot (WMRR), which is designed and implemented as a part of this research project, is introduced. Furthermore, a framework for representing MRRs is devised. The TBCO problem is explored in more detail and a literature review on the existing methods is presented. The IK problem, another focus of this thesis, is reviewed and the existing approaches to solve this problem are summarized. Moreover, the necessity of considering multiple solutions of the IK in the TBCO and the trajectory optimization are discussed. In the final part of the chapter, a brief review of Genetic Algorithms is conducted.

In Chapter 3, a niching GA for solving the IK of serial robotic manipulators is introduced. The algorithm can find multiple solutions of the IK. An enhanced filtering and clustering phase is responsible for identifying and processing the outputs of the proposed GA. For the post-processing stage, a numerical IK solver is used to achieve convergence to the desired accuracy. The algorithm is validated numerically on several distinct KCs.

Chapter 4 introduces another novel solver for obtaining multiple solutions of the IK problem. The proposed algorithm employs a known IK solution to determine the rest of the solutions. The first step of the algorithm is to obtain all the positioning IK solutions. To solve for such solutions, a Joint Reflection Operator(JRO) is proposed. The JRO reflects the joint pairs of the known solutions to new solutions. The joint reflection operator is used iteratively to find all of the positioning IK solutions. Then, the newly acquired positioning solutions are applied to obtain the combined positioning and orienting IK solutions.

A method for utilizing the multiple solutions of the IK in a trajectory optimization is described in Chapter 5. The method has the additional benefit of finding the optimal order of performing the tasks. To include multiple solutions of the IK in the trajectory optimization, a novel formulation is presented which converts the problem into a Generalized Traveling Salesman problem. The resulting problem is solved with a Mixed Integer Programming algorithm to reach the desired solutions. The results are compared with an exhaustive search method to validate the effectiveness and superiority of the proposed algorithm.

In Chapter 6, an MA-based TBCO is proposed. It employs a novel elitism and restart method to prevent premature convergence. Furthermore, a method, for determining the dimensions of

a manipulator to enable it to perform a task, is proposed. The TBCO algorithm, adopts the trajectory optimization algorithm in Chapter 5 to incorporate the multiple solutions of the IK. For solving the IK problem for all the existing solutions, the JRO method, proposed in Chapter 4, is employed. The performance of the algorithm is validated by several numerical analysis.

Chapter 2

Background and Preliminaries

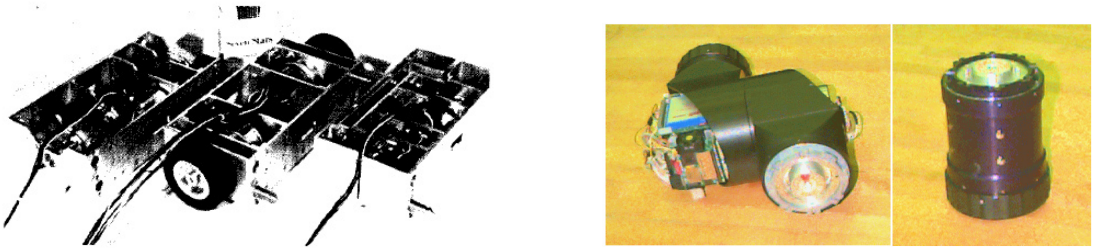
2.1 History of Modular and Reconfigurable Robots

Based on functionality, Modular and Reconfigurable Robots are categorized as: Industrial Modular and Reconfigurable Robots (MRRs), which are the topic of this literature review, and Self Reconfigurable Robots (SRRs).

Several MRRs have been designed, manufactured and presented in the literature. Since the structure and application of link modules are relatively simple, the differences of these robots stem from the joint module design and the controller architecture. The rest of this section is a review of the prominent MRR designs.

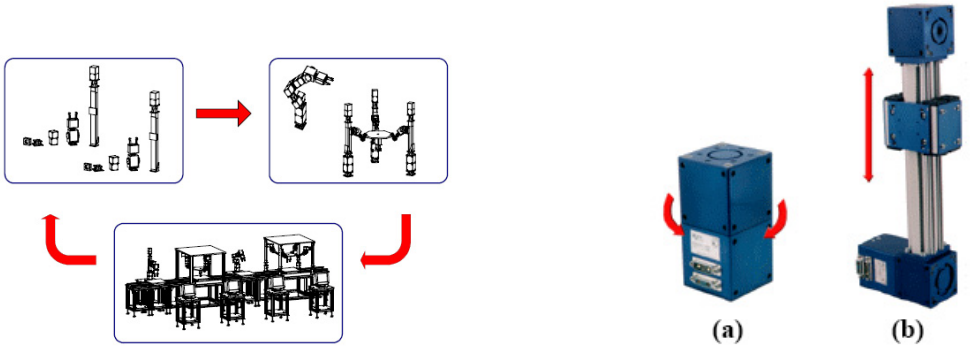
Fukuda proposed a generic architecture and framework applicable to MRRs and SRRs [7]. A centralized controller in addition to the local controllers in each joint has been described. As denoted in Fig. 2.1(a), three different module classes have been manufactured: bending joints, rotational joints, and wheel joints.

Also, design and implementation of pivotal and rotational joints for a Reconfigurable Modular Manipulator System (RMMS) has been reported [8]. The controller architecture is based on a central controller that communicates with the actuator drives of the joint modules through an 8-bit data/address bus and a 4-bit ID bus. The forward kinematics of the robot is obtained by an automatic Denavit-Hartenberg(DH) calculation algorithm. It calculates the DH parameters from a description of the joint and link modules as well as the sequence in which they are connected. A modified numerical method to solve the IK with singularity avoidance is also introduced. Paredis improved the RMMS by allocating a local controller to each of the joint modules [9, 10]. The architecture provides the flexibility of adopting either a centralized or a distributed control scheme. Fig. 2.1(b) reflects a joint and a link of the RMMS.



(a) (b)

Figure 2.1: Joint and link modules: (a) bending, wheel, and rotational joint modules; and, (b) joint and link modules design



(a) (b)

Figure 2.2: Rapidly reconfigurable robotic workcell: (a) workcell in a manufacturing line; and, (b) prismatic and rotary joints

Chen demonstrated a rapidly reconfigurable robotics workcell consisting of several MRRs [11, 12, 13]. Fig. 2.2(a) depicts the application of the workcell in a manufacturing line. The MRR is assembled from a module inventory, consisting of link, rotational, pivotal, and prismatic joint modules. In Fig. 2.2(b), a prismatic and a rotational joint are portrayed. The control architecture is based on a host controller for the coordination between the modules and local controllers of the modules. The communication is provided by a Controller Area Network (CAN) bus.

In another article, an MRR for experimental studies in grasping, manipulation, and force control is explained [14]. The layout of the system includes two manipulators, each with four rotary joints. One type of joint module for producing both rotational and pivotal movements has been constructed.

The Toshiba Modular Manipulator System (TOMMS) can be assembled from one kind of joint module, capable of rotational and pivotal movements, to form a robot with a maximum 3 DOF [15]. A central controller scheme supervises the operation of the joints. The inputs to the controller are derived from a joystick, and the control algorithm depends on the inverse Jacobian

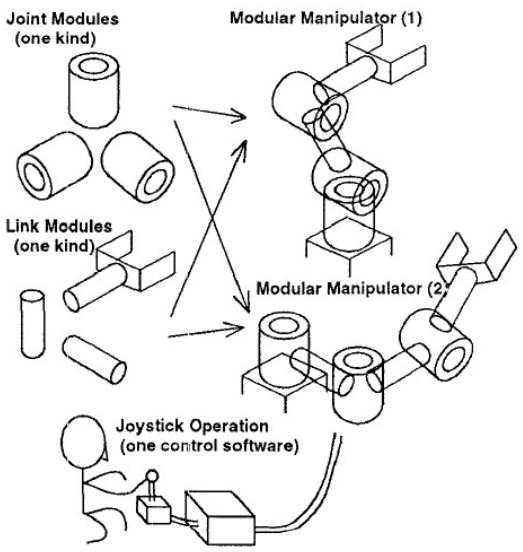


Figure 2.3: Schematic diagram of TOMMS

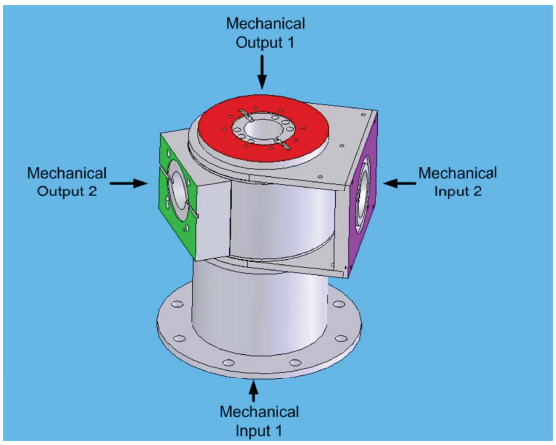
Table 2.1: Joint types achievable by connecting the links to distinct mechanical ports

	Joint type	Input Port	Output Port	Figure
1	Rotational	1	1	Fig. 2.4(b)
2	Pivotal	2	2	Fig. 2.4(c)
3	Perpendicular Rotational	1	2	Fig. 2.4(d)

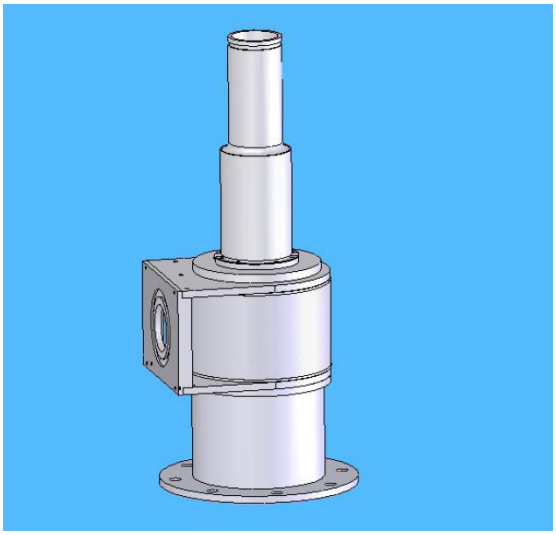
Matrix to find the desired joint variables. The joint modules are designed to be used as rotational or pivotal. Fig. 2.3 is a schematic of the TOMMS.

2.2 Waterloo Modular and Reconfigurable Robot (WMRR)

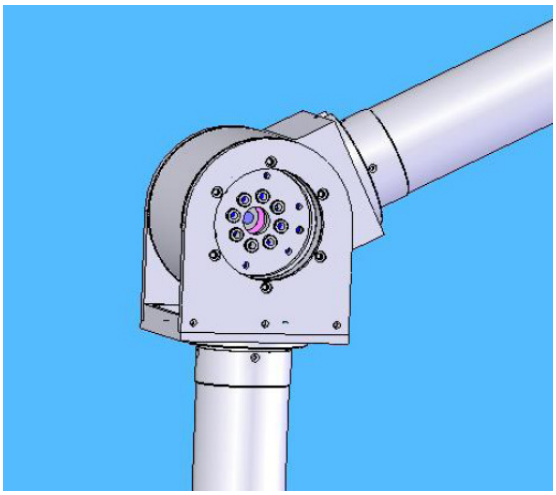
An MRR, called the Waterloo Modular and Reconfigurable Robot (WMRR) has been developed at the University of Waterloo. In Appendix B, the architecture and controller node specifications of the WMRR are explained in detail. Three types of rotary joints with different power production capabilities are designed and implemented [16]. The generic design of the joint modules is shown in Fig. 2.4(a). The links are connected to the joint module by its different mechanical connections, providing distinct types of DOF. Table 2.1 lists the combinations of the connections to the ports and the resulting joint types. Figs. 2.4(b), 2.4(c), and 2.4(d) denote the distinct achievable joint types. Fig. 2.5 reflects the WMRR, when it is assembled in a 3DOF PUMA arm.



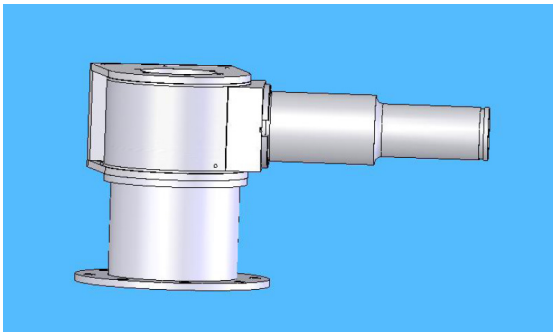
(a)



(b)

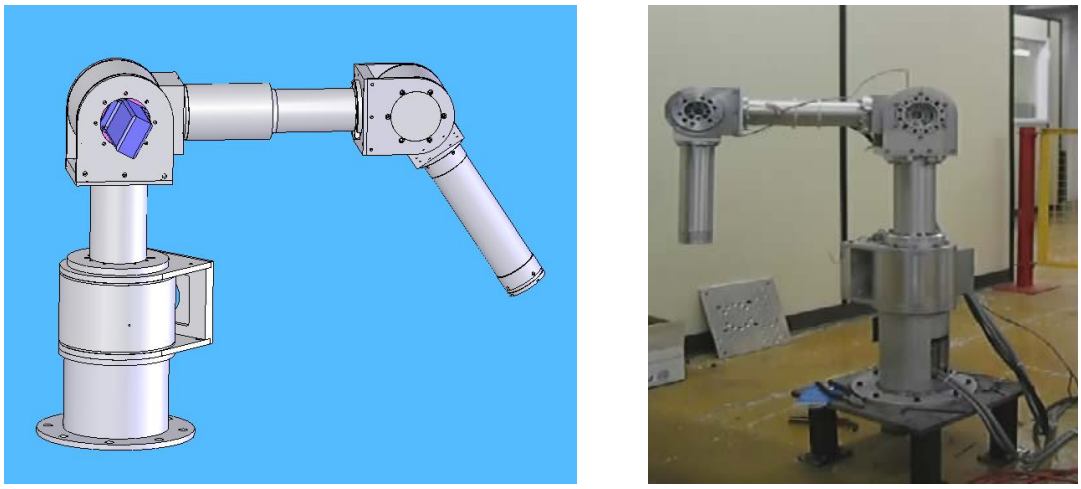


(c)



(d)

Figure 2.4: WMRR Mechanical design principles: (a) mechanical input and output ports on the joint modules; (b) joint in the rotational configuration: mechanical input 1, mechanical output 1; (c) joint in the pivotal configuration: mechanical input 2, mechanical output 2; and, (d) joint in the perpendicular rotational configuration: mechanical input 1, mechanical output 2



(a) (b)

Figure 2.5: WMRR assembled in a 3DOF PUMA configuration

2.3 MRR Representation

The WMRR joint modules can be assembled to provide three types of DOF. On this foundation, it is possible to define three distinct classes of joint modules: Rotational, Pivotal, and Perpendicular-rotational as depicted in Fig. 2.6. The set of three joint classes is called the standard modular joint set. It can be shown that a wide range of existing industrial manipulators can be represented by a KC, consisting only of the standard modular joints. Also, an n-DOF manipulator, which only includes standard modular joints is represented by $\mathbf{m}_n$.

To represent the joint modules kinematically, all the joint classes are considered as a generic joint module with an input and an output port, called the mechanical ports. They are attached to the top and bottom of the generic joint module as shown in Fig. 2.6(a). An input mechanical port should be connected to the link closer to the base of the robot, whereas the output port is connected to the link closer to the end-effector. By attaching a coordinate frame to the input and output port, each joint module can kinematically be modelled by the relative location and orientation of the input and output port frames. The input and output frames of the modular joint set are illustrated in Fig.2.6.

A manipulator $\mathbf{m}_n$ can be fully described by the joint types, the relative assembly orientation of two consecutive joints, and the length of the links. These characteristics are conveniently expressed in a $(n+1) \times 3$ array as

Table 2.2: Permissible values for the variables of the manipulator representation matrix

	Name	Permissible value	Description
1	ϕ_i	0	0 Radians
		$\pi/2$	$\pi/2$ Radians
2	m_i	P	Pivotal joint
		R	Rotational Joint
		PR	Perpendicular-rotational joint
3	l_i	$l_i \in [l_{min},\ l_{max}]$	Length of the link (cm)

$$\mathbf{m}_n = \begin{bmatrix} 0 & 0 & l_0 \\ \phi_1 & m_1 & l_1 \\ \phi_2 & m_2 & l_2 \\ & \vdots & \\ \phi_n & m_3 & l_n \end{bmatrix}. \tag{2.1}$$

In this matrix, the elements of the first column, ϕ_i, represent the orientation of joint i relative to joint $i-1$. This orientation is determined by the angle between the x_{out} axis of joint $i-1$ and x_{in} axis of joint i. In the second column, the joint types m_i are stored. The third column represents the length of the links, l_i. Although l_i can be a continuous variable (for telescopic links and custom made links) or a discrete variable (for fixed size modular links), in this thesis it is assumed that the link lengths are continuous variables such that $l_i \in [l_{min},\ l_{max}]$. The permissible values for each variable are shown in Table 2.2.

The first row of the matrix represents the first link of the robot. This link is perpendicular to the ground and can connect the first joint to the base of the robot. The last element of the matrix, at row $n+1$ and column 3, corresponds to the length of the last link which can be a tool.

For instance, the matrix representation of the 2-DOF manipulator in Fig.2.7 is

$$\mathbf{m}_2^{sample} = \begin{bmatrix} 0 & 0 & L_0 \\ 0 & P & L_1 \\ \pi/2 & P & L_2 \end{bmatrix}. \tag{2.2}$$

2.4 Definitions

To establish the preliminaries of this research, the following concepts are defined:

Task Point $\mathbf{T}_{des}$: A Task point is a desired position and orientation in the Cartesian space which the robot should reach. Task points are defined with respect to a reference frame which

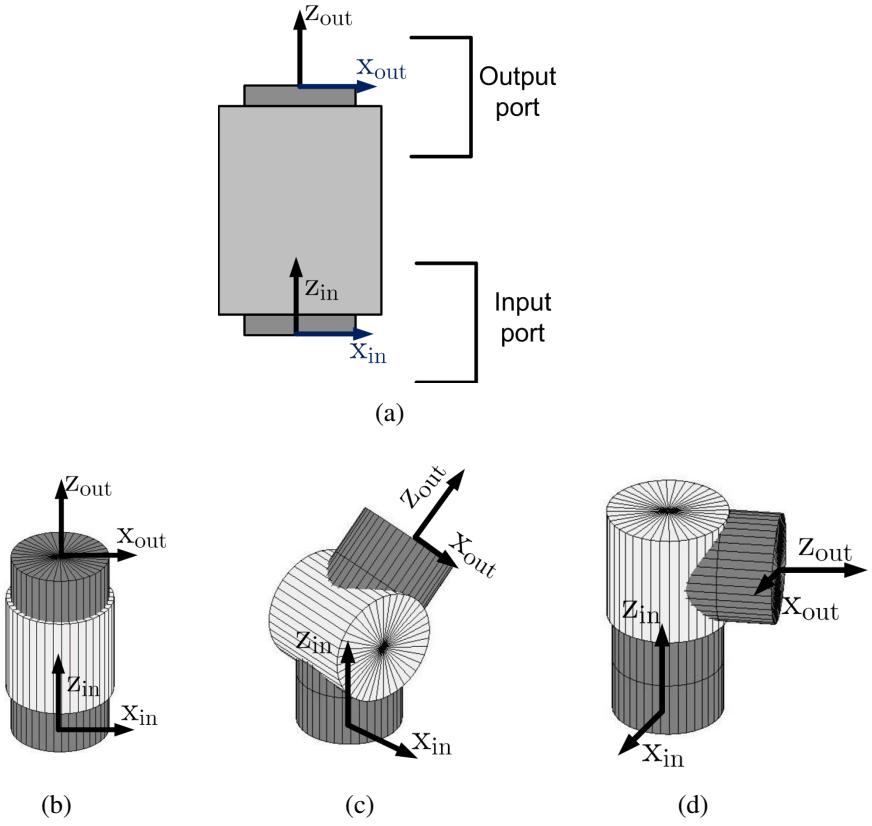


Figure 2.6: Standard modular joint set and the corresponding mechanical input and output frames: (a) box representation of joint modules; (b) rotational joint; (c) pivotal joint; and, (d) perpendicular-rotational joint

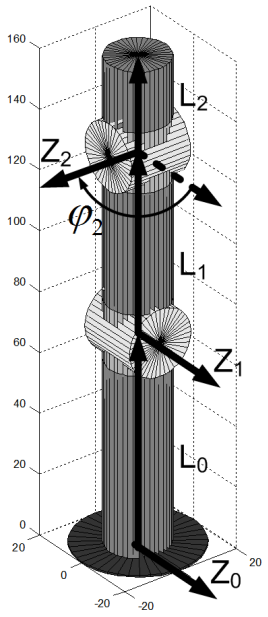


Figure 2.7: Manipulator described by the matrix representation of $\mathbf{m}_2^{sample}$

is usually positioned at the base of the manipulator. A task point can be a location on the manipulated object, a point in space that the manipulator should reach, or a point that the robot should pass in order to avoid an obstacle. In this work, the position and orientation of the i-th task point, with respect to the reference frame, is represented by the Homogenous Transformation, T_{des}^i as follows

$$T_{des}^i = \left(\begin{array}{ccc|c} & R_{des}^i & & P_{des}^i \\ \hline 0 & 0 & 0 & 1 \end{array}\right), \tag{2.3}$$

where R_{des}^i and P_{des}^i represent a 3×3 rotation matrix representing the desired orientation, and a 3×1 translation matrix representing the desired position of the i-th task point in the Cartesian space, respectively.

End-Effector's Position and Orientation $\mathbf{T}_{ee}$: The position and orientation of the end-effector of an n-DOF manipulator is a function of its joint variables. The position and orientation of the end-effector, with respect to the reference frame, is represented by a Homogenous Transformation (T_{ee}) as a function of the joint variables $[q_1\ q_2 \cdots q_n]$. For a $\mathbf{m}_n$ manipulator the joint variables are all angles and represented by $\Theta = [\theta_1\ \theta_2 \cdots \theta_n]$, as shown in

$$\mathbf{T}_{ee}^{\mathbf{m}_n}(\Theta) = \left(\begin{array}{ccc|c} & \mathbf{R}_{ee}(\Theta) & & \mathbf{P}_{ee}(\Theta) \\ \hline 0 & 0 & 0 & 1 \end{array}\right) . \tag{2.4}$$

R_{ee} and P_{ee} are the rotation and translation matrices representing the orientation and position of the end-effector in the Cartesian space, respectively.

Task $\mathbb{T}_t$: A Task is a set of the task points which should be reached by the manipulator to accomplish an objective. For example, all the task points which a robot should reach in order to assemble a car seat, form a task. In this research, a Task with t task points is represented by $\mathbb{T}_t$, or simply $\mathbb{T}$, and is defined as

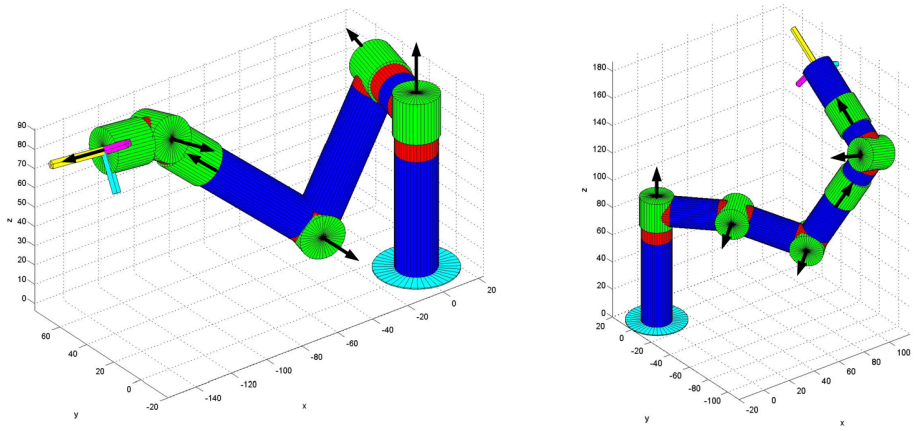
$$\mathbb{T}_t = \{\ \mathbf{T}_{des}^1,\ \mathbf{T}_{des}^2,\ \cdots\ \mathbf{T}_{des}^t\ \}\,, \tag{2.5}$$

where $\mathbf{T}_{des}^i$ represents the i-th task point of the task $\mathbb{T}$.

Kinematic Configuration: Kinematic Configuration(KC) refers to the chained arrangement of the joints in a robot. It is defined by the number of DOFs, the joint types, dimension of the links, and the angle between the rotation axes of two consecutive joints. KCs are categorized as: serial, parallel, and hybrid. Serial manipulators with structures similar to the WMRR are the most common robots in industry, and are the focus in this thesis.

In the most general case, the configuration is defined by the Denavit-Hartenberg parameters. For the MRR, the configuration can also be described as the sequence of the assembled modules and their relative orientation. As mentioned in Section 2.3, an n-DOF KC, assembled only from the standard modular joints, is represented by $\mathbf{m}_n$. In Fig .2.8, two distinct $\mathbf{m}_6$ KCs and the corresponding axes of the joints are illustrated.

n-DOF Kinematic Configuration Space: The space that includes all the distinct KCs which can be assembled by using n joint modules of the standard modular joints is called an n-DOF KC Space, that is, each member of such a space corresponds to an $\mathbf{m}_n$. n has a direct impact on the size, dimension, and characteristics of this space. For MRRs with a set of fixed links of different sizes, this space is a discrete space. If the length of the links are considered continuous, this space is a hybrid continuous-discrete space (continuous with respect to the link lengths and discrete with respect to the joint types and relative orientation of the joints). In this thesis, an n-DOF KC space is represented by $\mathbb{M}_n$ such that



(a) (b)

Figure 2.8: Two members of the KC space $\mathbb{M}_6$: (a) 6-DOF PUMA type configuration; and, (b) sample 6-DOF configuration

$$\mathbb{M}_n = \bigcup_{i=1,2,\cdots} \mathbf{m}_n^i \ . \tag{2.6}$$

Manipulator Workspace: The workspace of a manipulator is the region in the Cartesian space which can be reached by the manipulator end-effector. The dexterous primary workspace is the region which the manipulator can reach with any orientation of the end-effector. The secondary workspace is the volume, reachable by the manipulator from a limited number of orientations [17]. In this thesis, the secondary workspace is simply called the manipulator workspace, W_T, and is defined as

$$W_T\left(\mathbf{m}_n\right) = \left\{ \mathbf{T}_{ee}^{\mathbf{m}_n}\left(\Theta\right) \mid \forall \Theta \ : \ \Theta_{max} \geq \Theta \geq \Theta_{min} \right\} \ . \tag{2.7}$$

Here, the workspace of a manipulator includes all the position and orientations of the end-effector which can be reached with joint angles within the feasible range.

2.5 Task-Based Configuration Optimization

2.5.1 Analysis and Mathematical Representation

The goal of a TBCO is to find a KC that is capable of performing a certain task $\mathbb{T}$ more efficiently than other KCs. Therefore, the capability of performing $\mathbb{T}$ is a necessary condition for a manipulator to be the solution of the TBCO.

The TBCO determines an $\mathbf{m}_n$, which is capable of performing the desired task $\mathbb{T}$, while simultaneously, minimizing the efficiency measure f_{obj}. Thus, the TBCO is mathematically formulated as

$$\begin{cases} \arg \min\limits_{\mathbf{m}_n \in \mathbb{M}_n} f_{obj}(\mathbf{m}_n, \mathbb{T}_t) \\ \\ subject\ to: \ \mathbb{T} \subset W_T(\mathbf{m}_n) \end{cases} . \tag{2.8}$$

It is assumed that f_{obj} is derived such that lower values are preferable, hence converting the optimization problem into a minimization. The constraint in (2.8) states that the desired task should be a subset of the reachable workspace of any solution of the TBCO. Determining the workspace of all the $\mathbf{m}_n$ manipulators is a computationally expensive process. Therefore, the TBCO is expressed by using the following alternative formulation

$$\begin{cases} \arg \min\limits_{\mathbf{m}_n \in \mathbb{M}_n} f_{obj}(\mathbf{m}_n, \mathbb{T}_t) \\ \\ subject\ to: \ \{ \exists\, \Theta_i^s \text{ for } \mathbf{m}_n \quad | \quad \mathbf{T}_{ee}^{\mathbf{m}_n}(\Theta_i^s) = \mathbf{T}_{des}^i \quad \text{, for } \forall i = 1..t \quad \} \end{cases} . \tag{2.9}$$

The difference between (2.8) and (2.9) is that, according to the former, the workspace of each $\mathbf{m}_n$, $W_T(\mathbf{m}_n)$, should initially be determined, whereas in the latter, each $\mathbf{m}_n$ is examined for performing all the task points. In this thesis, (2.9) is utilized as the approach.

In essence, the idea of a TBCO is to find an $\mathbf{m}_n$, which can perform the desired task, $\mathbb{T}$, with an f_{obj} lower than all the other members of $\mathbb{M}_n$. Since $\mathbb{M}_n$ is represented by the MRR representation in Section 2.3, the TBCO can be viewed as determining all the elements of the matrix representation of an $\mathbf{m}_n$ to satisfy the constraints while minimizing the objective function.

The TBCO problem can be considered as a mapping from the Cartesian task space, a space that consists of positions and orientations of task points, to $\mathbb{M}_n$. As in Fig. 2.9(a), depending on $\mathbb{T}$ and n, a mapping could exist which maps every task point in the Cartesian space to KCs capable of reaching the task point. Generally, this mapping has more than one solution for any task point in the Cartesian space, i.e., more than one KC is capable of reaching a certain task.

Three possible cases can occur if the mapping is applied to the set of task points in a task:

- **No common point in the configuration space exists:** Here, no single KC exists that can reach all the task points. Fig. 2.9(b) shows an instance of this case. To solve the TBCO, n should be increased. Since the configurations with higher DOFs are capable

of performing more complex tasks, the chance of finding a KC for such a task increases. With the increase in n, the dimension of the search space also increases and a larger set of KCs becomes realizable.

- **One common point in the configuration space exists:** This case is shown in Fig. 2.9(c). Since only one KC is capable of reaching all the task points, the TBCO is irrelevant, and the single KC capable of executing the task is the solution of the TBCO problem.
- **More than one common point in the configuration space exist:** Among all the configurations for achieving the task, the one which performs most efficiently, according to the set of considered optimization criteria, is the solution of the TBCO. Fig. 2.9(d) illustrates such a case.

By obtaining the mapping from the Cartesian space to the KC space, the most suitable KC for a task is found. However, this mapping is highly nonlinear with a large number of mixed discrete and continuous variables. Furthermore, the mapping varies as the number of DOFs, n, changes.

Therefore, the general approach to solving the TBCO is not to find the mapping itself, but to conduct an extensive search of $\mathbb{M}_n$ for KCs capable of performing the task. Among those configurations, the one that can minimize the considered optimization criteria becomes the solution.

2.5.2 Design Variables

In the TBCO, the goal is to find the optimum KC to achieve a certain task, given a set of optimization criteria. KC of an MRR consists of the number of DOFs of the robot, the length of the links, and the type and relative orientation of the joints, as defined in the matrix representation of MRRs. In this section, the design variables in the TBCO problem are discussed in more detail.

Degrees-of-Freedom(DOF) Since most of the industrial robotic joints have only 1-DOF, the DOF of a robot is equal to the number of joints. The DOF of a manipulator has significant effect on the workspace. For instance, for pick and place applications in a 2D plane, a 4DOF SCARA robot is sufficient, whereas for a general-purpose spatial task in the 3D space a PUMA robot with 6-DOF is required. Under normal circumstances, robots with fewer DOFs are preferred, since they offer less complexity in control, less mass and less power consumption. The DOF of a robot is the first design parameter that should be determined in the TBCO.

To reduce the complexity of the search in $\mathbb{M}_n$ for all n, the size of the search space is reduced by executing the search in $\mathbb{M}_n$ with a fixed n. When n is changed iteratively for all the viable values of n, the best overall KC in $\mathbb{M}_n$ for all n is attained.

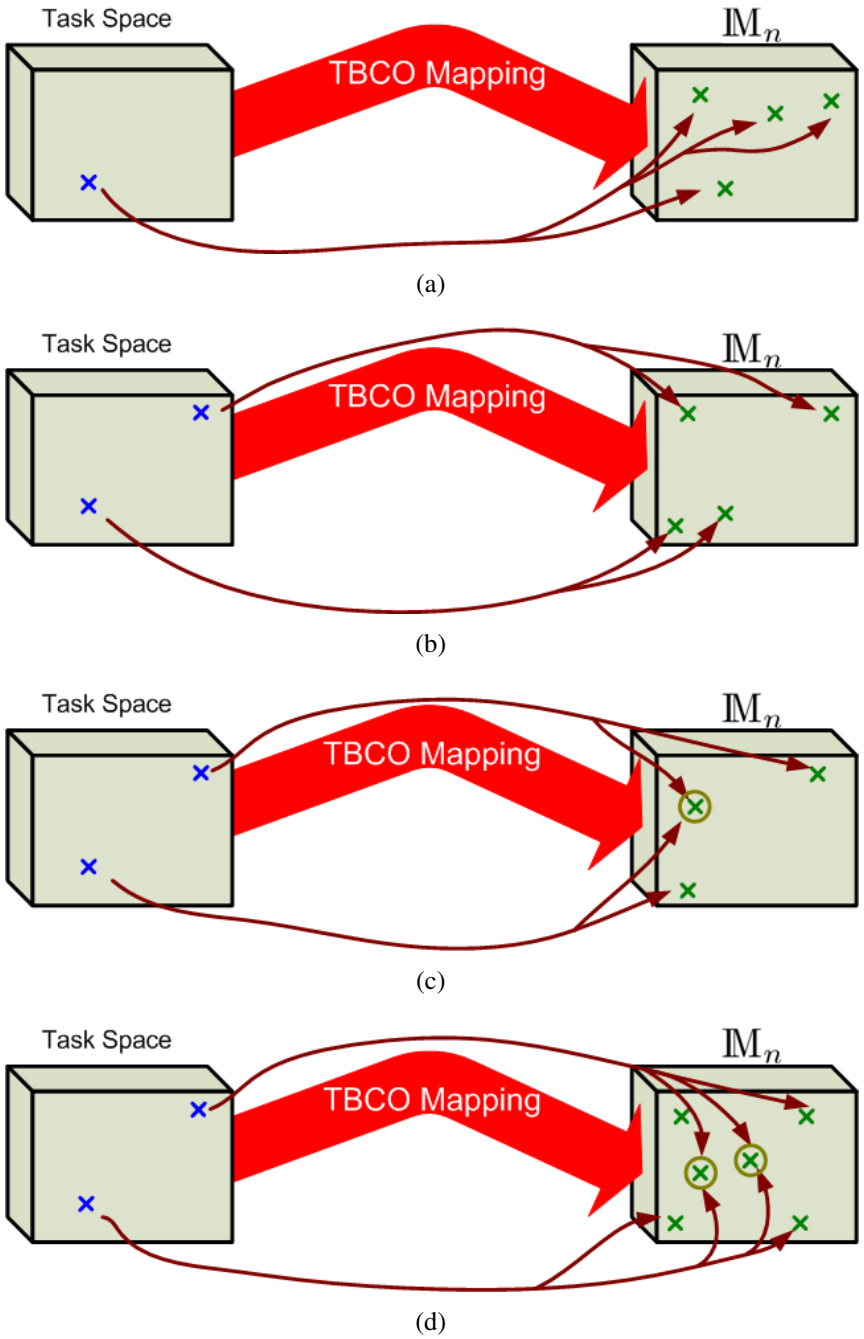


Figure 2.9: Mapping from the cartesian task space to the kinematic configuration space $\mathbb{M}_n$: (a) mapping for one task point; (b) no common solutions in $\mathbb{M}_n$; (c) one common solution in $\mathbb{M}_n$; and, (d) numerous common solutions in $\mathbb{M}_n$

Type and relative orientation of the joints Typically, joints of robotic manipulators are grouped into prismatic, spherical, cylindrical, and revolute joints. The standard modular joints of the WMRR consists of only revolute joints.

In the TBCO, the type of all the joints should be obtained. Furthermore, most MRR joints can be connected to links in different orientations, producing distinct types of DOF. These orientations should also be obtained in the TBCO.

Length of the links The length of the links is another set of design parameters that should be obtained by the TBCO. In this research, only rigid links are considered. In MRR, the link length is considered as a discrete or continuous variable. When discrete link lengths are considered, a set of links with different lengths are designed to provide more flexibility in terms of the reach of the robot. However, since the links are the least complicated parts of an MRR, they can easily be manufactured specifically for each application. Therefore the length of the links can also be considered as continuous variables. In addition, telescopic links are a type of link usable in MRRs which provide a continuously variable link length within a minimum and a maximum limit.

2.5.3 Optimization Constraints

As mentioned in Section.2.5, the TBCO can be converted to a constrained minimization problem. The constraint states that all the solutions of the TBCO should be capable of performing the desired task. In other words, every solution should reach all the task points of the desired task. To formulate the constraints, a mathematical measure for determining whether a KC can reach a task point is required. For a manipulator to reach a task point, it must put the end-effector at the position of the desired task, while assuming the same orientation as that of the desired task point. Moreover, the manipulator should reach the task points with a physically feasible pose. The optimization constraints of the TBCO are summarized as follows:

Position Reachability Error ($f^p_{rch,i}$): The position reachability error is a measure of how close a KC can position its end-effector from the spatial coordinates of the task point. When $\mathbf{T}_{ee}$ is as close to the i-th task point as the manipulator permits, the distance between the positions of $\mathbf{T}^i_{des}$ and $\mathbf{T}_{ee}$ is the position reachability error. The smaller the $f^p_{rch,i}$, the closer a robot can position the end-effector to the task point. When a KC can reach a task point, the position reachability error is zero $f^p_{rch,i} = 0$. $f^p_{rch,i}$ for the i-th task point, mathematically represented as

$$f^p_{rch,i} = \min_{\Theta} \|\mathbf{P}^i_{des} - \mathbf{P}_{ee}(\Theta)\| \,, \tag{2.10}$$

where $\mathbf{P}^i_{des}$ and $\mathbf{P}_{ee}$ correspond to the position of the task and the position of the end-effector, respectively. Θ is the joint angle in which the end-effector is positioned as close as possible to the desired task.

Orientation Reachability Error ($f^o_{rch,i}$): In the TBCO, a KC is desired which can assume the same orientation as that of the task point. Since, by itself, the orientation reachability error does not convey a measure of how well a manipulator can perform a task, the orientation reachability error does not have a substance independent of the position reachability error.

Euler angles (α, β, γ) which give the same orientation as $(\alpha+\pi, -\beta, \gamma-\pi)$, are commonly reported in the literature to represent the orientations of the end-effector and the task point. Therefore, $f^o_{rch,i}$ is expressed as

$$f^o_{rch,i} = \min(\|(\alpha^i_{des}, \beta^i_{des}, \gamma^i_{des}) - (\alpha_e, \beta_e, \gamma_e)\|, \|(\alpha^i_{des}, \beta^i_{des}, \gamma^i_{des}) - (\alpha_e + \pi, -\beta_e, \gamma_e - \pi)\|), \tag{2.11}$$

where $(\alpha_e, \beta_e, \gamma_e)$ and $(\alpha^i_{des}, \beta^i_{des}, \gamma^i_{des})$ represent the Euler angles of $\mathbf{R}_{ee}(\Theta)$ and $\mathbf{R}^i_{des}$, respectively. $\mathbf{R}_{ee}(\Theta)$ is the rotation matrix of the Homogenous Transformation of the end-effector, when it is as close as possible to the desired task point in terms of the orientation.

In this thesis, the quaternions, which provide a more convenient method for mathematically expressing the orientation reachability error, are applied. Quarternions are explained in more detail in Section.2.6.3.

Total Reachability Error ($f_{rch,\mathbb{T}}$): The total reachability error for task $\mathbb{T}$ is the weighted sum of the position and orientation reachability errors for all of the desired task points and is written as

$$f_{rch,\mathbb{T}} = \sum_{i=1\cdots t} \left(w^p_i . f^p_{rch,i} + w^o_i . f^o_{rch,i} \right) . \tag{2.12}$$

To guarantee that both the orientation and the positioning reachability errors contribute equally to the total reachability error for each task, w^p_i and w^o_i are written as

$$\begin{cases} w^p_i &= 1 \\ w^o_i &= \frac{\|\mathbf{P}^i_{des}\|}{\pi\sqrt{3}} \end{cases} . \tag{2.13}$$

w^p_i and w^o_i are calculated by scaling the orientation reachability such that the maximum possible orientation reachability error, which occurs in $\alpha = \pi$, is large enough to compete

Table 2.3: Common constraints in TBCO

	Constraint Parameter	Symbol	Mathematical Expression
1	Position Reachability Error	$f^p_{rch,t}$	$\min_\Theta \|P^t_{des} - P_{ee}(\Theta)\|$
2	Orientation Reachability Error	$f^o_{rch,t}$	$\min\left(\|(\alpha_t,\beta_t,\gamma_t) - (\alpha_e,\beta_e,\gamma_e)\|,\right.$ $\left.\|(\alpha_t,\beta_t,\gamma_t) - (\alpha_e+\pi,-\beta_e,\gamma_e-\pi)\|\right)$
3	Total Reachability Error	$f_{rch,\mathbb{T}}$	$\sum_{i=1\cdots t}\left(w^p_i.f^p_{rch,i} + w^o_i.f^o_{rch,i}\right)$
4	Total Joint Limit	$f^i_{JL,t}$	$\sum_{i=1}^{t} f_{JL,i}$

with the positioning reachability error which is a function of the distance of the task points from the base of the robot ($\|\mathbf{P}^i_{des}\|$).

Joint Limit ($f_{JL,i}$): Most of the robotic joints have a mechanical limitation on the joint angles. A KC should be able to reach the task points without violating these limits. The i-th task point joint limit constraint, is given by $f_{JL,i}$ and can be expressed as

$$f_{JL,i} = \frac{1}{2}\sum_{j=1}^{n}\left(\frac{\theta_{i,j} - \theta_{j,mid}}{\theta_{j,max} - \theta_{j,min}}\right)^2 , \tag{2.14}$$

where $\theta_{i,j}$ is the angle of the j-th joint, where the manipulator has reached the i-th task point [18]. $\theta_{j,min}$, $\theta_{j,max}$, and $\theta_{j,mid}$ are the minimum, maximum, and the midpoint of the permissible range, respectively. Small values of $f_{JL,i}$, which correspond to the joint angles far from the limits, are desired. The total joint limit constraint of a manipulator for task $\mathbb{T}$ is represented by $f_{JL,\mathbb{T}}$ and can be expressed as

$$f_{JL,\mathbb{T}} = \sum_{i=1}^{t} f_{JL,i}\ . \tag{2.15}$$

Table 2.3 summarizes the TBCO constraints and the corresponding mathematical formulation.

With the TBCO constraints defined, the mathematical formulation of (2.9) can be written as

$$\begin{cases} \arg\min\limits_{\mathbf{m}_n \in \mathbb{M}_n} f_{obj}\left(\mathbf{m}_n, \mathbb{T}_t\right) \\ \\ subject\ to: \begin{cases} f_{rch,\mathbb{T}}\left(\mathbf{m}_n, \mathbb{T}_t\right) &= 0 \\ f_{JL,\mathbb{T}}\left(\mathbf{m}_n, \mathbb{T}_t\right) &= 0 \end{cases} \end{cases} . \tag{2.16}$$

2.5.4 Optimization Criteria

As discussed in Section 2.5.1, for a KC to be selected as a TBCO solution, it should first satisfy the constraints, and then prove to be the best according to a set of optimization criteria. In the following, the more common optimization criteria are explained.

Dexterity Measures (W, M_r): Numerous dexterity measures have been introduced in the literature [19, 20, 21, 22, 23]. Most of these measures quantify the performance of the robot at the end-effector. Typically, these measures are functions of the Jacobian of the robot. The most common dexterity measure is the Measure of Manipulability (simply Manipulability) which is defined as

$$W = \sqrt{\det(JJ^T)}\,. \tag{2.17}$$

Because of the dependency on the Jacobian matrix, the dexterity measures also vary with the dimension of the robot. However, in the TBCO a dexterity measure independent of the size of the robot is required. Relative manipulability is a dexterity measure that is independent of the dimensions of the robot [24]. Relative manipulability is defined as

$$\begin{cases} M_r &= \dfrac{\sqrt[m]{\det(JJ^T)}}{f_M} \\ f_M &= \left(\sum_{i=1}^{n} \sqrt{a_i^2 + d_i^2}\right)^2 \end{cases}, \tag{2.18}$$

where m and n are the number of rows in the Jacobian matrix (the dimension of the task space) and the number of joints, respectively. a_i and d_i are the link length and joint offset of the i-th joint. Larger values of the relative manipulability are preferred.

Robot Dimension (f^a_{DIM}): Usually, it is desirable to have the smallest possible manipulator. So, a robot with smaller link sizes, rather than larger robots are preferred. This optimization criterion is formulated as

$$f^a_{DIM} = \sqrt{\sum_{i=1}^{n} l_i}\,. \tag{2.19}$$

where l_i is the length of the i-th link.

Links Relative Dimension (f^r_{DIM}): According to this parameter, the robots that have smaller links closer to the end-effector are preferred. The length of each link is compared to the

length of the previous link, and if the link is smaller, it receives a bonus. f^r_{DIM} can be expressed as

$$f^r_{DIM} \;=\; \sum_{i=2}^{n+1} \max\left(\, 0 \;,\; l_i - l_{i-1} \,\right) \;. \tag{2.20}$$

Degree-of-Freedom (f_{DOF}): With the addition of each joint to a robot, the complexities in control and modeling increases. In addition, more joints can be translated to more total mass and power consumption. Hence, the tendency is to choose a robot with fewer number of joints and DOF [24, 25].

Power Consumption: Since, in most applications, the manipulator performs a certain task numerous times, a manipulator, which consumes the minimum possible power to perform the task, is beneficial. In Chapter 5 this criteria is discussed.

Task Completion Time: In a wide range of applications, the time for performing a certain task is crucial. Therefore, the required time to finish a task can be considered as an optimization criterion.

Obstacle Avoidance: In some applications, the robot should be able to reach a task point without violating the obstacles in the workspace. The obstacles can be parts of the machinery in the workspace or parts of the product in the manufacturing line. This criteria can also be categorized as constraint parameters. In the literature, different approaches to represent this criterion are introduced. For instance, Paredis defines a penalty function, to be minimized [26]. When a robot intrudes the space of an obstacle, this penalty function increases proportional to the depth to which the robot is inside the obstacle.

Self Collision: When a manipulator is performing a task, it should be verified that the robot links and joints are not colliding with each other. This criteria can be represented as another type of obstacle avoidance with the position of the obstacles varying with respect to the KC and joint angles of the manipulator. This parameter is specially important in redundant robots, where the high dexterity of the robot can cause collisions.

Table 2.4 is a summary of the optimization criteria.

2.5.5 Tasked-Based Configuration Optimization: A Literature Review

In this section, the existing TBCO approaches in the literature are reviewed.

Table 2.4: Common optimization parameters in TBCO

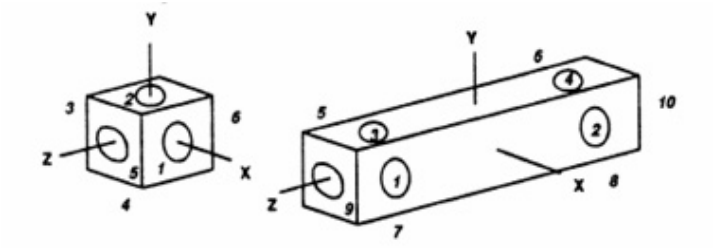
	Optimization Parameter	Symbol	Mathematical Expression
1	Manipulability	W	$\sqrt{\det(JJ^T)}$
2	Relative Manipulability	M_r	$\frac{\sqrt[m]{\det(JJ^T)}}{f_M}$
3	Robot Dimension	f^a_{DIM}	$\sqrt{\sum_i (a_i^2 + d_i^2)}$
4	Links Relative Dimension	f^r_{DIM}	$\sum_{i=2}^{n}(l_i - l_{i-1})$
5	DOF Number	f_{DOF}	n
6	Power Consumption	——	——
7	Task Completion Time	——	——
8	Obstacle Avoidance	——	——
9	Self Collision	——	——

The works in this field can be categorized into two approaches. I-Ming Chen et al. have modelled a modular robot as a sequence of modules with distinct characteristics, and Khosla et al. have represented the manipulators with the Denavit-Hartenberg parameters.

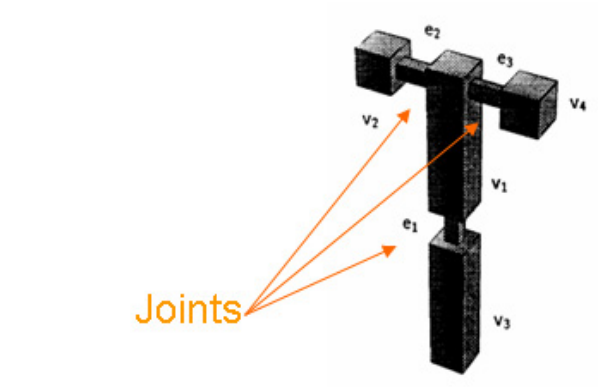
Chen has introduced a representation for MRRs, where the links of a modular robot are considered as squared prisms or cubic boxes with ports on each side [27, 28, 29]. These ports are used to connect two links to each other through a joint. In Fig. 2.10(a), the links and their corresponding ports (the numerated circles) are illustrated. The joints are considered as connectors for attaching the different ports of two neighbouring links as in Fig. 2.10(b). A kinematic graph is chosen to express the configuration of a robot, and from the graph an *Assembly Incident Matrix* (AIM) is extracted. In these works, the AIM is translated to a string and used in a GA, with the reachability error and manipulability as the optimization criteria. An IK solver, based on Products-of-Exponentials (POE) for the proposed AIM representation is introduced [30, 31, 32]. POE of AIM representation is also used to implement a closed-form IK solver [33].

The same approach is expanded to modify the AIM to incorporate the port vectors [25, 34]. The mutation and crossover operators are also modified such that they can directly be applied to the AIMs. In these works, the reachability error, joint limits, manipulability, mechanical constructibility, and the minimum DOF are considered as the objective functions.

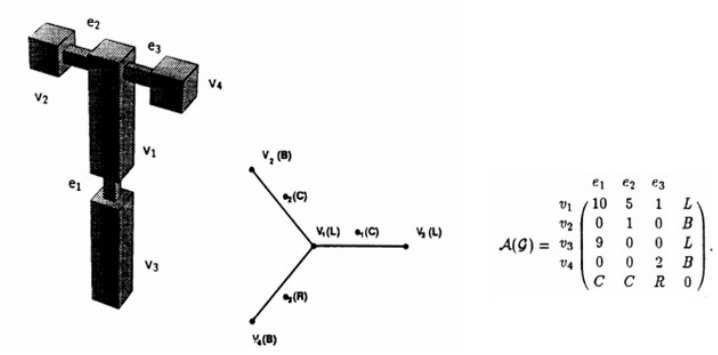
Ramachandran has utilized a distributed agent-base method to use the idle time of the computers connected to a network to speed up the TBCO [35]. Simulated Annealing is the chosen optimization algorithm.



(a)



(b)



(c)

Figure 2.10: Robot representation in Chen's work: (a) link modules; (b) joints; and, (c) kinematic graph and AIM matrix of a robot

Leger's approach can be categorized in Chen's group [36, 37, 38]. The links and joints are all considered to be modular with specific shapes and dimensions. An object-orientated structure to represent a robot in C++ is defined. The algorithm can search in $\mathbb{M}_n$ for all n, where the dynamic behaviour of the robot is also considered. In some cases, the algorithm relies on a human expert to redirect the search. The algorithm is run on a network of 10-30 computers for 9-15 hours to produce the desired results.

In Khosla's approach, the manipulators are modelled by the DH parameters. Paredis and Khosla have obtained the mapping from the task space to the configuration space for a 2-DOF planar robot analytically and numerically [39]. The method has been expanded to include obstacle avoidance and applied to a more general 3-DOF robot [26].

Kim has solved the TBCO with the addition of numerous new constraints and optimization criteria [24, 40]. A multi-population GA is used as the optimization engine. The IK problem is solved numerically, and provisions are made to prevent the robot from reaching two consecutive task points by passing through singularities. The method is used to design a manipulator for servicing the ceramic tiles of the space shuttle [41]. Since the mentioned manipulator is assumed to be mounted on a mobile robot, the position of the base of the mobile robot is also considered as a design parameters.

Paredis and Khosla have enhanced the method by implementing an Agent-Based software. They have applied their method to design a fault-tolerant redundant manipulator for satellite docking aboard the space shuttle [10, 42].

Fukuda has proposed an algorithm to search for KCs, capable of performing a task, by extending a link from the base to the task point and then adding joint modules when the connection between the two points was not possible [7]. A similar approach has been employed with GA to find an optimized KC [43].

Bi has examined the application of the GAs in the TBCO [44]. Furthermore, a comprehensive list of the most important constraints and optimization criteria for industrial applications has been presented.

A two level GA approach has been adopted by Chocron. The upper level GA searches for the most suitable configuration, and the lower GA solves the IK problem [45, 46].

Park has proposed a method for finding the optimized dimensions of a manipulator in performing a task using a variation of the Grid Method, which is used in the finite difference and heat transfer/fluid analysis [47, 48]. The grid method is used to find the optimized dimensions for a handicapped assisting robotic arm [49].

A method for obtaining the dimensions of the links of a manipulator to minimize the required torque, and consequently, power, has also been proposed [50]. The method employs a GA as

the optimization engine and utilizes a Dynamic analysis package (DADS) to calculate a fitness value for the GA individuals. DADS is employed to find the dimensions of a manipulator for minimizing the required torque and the end-effector deflection [51]. The algorithm is tested and compared with a simple GA, a Differential evolutionary technique, and an elitist GA.

Although the objective of another group in the literature is automated kinematic design, they do not approach the problem with the desired task as the main input of the algorithm. Pattersson has reported an optimization method for the design of the industrial robots in the conceptual design phase [52]. The length of the links and the gearbox size of the joints are the design variables. A "Complex Optimization Algorithm" is the optimization algorithm.

Other literature discuss methods to optimize the working volume of the robotic manipulators. The focus is designing a robot that has the largest dexterous work space [53, 54]. Kircanski has proposed an analytical approach to finding the dimensions of a manipulator to maximize the condition number [55].

Rout has utilizes a Differential Evolutionary algorithm to obtain the optimal dimension and mass of the links to decrease the sensitivity of the manipulator to noise [56].

A summary of the prominent literature is shown in Fig. 2.5 and the deficiencies of the existing methods are listed as following.

- In the existing methods, a lack of performance in terms of computational requirements and speed is tangible. Most of the algorithms rely on multi-agent architectures to use a network of computers for several hours.
- Although IK is a problem with multiple solutions, most of the existing methods rely on single solution numerical algorithms to solve the IK problem. One reason for such an approach is the lack of computationally efficient multi-solution IK solvers.
- Since most of the existing algorithms use single solution IK solvers, no optimization criteria exist to incorporate multiple IK solutions in the related literature.

2.6 Inverse Kinematics

Chapters 3 and 4 are allocated to the development of efficient multi-solution IK solvers for MRRs. Therefore, in this section, the IK problem is discussed in more detail and a review of the existing approaches is provided.

The nonlinear mapping from end-effector Cartesian space coordinates into corresponding joint positions is referred to as the IK of a robot. Finding the position and orientation of the

Table 2.5: Summary of the existing TBCO literature

		Optimization Constraints			Optimization Criteria								TBCO Algorithm			
	Reference	Position Reachability Error	Orientation Reachability Error	Joint Limits	Dexterity Measure	Robot Dimensions	Links Relative Dimension	DOF	Power Consumption	Task Completion Time	Obstacle Avoidance	Self Collision	Optimization Algorithm	Considered Task Points	Kinematic Representation	IK Solver
1	[27], [28], [29]	✓	✓	–	✓	–	–	–	–	–	–	–	GA	6	AIM	Numerical Single Solution
2	[25], [34]	✓	✓	–	✓	–	–	✓	-	-	-	-	GA	4	Modified AIM	Numerical, Partially Closed-Form
3	[35]	✓	–	–	✓	–	–	✓	-	-	-	-	SA	7	Modified AIM	Numerical Single Solution
4	[26], [39]	✓	✓	✓	–	–	–	–	–	–	✓	–	SA	10	DH	Numerical Single Solution
5	[24], [41], [40]	✓	✓	✓	✓	✓	✓	✓	–	–	–	–	GA	7	DH	Numerical Single Solution
6	[10], [42]	✓	✓	✓	✓	✓	✓	✓	–	–	–	–	GA	4	DH	Numerical Single Solution
7	[7], [43]	✓	✓	✓	–	–	–	–	–	–	✓	✓	GA	4	N/A	N/A
8	[45]	✓	✓	–	✓	✓	–	–	–	–	✓	–	GA	4	Module Sequence	GA Single Solution
9	[36], [37], [38]	✓	✓	✓	–	✓	–	✓	–	–	✓	✓	GA	16	Module Sequence	Numerical

end-effector from the joint angles is called the Forward Kinematics problem. The FK of a robot manipulator can be conveniently formulated if the link parameters and joint variables of a robot are known. However, the IK is a nonlinear KC-dependent problem that can have multiple solutions [57].

To calculate the position and orientation reachability errors of a manipulator for a desired task point, the set of joint angles, in which the end-effector of a manipulator is closest to the position and orientation of the task point, should be calculated first. Furthermore, it is necessary to determine if the pose of the manipulator at a task point is physically feasible in terms of the mechanical joint limits. Therefore, to ensure the constraints of the TBCO problem (as defined in (2.9)) are satisfied, solving the IK problem is unavoidable. Similarly, a solution to the IK problem is necessary for calculating a large number of TBCO optimization criteria, including the Manipulability, power consumption, task completion time, obstacle avoidance, and self collision check.

2.6.1 Forward and Inverse Kinematics

The FK problem can be solved by specifying the position and orientation of each link with respect to the previous link as a function of the joint variable. This relative position and orientation of two consecutive links (according to the Denavit-Hartenberg(DH) convention) is described by the Homogenous Transformation of a coordinate frames attached to the end of the link with respect to a fixed frame that is connected to the origin of the frame [57]. The Homogenous Transformation has the following form

$$\begin{aligned}
\mathbf{T}_{i-1,i}(\theta_i) &= \left[\begin{array}{c|c} \mathbf{R}_{i-1,i}(\theta_i) & \mathbf{P}_{i-1,i}(\theta_i) \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right] \\
&= \left[\begin{array}{ccc|c} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right],
\end{aligned} \tag{2.21}$$

where $\mathbf{R}_{i-1,i}(\theta_i)$ and $\mathbf{P}_{i-1,i}(\theta_i)$ describe the relative orientation and position of frame i with

respect to frame $i-1$. α_i, a_i, d_i, and θ_i are the twist angle, link length, offset length, and joint angle, respectively. These parameters are extracted from the physical shape and KC of a robot.

To calculate the position and orientation of the end-effector (T_{ee}), with respect to the base of the robot, for an arbitrary set of joint angles, $\Theta = [\theta_1,\ \theta_2,\ \cdots\ \theta_n]$, the transformation is:

$$\begin{aligned} \mathbf{T}_{ee} &= \left(\begin{array}{ccc|c} & \mathbf{R}_{ee} & & \mathbf{P}_{ee} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \\ &= \prod_{i=1}^{n} \mathbf{T}_{i-1,i}(\theta_i)\ . \end{aligned} \tag{2.22}$$

The FK transformation of joint angles, $\Theta = [\theta_1\ \theta_2\ \cdots\ \theta_n]$, are represented by $K(\Theta)$ as follows

$$\mathbf{T}_{ee} = K(\Theta)\ , \tag{2.23}$$

where K is defined as follows

$$K(\Theta) = \prod_{i=1}^{n} \mathbf{T}_{i-1,i}(\theta_i)\ . \tag{2.24}$$

The inverse problem of the FK, the IK, is the problem of finding Θ from an arbitrary $\mathbf{T}_{ee}$. This problem is expressed as

$$\Theta = K^{-1}(\mathbf{T}_{ee})\ . \tag{2.25}$$

This problem is a mapping from the 3D task space, usually presented in a desired Homogenous Transformation, $\mathbf{T}_{ee} = \mathbf{T}_{des}$, to the joint angle space and usually has more than one solution. For example, a 6-DOF PUMA robot can have four or eight IK solutions, depending on the mechanical limitations of the joints [57].

2.6.2 Inverse Kinematics : A Literature Review

The approaches for solving the IK problem can be categorized into three prominent groups [18]: closed-form, numerical, and meta-heuristics algorithms.

For a handful of robotic manipulators, closed-form solutions of the IK exist (e.g. PUMA and FANUC) [57, 58, 59, 60]. In the case of 6-DOF manipulators, the sufficient condition of having a closed-form solution is that three joint axes intersect at a point[1] [61, 62]. When closed-form solutions exist, finding all of the solutions of the IK is convenient and fast. For many other serial manipulators, an analytical solution does not exist.

Geometric methods [63, 64] can be considered as a subcategory of closed-form techniques. Although the geometric methods seem promising in dealing with the IK problem of manipulators with fewer complex kinematic structures, geometric methods are not as robust as the rest of the existing methods in solving the IK for more complicated KCs.

Another approach is to use numerical methods to solve the IK problem. The advantage of numerical methods is that they are applicable to all KCs. The disadvantage is that numerical methods are usually slower than closed-form solutions. The numerical methods can be divided into the categories of symbolic elimination, continuation methods, and iterative methods.

Symbolic elimination methods employ analytic manipulations to reduce the set of nonlinear kinematic equations to a smaller set of equations. The set of kinematic equations can be systematically reduced to a single high degree polynomial of just one joint variable. The final polynomial has a degree of 16 for general 6-DOF manipulators. By back-substitution of the solutions of the polynomial into other equations, the remainder of the IK solutions are determined [65]. The same approach can be formulated into an eigenvalue problem to improve the numerical properties of the technique [66]. Also a symbolic elimination technique can be employed for redundant manipulators to find six of the joint angles of the general 7-DOF manipulators as a function of the 7-th joint [67]. In such methods, since the degree of the resulting polynomials is higher than four, numerical algorithms are required to solve the equations. Due to the extensive analytical manipulation required, the elimination methods can be categorized as closed-form techniques. The reason that here the symbolic elimination techniques are included in the numerical methods is that the final polynomial is solved by numerical means. It is noteworthy that symbolic elimination methods can turn into high complexity problems, introducing extraneous roots and can only be applied to relatively simple systems of equations. Moreover, such methods can require the solution of a high-degree polynomial, which can be a numerically ill-conditioned problem [68]. Elimination methods require extensive symbolic manipulation of the kinematic equations, specific to each type of KC prior to solving the actual problem. They are not suitable for use as a generic technique for the TBCO due to the fact that in the TBCO the IK problem should be solved for a wide range of manipulators with distinct KCs.

The second class of IK solvers, continuation methods, tracks a set of solution paths from an initial system with the known solutions to the system for which the solutions are sought [69]. The

[1] Since three parallel vectors intersect in infinity, three parallel joint axes are considered as a set of intersecting joint axes.

continuation method was first introduced as the Bootstrap method [70]. The continuation method has been responsible for the first solution of many long-standing problems of kinematics. The properties of polynomial systems can be exploited to obtain all the IK solutions [71]. Compared to the continuation methods, elimination algorithms tend to be faster and to have acceptable accuracy, when the number of roots is moderate. However, the continuation methods tend to be faster and more accurate when the number of roots is larger [72]. Although continuation methods are more efficient than elimination methods, in terms of speed in more complex manipulators, their computational performance is still one of their limiting factors.

Another category of numerical methods are iterative algorithms. Most of these depend on a class of hill climbing or a gradient descent strategy to reach a solution [73, 74, 75]. In most of the iterative methods, the algorithm converges on a single solution closest to the initial starting point. Wang has used a cyclic coordinate descent(CCD) method to rapidly find a feasible point that is close to the true solution. Then a Broyden-Fletcher-Shanno(BFS) variable metric method has been utilized to obtain a solution at the desired degree of precision [76]. Chen proposed algorithms to numerically solve the IK for MRRs represented in the AIM format [31, 32]. Also, a method for analytically solving the IK for a number of MRR KCs has been presented [12]. The algorithm is based on the Product-of-Exponentials formula and is developed with the intention of finding a single solutions of the IK. The solver can cope with the IK of all robots with 4-DOF or fewer, 90 percent of the 5-DOF robots, and 50 percent of the 6-DOF robots. Also the problem of solving IK for two task points with priority orders for a redundant manipulator is addressed[77].

The numerical iterative algorithms are the most common approaches for solving the IK problem of redundant manipulators. Most of the approaches involve the Jacobian matrix in an iterative method for converging to a solution of the IK problem. In pseudoinverse IK solvers, the pseudo-inverse of the Jacobian is utilized to iteratively decrease the distance of the end-effector of the manipulator from the desired task [78]. Since redundant manipulators have infinite IK solutions, in redundancy null space methods, the idea is to find joint angles which optimize certain criteria in addition to reaching the desired task point. Most of the null space algorithms use the pseudo-inverse of the Jacobian. The null space method can be used for maximizing the manipulability [22]. The same method can be employed for obstacle avoidance [79]. Damped least square methods provide a more resilient approach for negotiating the singularities of the manipulator [80, 81] . In damped square methods, by using a damping factor, a trade-off between the accuracy and the feasibility of the solutions are achieved. In Jacobian transpose techniques, the transpose of the Jacobian, instead of its inverse, is adopted [82]. The extended Jacobian technique, is considered as a more sophisticated null space method [83, 84]. It enforces an appropriate number of functional constraints to be fulfilled, along with the original end-effector task to identify a single solution among the infinite possible ones. Another approach, the task-space

augmentation, introduces a constraint task to be fulfilled, along with the end-effector task. Then, an augmented Jacobian matrix is formed. The inverse of the augmented Jacobian is adopted to determine the joint velocity solution [85, 86, 87].

Interval Analysis [88, 89, 68] is an iterative technique which is not based on the concept of Gradient descent. Although interval analysis can be used to find all the solutions of the IK problem, it proves to be slow for manipulators, in which the maximum number of the IK solutions are unknown or variable with respect to the desired task [90]. As a result, interval analysis can take a very long time to find all the solutions for MRRs.

A compromise between the closed-form and the numerical methods can be reached by hybrid approaches. Grudic has described an algorithm which converts the IK of an n-DOF manipulator into a bounded three dimensional problem [91]. Another algorithm decouples the manipulator into a positioning and orienting modules and then solves each separately [92]. Balkan has reported a method for making the common numerical methods more resilient to singularities [93]. All of the mentioned hybrid IK solvers are only applicable for a specific class of manipulators and should be revised when the KC of the manipulator they are designed for changes.

Metaheuristics algorithms such as GAs are another approach to solving the IK. To solve the IK for a redundant robot, a GA has been used by Parker, where the focus is on finding the solution that minimizes the joint displacements among all the possible solutions [94]. The IK problem of a 12-DOF redundant robot has also been solved. According to the article, even after the addition of heuristics to the algorithm, the results are still not completely satisfactory [95]. A GA in conjunction with fuzzy systems and hybrid immune algorithms has been used to solve the FK of a Stewart platform [96]. A GA and a Neural Network can also solve the IK for the optimal joint motions [97]. Chapelle has used a Genetic Programming algorithm to produce an estimate analytical formula for the IK [98]. The extracted formula is then applied to solve for an approximate solution to the IK at a high speed. Chocron employed a GA to find the optimized kinematic structure and the pose of a manipulator capable of performing a certain task [99]. The IK has been solved by using the same GA for resolving the kinematic optimization problem to find the optimized structure and pose of a manipulator for the prescribed task, simultaneously. Karla has reported a fitness sharing niching GA to find multiple solutions of the IK for positioning of a 2-DOF planar robot [100, 101]. A drawback of these works is that they require numerous unknown parameters. These parameters significantly depend on the nature of the search space, and differ from one robot configuration to another. More important, in these works, no consideration has been made for robots with more than 3-DOF. The identification of the solutions among the outputs of the GA is not possible with further observation for robots with more than 3-DOF. This is viewed as a major drawback, since most articulated industrial robots use at least a 3-DOF arm for end-effector positioning and at least one more DOF for orienting the end-effector. Moreover, in GA based IK solvers, the obtained solutions lack the

desired accuracy and resolution for precise position control of the end-effector.

Fig.2.11 is a diagram of the categorization of the existing IK solvers. In Table 2.6, a summary of the characteristics of the existing IK solvers is shown. In the first column, the generality expresses the ability of the algorithm to be applied to all possible KCs. Completeness of a method shows if it can return all the solutions of the IK problem. The third column indicates if a method requires a priori knowledge of the number of IK solutions. In the fourth column, if an IK solver is manipulator specific, the algorithm should be modified for each manipulator it is used for. In the last column, the relative speed of the algorithms are indicated. It should be mentioned that the speed of the methods, indicated in this column, is a rough estimation at best, and for an accurate comparison, the methods should be implemented and compared in similar conditions and cases.

As observed from the table, an unaddressed niche for IK solvers, specifically suitable for MRRs and the problem of TBCO, exists. Such an IK solver should have the following features:

- **It should be general.** Such an algorithm should be applicable to all the possible KCs which can be assembled from the inventory of modules.
- **It should be complete.** In order to incorporate all the IK solutions in the TBCO and for the optimum trajectory generation of MRRs, the desired solver must be able to find all the solutions of the IK problem.
- **It should obtain the IK solutions without any prior knowledge of the number of solutions.** Since the number of possible KCs is very large, there is no information about the number or approximate locations of the IK solutions in the joints space.
- **It should be applied to distinct KCs with minimum modifications.** Since such an IK solver is used for solving the IK problem of a wide range of distinct KCs, modifying the algorithm symbolically or analytically each time it is being applied to a new KC can prove to be an obstacle.
- **It should be fast.** Since, in the TBCO problem, the IK problem should be solved numerous times, a higher speed of IK solver is directly translated into a faster TBCO. Hence, the speed of such an algorithm is crucial to this application.

2.6.3 Objective Function of the Inverse Kinematic Problem

In closed-form IK solvers, the goal is to obtain exact algebraic solutions of the IK problem, whereas in numerical iterative methods, the approach is to formulate IK as a minimization

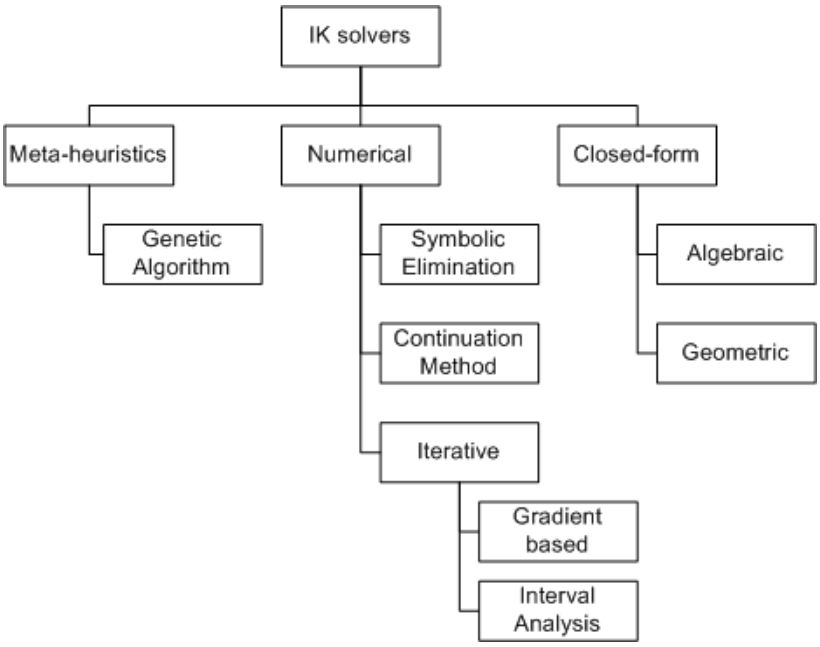


Figure 2.11: Categorization of the existing IK solvers

Table 2.6: Comparison of the existing IK solvers

Method	Generality	Completeness	Number of Solutions Required	Manipulator Specific	Speed
Closed-Form	×	✓	×	✓	Very High
Symbolic Elimination	✓	✓	×	✓	High
Continuation Method	✓	✓	×	×	Low
Iterative Gradient Based	✓	×	×	×	Medium
Interval Analysis	✓	✓	✓	×	Medium
Genetic Algorithms	✓	×	×	×	Low

problem, and then obtain the solution by employing a minimization algorithm. Therefore, the first step is to determine an optimization measure which represents the positioning/orienting error of the manipulator, when the joint angles change. For an arbitrary set of joint angles, $\Theta = [\theta_1\ \theta_2\ \cdots\ \theta_n]$, this measure is defined as the difference between the end-effector position and orientation and those of the desired task point.

In Section 2.5.3, the concepts of position and orientation reachability errors are introduced, and similar measures can be employed as the objective function in the IK problem. In solving the IK problem, a set of *joint angles* must be found to minimize the position and orientation reachability errors of a manipulator in performing a task. In the TBCO, a *manipulator* must be found to minimize the reachability errors. Therefore, the IK problem is solved as an intermediate stage in TBCO to determine the reachability errors of the manipulators. In the IK problem, the differences between the position and orientation of the end-effector from that of the desired task are called the positioning and the orientating error, whereas in the TBCO, they are called the positioning and orienting reachability error.

In the IK, the positioning and orienting difference of the end-effector, with reference to the task point produced by the joint angles Θ, can be used as the fitness measure. If the Homogenous Transformation of the task point in the cartesian space is represented by

$$\mathbf{T}_{des} = \left(\begin{array}{ccc|c} & \mathbf{R}_{des} & & \mathbf{P}_{des} \\ \hline 0 & 0 & 0 & 1 \end{array}\right), \tag{2.26}$$

and if the Homogenous Transformation of the end-effector of the manipulator at the arbitrary joint angle Θ is represented by

$$\mathbf{T}_{ee}(\Theta) = \left(\begin{array}{ccc|c} & \mathbf{R}_{ee}(\Theta) & & \mathbf{P}_{ee}(\Theta) \\ \hline 0 & 0 & 0 & 1 \end{array}\right), \tag{2.27}$$

the Euclidean norm of the difference between the end-effector position of the manipulator at Θ and that of the desired point in the Cartesian space can be used as a measure of the positioning error, i.e.,

$$\mathbf{E}_P(\Theta, \mathbf{T}_{des}) = \|\mathbf{P}_{des} - \mathbf{P}_{ee}(\Theta)\|. \tag{2.28}$$

The error in the end-effector orientation is defined by the rotation matrix, $\mathbf{R}_{error}$, which rotates the end-effector of the manipulator at Θ to the desired orientation $\mathbf{R}_{des}$. $\mathbf{R}_{error}$ can be calculated by

$$\mathbf{R}_{error} = \mathbf{R}_{des} \,.\, \mathbf{R}_{ee}^{-1}\left(\Theta\right) = \mathbf{R}_{des} \,.\, \mathbf{R}_{ee}^{T}\left(\Theta\right) \ . \tag{2.29}$$

In this thesis, the quaternion representation of frame's rotation is used for calculating the orientation error. Quaternions provide a convenient mathematical mean for representing the orientations of objects in three dimensions. Compared to Euler angles, quaternions are simpler to compose, more numerically stable, and more efficient in some situations [102], [103].

If the quaternion coordinate of a point in the Cartesian space is represented by $\mathbf{V}$, it can be shown that the quaternion product $\mathbf{QVQ}^{-1}$ yields the vector $\mathbf{V}$ rotated by an angle α around the axis of the quaternion vector $\mathbf{U}$. Here, $\mathbf{Q}$ is the rotation vector and can be represented as $\mathbf{Q} = [\,\cos(\alpha/2) \quad \sin(\alpha/2).\mathbf{U}\,]$.

In the IK formulation, the rotation matrix, required to rotate the end-effector Homogenous Transformation to the desired orientation, can be conveniently converted to the quaternion format. The α can then be extracted from the quaternion to represent the orientation error.

If $\mathbf{R}_{error}$ is defined in the quaternion representation by $[q_w \ q_x \ q_y \ q_z]$, where $[q_w \ q_x \ q_y \ q_z] = [\,\cos(\alpha/2) \quad \sin(\alpha/2).\mathbf{U}\,]$, and operator $\mathcal{Q}$ is defined as

$$\mathcal{Q}\left(\mathbf{R}_{error}\right) = \alpha \ , \tag{2.30}$$

The orientation error can then be written as

$$\mathbf{E}_O\left(\Theta, T_{des}\right) = \mathcal{Q}\left(\mathbf{R}_{error}\right) \ . \tag{2.31}$$

By using (2.28) and (2.31), the objective function of the manipulator at Θ, consisting of both positioning and orienting errors with reference to a desired task point $\mathbf{T}_{des}$ can be written as

$$F_{obj}\left(\Theta, \mathbf{T}_{des}\right) = w_p\, \mathbf{E}_P\left(\Theta, \mathbf{T}_{des}\right) + w_o\, \mathbf{E}_O\left(\Theta, \mathbf{T}_{des}\right) \ , \tag{2.32}$$

where w_p and w_o are the positioning and orienting weighting factors. These weighting factors are used to normalize the corresponding values of $\mathbf{E}_P$ and $\mathbf{E}_O$. Following the same course of reasoning for (2.13), w_p and w_o are considered as follows:

$$\begin{cases} w_p &= 1 \\ w_o &= \frac{\sqrt{\lambda_{KC} \cdot \|\mathbf{P}_{des}\|}}{\pi} \end{cases} , \tag{2.33}$$

where λ_{KC} is a coefficient that depends on the dimension of the robot. Total length of all the links of the robot can be used as λ_{KC}.

The goal of the numerical IK algorithms is to solve for Θ^m such that

$$\Theta^m = \arg\min_{\Theta} F_{obj}(\Theta, T_{des}) \, . \tag{2.34}$$

In a manipulator capable of reaching $\mathbf{T}_{des}$, if $F_{obj}(\Theta^m, \mathbf{T}_{des})$ is not zero, Θ^m is a local optimum of the function F_{obj}, and is not a solution to the IK problem. A manipulator pose, $\Theta^s = [\theta_1^s, \, \theta_2^s, \, \cdots \, , \theta_n^s]$, is an IK solution, if and only if

$$F_{obj} \left(\Theta^s, T_{des} \right) = 0 \, , \tag{2.35}$$

which implies that

$$\mathbf{E}_P \left(\Theta^s, T_{des} \right) = 0 \, , \tag{2.36}$$

and

$$\mathbf{E}_O \left(\Theta^s, T_{des} \right) = 0 \, . \tag{2.37}$$

2.6.4 Multi-Solution Versus Single-Solution IK Solvers

In this section, the two fields that benefit the most by employing multi-solution instead of single solution IK solvers, are discussed.

Manipulator Kinematic Design

Typically, all industrial manipulators have specific kinematic designs such that closed-form IK solutions can be developed [104]. In other words, one of the reasons that almost all the existing serial manipulator designs are minor variations of a few well known KCs such as PUMA and Stanford is to make a closed-form IK solution possible. To guarantee closed-form solutions,

almost all of the existing industrial manipulators are designed to have three intersecting or parallel consecutive joint axes [61, 62]. Since, in the manipulator design, the main limitation is the existence of closed-form solutions, only a fraction of all the possible KCs are synthesized, designed, and developed. Hence, a large number of KCs which can offer better performance are neglected.

Although numerical IK solvers provide a competitive replacement for substituting closed-form solutions, especially for KCs without closed-form solutions, the numerical IK solvers lack the capability to find all the IK solutions. The single solution nature of numerical IK solvers results in new limitations in terms of joint mechanical limitation avoidance, obstacle avoidance, trajectory planning, and trajectory optimization. This limitation has restricted the practical application of numerical IK solvers such that the closed-form still remains the preferred approach for solving the IK problem.

Therefore, an IK solver algorithm that is not restricted by the limitations of closed-form and numerical solvers, i.e. one which is capable of obtaining multiple IK solutions for all KCs, provides the designers with the required tools to create manipulators which can perform more efficiently in general tasks or manipulators which are optimized for certain tasks.

Trajectory Optimization

Industrial robotic manipulators are designed to perform a certain task numerous times during their operational age. Consequently, any change to these manipulators, which might improve the performance has considerable rewards in terms of decreasing operations, maintenance, and repair costs, as well as the manufacturing time.

Different approaches and solutions can be found for optimizing the performance of robotic manipulators. The least costly, in terms of material and time, is probably to modify the robot trajectory such that a set of operational performance parameters of the robot can improve. For instance, the trajectory of the manipulator can be optimized in order to minimize the required torque, power consumption, and operation time.

Multiple solutions of the IK can lead to more optimal solutions to the trajectory generation problems. The effect of considering multiple IK solutions in the trajectory optimization, are illustrated through the following cases:

Example 1: For each task point in the Euclidean space, the 6-DOF PUMA manipulator in Fig.2.12 has eight distinct IK solutions. For the purpose of comparison, a task for the manipulator consisting of moving from task point 1 ($\mathbf{T}^1_{des}$) in the Euclidean space to task point 2 ($\mathbf{T}^2_{des}$) is assumed. Fig.2.12(a) and Fig.2.12(b) show two out of the eight distinct

IK solutions for $\mathbf{T}^1_{des}$, called elbow-up and elbow-down poses. Fig.2.12(c) and Fig.2.12(d) demonstrate the elbow-up and elbow-down poses for $\mathbf{T}^2_{des}$. It can be observed that four cases occur, when a trajectory from $\mathbf{T}^1_{des}$ to $\mathbf{T}^2_{des}$ is generated.

1. The manipulator moves from the elbow-up pose in $\mathbf{T}^1_{des}$ to the elbow-up pose in $\mathbf{T}^2_{des}$.
2. The manipulator moves from the elbow-down pose in $\mathbf{T}^1_{des}$ to the elbow-down pose in $\mathbf{T}^2_{des}$.
3. The manipulator moves from the elbow-up pose in $\mathbf{T}^1_{des}$ to the elbow-down pose in $\mathbf{T}^2_{des}$.
4. The manipulator moves from the elbow-down pose in $\mathbf{T}^1_{des}$ to the elbow-up pose in $\mathbf{T}^2_{des}$.

For each of these cases, a 7-degree polynomial trajectory with the constant length of four seconds is created. The required joint torque and power consumption of each trajectory is calculated for the PUMA560 manipulator by using the Matlab Robotics Toolbox [105]. This toolbox utilizes the recursive Newton-Euler formulation to solve the Inverse Dynamics problem.

Table 2.7 and Table 2.8 show the maximum and the average required torque for each joint of the PUMA560 to follow the trajectory. Table 2.9 shows the average power required for each joint and the total required power for the manipulator.

It can be seen that the maximum and average required torque and the average required power change drastically from case to case. This change in the performance measure, caused by choosing different IK solutions for a unique task, suggests that the performance of the manipulator can be improved by a suitable selection of the IK solutions for each task. The selection of the best IK solutions depends on the application and whether the goal is minimizing power, maximizing the payload, maximizing the speed, identifying the optimal sequence to pass through the intermediate points of a given task, or a combination of them. Generally, the desired optimization measure should be evaluated and multiple options for the path planning should exist, i.e., more than one solution to the robot IK problem. If the manipulator has s IK solutions for each of t task point, s^t different cases exist. Thus, a search algorithm is required to examine such cases in order to find the IK solutions that optimize the trajectory for a given set of goals. In Chapter 5, an efficient algorithm to perform this search is proposed and tested.

If minimizing the power consumption of the manipulator is the goal, as seen from Table 2.9, choosing the trajectory of Case 2 offers the best improvement. Case 2 corresponds to the trajectory from the elbow-down pose of $\mathbf{T}^1_{des}$ to the elbow-down pose of $\mathbf{T}^2_{des}$.

Table 2.7: Example 1: Average required torque for the PUMA560 to follow the trajectories (N.m)

Torque(N.m)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Case 1	26.2323	28.0785	10.9776	0.8712	1.8214	1.6948
Case 2	26.2350	25.0806	1.1347	1.9331	0.5412	0.8107
Case 3	26.4836	35.4458	5.2365	2.0909	1.2226	0.8327
Case 4	26.2316	53.9799	12.3313	0.9127	2.6467	1.9210

Table 2.8: Example 1: Maximum required torque for the PUMA560 to follow the trajectories (N.m)

Torque(N.m)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Case 1	49.8973	40.4988	12.9466	1.7768	3.4713	3.3710
Case 2	48.5416	39.3951	13.0186	4.1371	1.1043	1.6374
Case 3	55.9225	71.1438	9.9031	4.4306	1.8467	1.2812
Case 4	56.5868	68.1273	24.6538	1.8695	4.7908	3.7374

As a result, multiple solutions of the IK comprise a first step in creating trajectories that are capable of optimizing the operation of a robotic manipulator. In Chapters 3 and 4, two novel algorithms, for obtaining multiple solutions of the IK problem for serial manipulators with revolute joints, are proposed.

Example 2: A common approach to trajectory optimization is to choose IK solutions such that the elbow-up or down stance of the manipulator does not change. Although this approach might prove to provide correct solutions in some cases, this example shows that such a method does not always produce the desired solutions.

A 6-DOF PUMA manipulator with an offset between the second and third joint for performing a two task point trajectory is considered. The PUMA is illustrated in Fig. 2.13. The eight IK solutions of the manipulator, when it reaches the task points, are computed, and the 64 combinations of moving from an IK solution of the first task to the second task are examined. The result of the comparison is shown in Fig. 2.13 in which the optimal pose at the initial and end tasks are shown in Fig.2.13(a) and Fig. 2.13(b), respectively.

Table 2.9: Example 1: Average required power for the PUMA560 to follow the trajectories (W)

Power(W)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	$\sum$
Case 1	24.9752	12.3913	2.6832	0.7107	5.3727	3.7535	49.8866
Case 2	26.4819	16.7639	0.1891	4.8971	0.3554	0.6260	49.3135
Case 3	24.1281	73.1677	11.4001	5.4981	0.6969	0.3513	115.2422
Case 4	24.5108	48.0250	37.0315	0.8285	10.9475	4.8569	126.2002

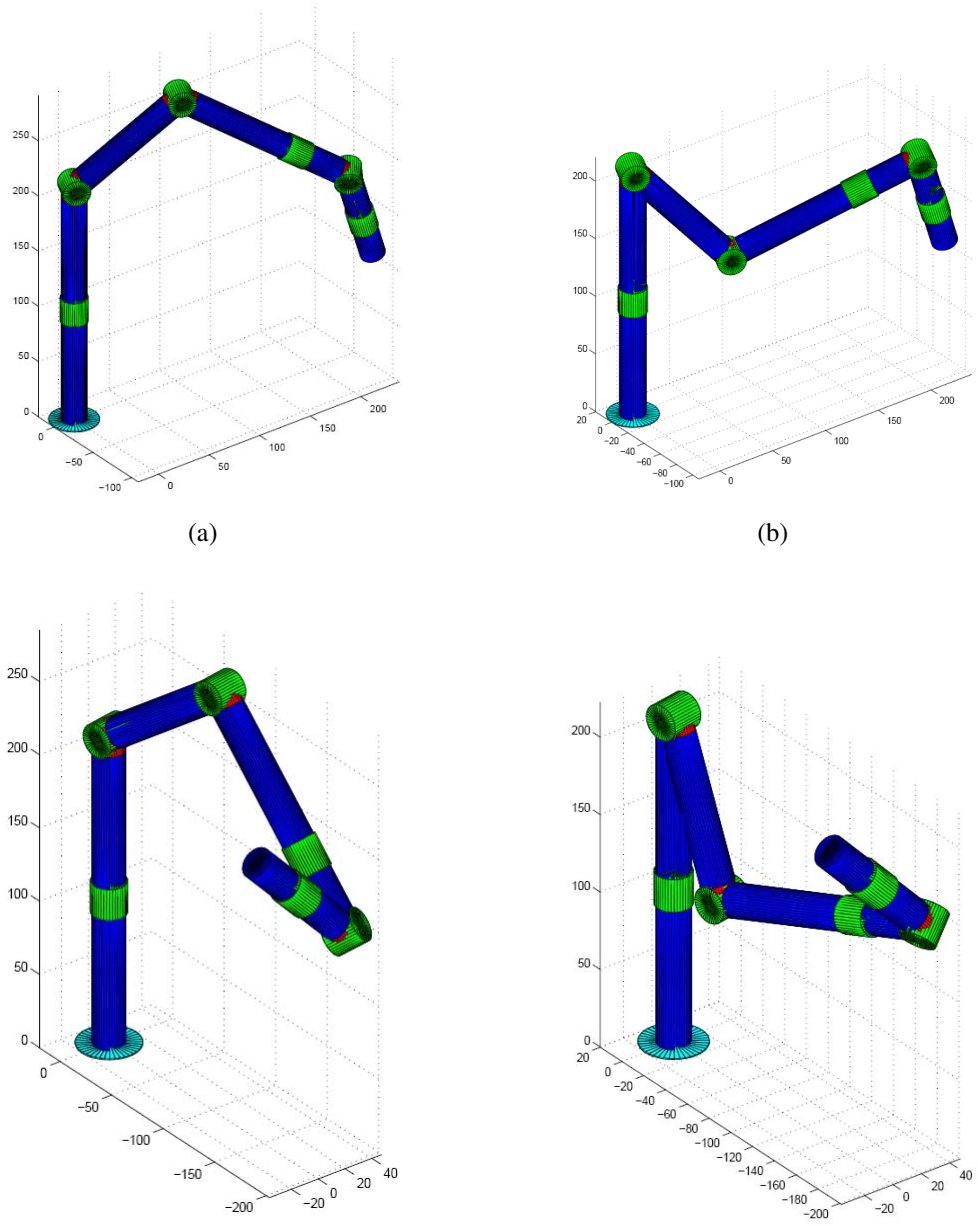
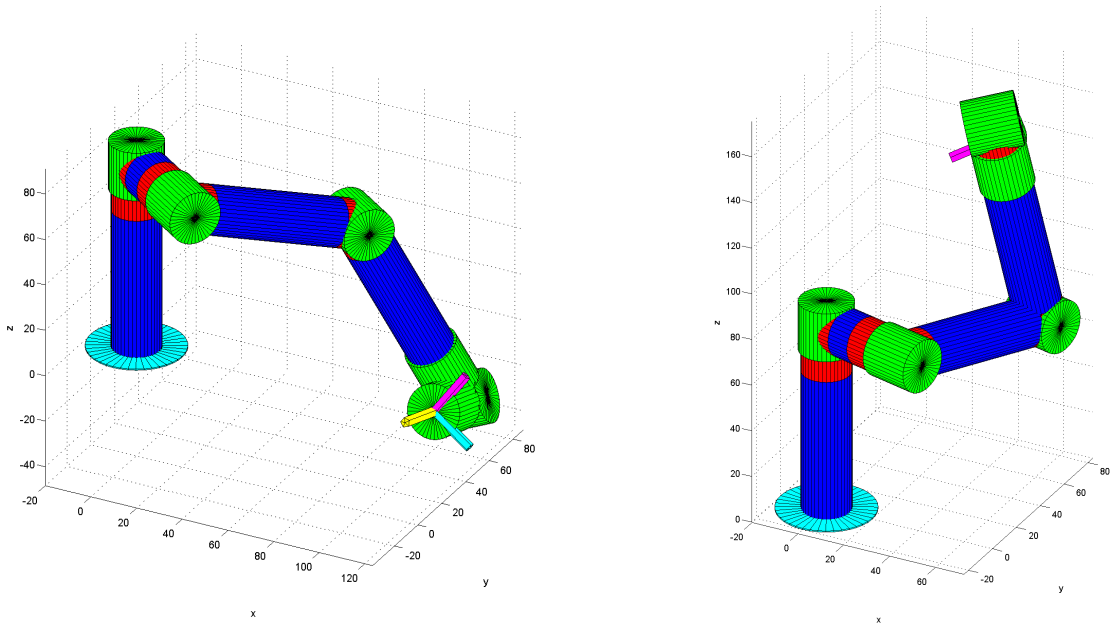


Figure 2.12: Example 1: Start and end point of a certain task considering two distinct solutions of the IK for a 6-DOF spatial manipulator: (a) start pose with IK solution 1; (b) start pose with IK solution 2; (c) end pose with IK solution 1; and, (d) end pose with IK solution 2



(a) (b)

Figure 2.13: Example 2: Optimized start and end pose of the 6-DOF PUMA manipulator in moving from $\mathbf{T}^1_{des}$ to $\mathbf{T}^2_{des}$: (a) start pose with IK solution 1; and, (b) start pose with IK solution 2

At the initial task point, the pose is elbow-up, and at the end, the pose is elbow-down. Therefore, preventing the manipulator from passing through singularities, by preventing it from changing its elbow-up or down stances, is not enough for optimizing trajectories. Consequently, a more comprehensive method for considering all the possible cases, and finding trajectories from one task point to another is required.

2.7 An Introduction to Genetic Algorithms (GAs)

Genetic Algorithms (GAs) are the most commonly used optimization algorithm in the TBCO literature. In this thesis, two distinct classes of GAs are employed for solving two different but related problems. In Chapter 3 a niching Genetic Algorithm is used to solve for multiple solutions of the IK problem. A Memetic Algorithm (MA) which is a hybrid of local search and genetic algorithms, is employed in Chapter 6 to solve the TBCO problem. Therefore, in this section, simple GAs are introduced. In the respective chapters, the complementary detail of the selected GA classes are presented.

2.7.1 Genetic Algorithms

According to Goldberg's most cited text book[2] [106]:

> Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance.

In GAs, the design parameters are translated to a string. Each string, called an *individual*, represents a point in the search space. Unlike other optimization algorithms, GAs manipulate a group of individuals (called a *population*) to reach a solution. The individuals in the initial population are generated randomly. A *fitness value* for each individual is calculated according to the criteria that need to be optimized. This fitness value is a measure of the suitability of each individual as a solution. For instance, in TBCO, each string represents an encoded KC in the $\mathbb{M}_n$ space, and the fitness value that should be minimized can be reachability errors or an optimization criterion. In this thesis, the i-th individual and its fitness value are represented by X_i and $F_{fv}(X_i)$, respectively, where X is substituted with the corresponding representation of the optimization variable. For instance, in the TBCO, each individual is a KC and $X_i = R_i$, while in an IK solver, each individual is a set of joint angles, $X_i = \Theta_i$.

GAs use the genetic operators to find the individual that can produce the lowest(highest) fitness value for a minimization(maximization) problem. The genetic operators consist of selection, crossover, and mutation. In Selection (also called reproduction), the algorithm selects the individuals with the best fitness values to form a parent pool. The parent pool is a group of the individuals with a high fitness value. The individuals in the parent pool are given the chance of producing offsprings. The members of the parent pool are called parents.

After selection, a simple Crossover can proceed in two steps. First, members of the parent pool are mated at random. Secondly, two new individuals are created by swapping the strings of the parents at a randomly selected position. Crossover produces a new population from the promising individuals of the previous generation. In doing so, by inheriting good genes from the parents, the average fitness value of the new generation usually decreases.

[2] www.citeseer.com, in artificial Intelligence category.

The mutation operator is then applied to a small percentage of the offsprings. Mutation changes a random part of the string. For example, in binary strings, simple Mutation changes one bit of the string (from 0 to 1 or vice versa). Mutation is needed because, "even though reproduction and crossover effectively search and combine extant notions, occasionally they may become overzealous and lose some potentially useful genetic material. In artificial genetic systems, the mutation protects against such irrecoverable loss" [106].

After mutation, a new generation of individuals is formed. Now, the GA operators are applied to the new generation, and this process continues, until the termination criteria is satisfied or the maximum number of the generation is reached.

Fig. 2.14 shows the schematics of a simple GA.

2.7.2 Genetic Algorithms in Task-Based Configuration Optimization

The current literature on optimization identifies three main types of search methods: calculus-based, enumerative, and random [106]. Calculus-based methods use the derivative of the objective function to find the optimized solutions. Enumerative methods look at the objective function values at each point in the space, one at a time. Though, these methods are simple and very effective in finding the optimums, they lack the efficiency required for searching large multi dimensional spaces. On the other hand, random search methods (including GAs) improve the efficiency of enumerative methods by sampling the search space to locate the optimums. For applications that demand high efficiency, the random methods are the logical choice. The drawback of random search methods is that a formal mathematical proof for their convergence rarely exists. Moreover, random search algorithms are usually slower than the other methods in converging to the solution.

Another approach to categorize the optimization algorithms is grouping them into local or global algorithms. Local optimization algorithms converge on the minimum which is closest to the initial search point. If the found minimum is not the minimum of the function in all of the search space, it is called a local optimum. A global optimization algorithm is independent of the initial starting point, and finds the absolute minimum of the objective function. Although in some applications, finding only a local minimum might suffice, in most applications, the global optimum is the desired point. Most enumerative methods are categorized in the local optimization groups. Calculus-based and random search algorithms are typically global optimization algorithms.

The characteristics and relevance of GAs to the TBCO can be summarized in the following [106].

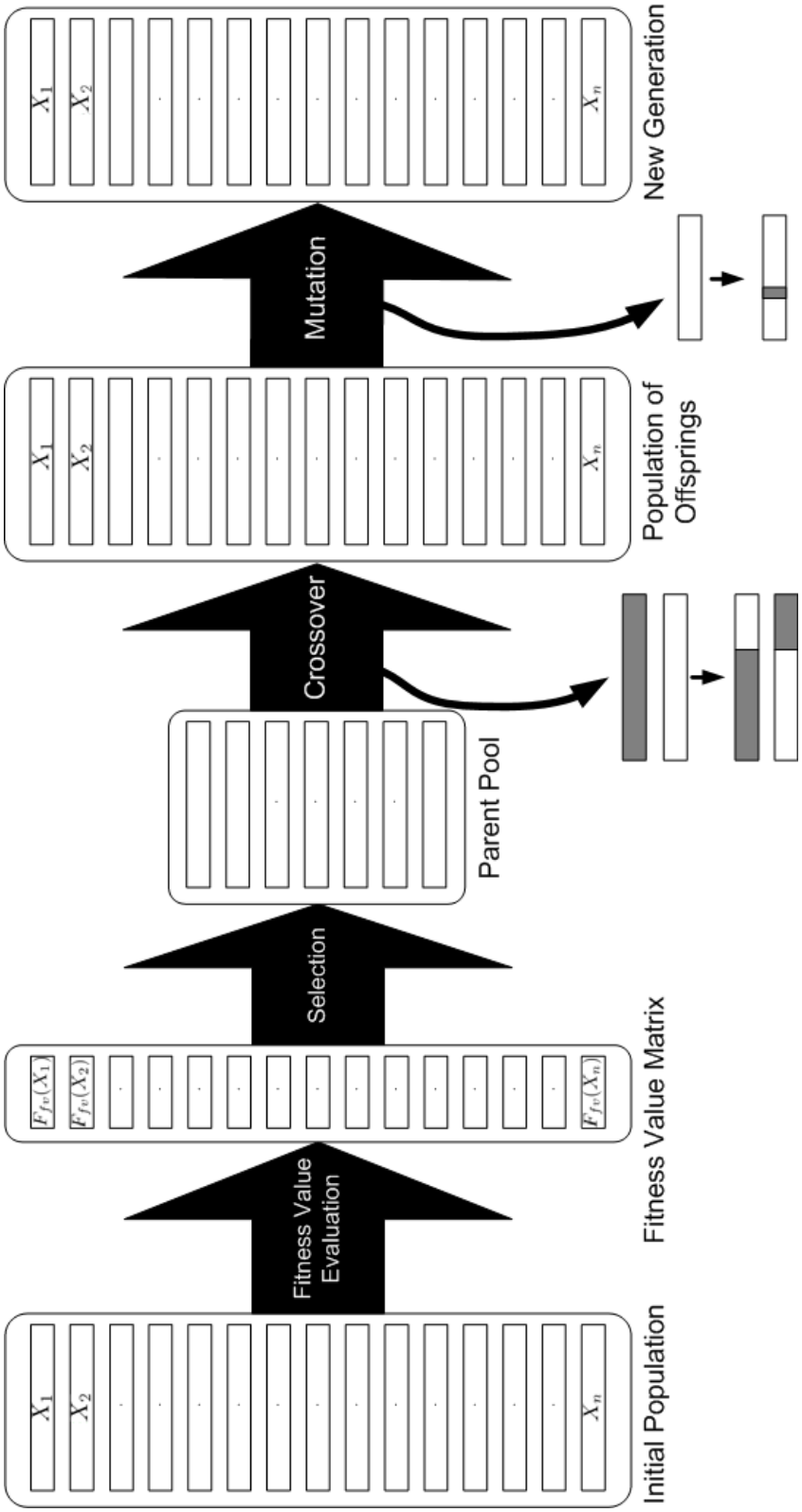


Figure 2.14: Block diagram of an iteration in Genetic Algorithms

1. *GAs work with a coding of the parameter set, not the parameters themselves.* This characteristic enables a search in a space that consists of points with numerous types of parameters. For instance, each of the individuals can include continuous, discrete, and binary parameters at the same time. This is an asset in formulating a complex problem such as the TBCO which includes different types of variables.

2. *GAs search from a population of points, not a single point.* In many optimization methods, each point is selected according to the fitness value of the last point. This makes the method susceptible to convergence on local optima. On the other hand, GAs produce individuals that climb multiple hills, simultaneously. This feature enables GAs to find the global optimums more conveniently. The TBCO is a highly nonlinear problem with numerous local optima. Hence, this feature of GAs is crucial in avoiding the local optima and moving toward the global optima.

3. *GAs use an objective function, not derivatives or other additional knowledge.* In the majority of complex optimization problems, including the TBCO, no information of the search space such as derivatives exists. Since GAs only require a fitness value corresponding to each of the individuals in the population, working in an unknown search space is convenient. However, for other methods, especially calculus-based methods, lack of derivatives is a huge obstacle.

4. *GAs use probabilistic transition rules, not deterministic rules.* GAs use random choice to guide the search towards regions of the search space with likely improvements. This feature increases the chance that GAs can avoid the local optimums.

Due to the advantages of GAs for solving optimization problems, in this thesis they are adopted as the preferred optimization algorithm for solving the TBCO problem. Furthermore, a class of GAs is employed for solving the IK problem for all of its solutions.

Chapter 3

Multi-Solution Inverse Kinematics Solver Based on Adaptive Niching Genetic Algorithms

3.1 Introduction

In this chapter, an adaptive niching GA method is proposed to find multiple solutions of the IK problem. This algorithm is based on a minimizing GA to find the joint angles that produce the least positioning and orientation error of the end-effector from those of the desired task point.

Each individual of the GA is encoded such that it represents a set of joint angles. For an n-DOF manipulator, the i-th individual of the GA, Θ_i, is expressed as follows

$$\Theta_i = \left[\theta_1^i \ \theta_2^i \ \cdots \ \theta_n^i \ \right] . \tag{3.1}$$

For an individual, Θ_i, a fitness value, $F_{fv}\left(\Theta_i\right)$, representing the suitability of the individual for being an IK solution, is calculated. The goal of the proposed algorithm is to find a group of individuals with F_{fv} within the permissible threshold. The weighted sum of the position and orientation error is adopted as the fitness value of each individual. According to Section 2.6.3, the fitness function of the i-th individual Θ_i with respect to the desired Homogenous Transformation T_{des} can be written as:

$$F_{fv}\left(\Theta_i\right) = F_{obj}\left(\Theta_i, \mathbf{T}_{des}\right) = w_p.\,\mathbf{E}_P\left(\Theta_i, \mathbf{T}_{des}\right) + w_o.\,\mathbf{E}_O\left(\Theta_i, \mathbf{T}_{des}\right) \tag{3.2}$$

Since a requirement for the algorithm is to find multiple solutions of the IK problem, a special class of GA called niching GA is employed.

The development of the proposed niching GA involves the following:

- By using the niching GA, the method can yield multiple solutions of the IK problem. The proposed method requires only a few parameters based on prior knowledge of the problem. Consequently, the method can be applied to a wide range of distinct KCs.
- A real coded Simulated Binary Crossover (SBX) is chosen. It enables the algorithm to search in a continuous joint space, not a discrete binary one.
- A novel formulation for incorporating the joint mechanical limits in the simulated binary crossover is introduced.
- An enhanced version of the adaptive niching via coevolutionary sharing method is adopted to increase the speed of the algorithm without sacrificing the performance.
- A post processing stage, consisting of filtering, clustering, and numerical IK is proposed. The filtering and clustering stages allow the algorithm to be used for robots with more than 3-DOF through a methodical identification of the solution regions. The numerical IK solver then achieves convergence at any desired joint angle accuracy. In addition, the numerical stage enables the algorithm to distinguish the global optimums from the local optimums in the output of the niching GA.

The performance of the algorithm is verified by finding multiple solutions of the IK problem for four distinct KCs. The algorithm is applied to instances of $\mathbf{m}_3$, $\mathbf{m}_4$, $\mathbf{m}_6$, and $\mathbf{m}_7$ manipulators for two distinct task points.

This chapter is organized as follows. In Section 3.2, the existing niching GA algorithms are reviewed. The niching GA algorithm for solving the IK problem is introduced in Section 3.3. Section 3.4 provides a discussion of the post processing stage to extract the IK solutions from the output of the niching GA algorithm. In Section 3.5, the results of employing the proposed method to solve the IK of four distinct KCs are presented. Section 3.6 is a summary the chapter.

3.2 Niching Techniques

A GA, through Selection, Cross-over, and Mutation operations, finds the individuals that have the best fitness values, and combines them to produce individuals that offer better fitness values than those of their parents. This process continues until the population converges around an individual that has the best fitness value. However, in a large number of applications with multiple global (or local) optimums, the identification of more than just one promising point is necessary.

For this purpose, niching methods are adopted to modify the simple GA by changing the fitness value such that a convergence to multiple solutions are encouraged [107]. In this section, the conventional niching techniques are reviewed. Then, the adaptive niching via the coevolutionary sharing technique is explained in detail.

3.2.1 Conventional Techniques

The sharing method, which is probably the most well-known niching technique, decreases the fitness value of the individuals in densely populated areas, decreasing their chance of being selected [108]. In sharing methods, a priori knowledge of the problem is required to tune the numerous parameters of the algorithm, including niche radius, σ_s [107]. Moreover, the algorithm is more suitable for problems with equidistant niches. Its limitation as an IK solver is that, prior to solving the IK problem, no knowledge of the relative proximity of the solutions exists. In addition, the number of niches, which represent the IK solutions, changes for different KCs. Also, these solutions change with the position of the end-effector, and are completely different from one robot configuration to another. Crowding methods, another approach to niching, includes standard crowding, deterministic crowding, and restricted crowding. These methods do not have the robustness of sharing methods [107].

3.2.2 Adaptive Niching via Coevolutionary Sharing

As mentioned, one of the disadvantages of fitness sharing is the need to set the niche radius σ_s accurately. This requires a priori knowledge of the proximity and distances between the solutions of the problem, a luxury which is not available in IK problems.

To address this drawback of sharing methods, Goldberg and Wang have introduced the adaptive coevolutionary shared niching (CSN) algorithm [109]. It is loosely based on the economic model of the *monopolistic competition*, in which businessmen try to position themselves, subject to a minimum distance among geographically distributed customers, to maximize their profit. In the CSN two populations, businessmen and customers, work to maximize their separate interests.

These two populations interact with each other according to the economic model. Businessmen try to maximize their profit by finding locations with more customers, whereas customers try to shop from businessmen with better service, that is, the closest businessman who is least crowded.

For the customer population, the fitness function modification resembles that of the standard fitness sharing. If, at any generation t, customer c is being served by the businessman b who is

the closest businessman, and that b is serving a total $C_{b,t}$ customers, the shared fitness of c is calculated by

$$F_{shared}(c) = \left. \frac{F_{fv}(c)}{C_{b,t}} \right|_{c \in C_b} , \tag{3.3}$$

where C_b denotes the customer set, whom businessman b serves, that is, each customer shares its fitness value with the other customers of the same businessman. A stochastic universal selection scheme and single point crossover have been used in the original paper.

The tendency of the businessmen is to place themselves in regions that are more densely populated by customers, subject to keeping a minimum distance of d_{min} from the other businessmen. The fitness value of the businessmen is simply the sum of the fitness values of its customers such that

$$F_{shared}(b) = \sum_{c \in C_b} F_{fv}(c) \, . \tag{3.4}$$

Goldberg and Wang have used only a mutation operation for the businessmen population. If a mutated individual is at least d_{min} far from other businessmen, and is an improvement over the original businessman, the individual replaces the original one. If not, the mutation operation is repeated up to a multiple of the businessman population. Also, an imprint operation has been suggested. It chooses a new businessman randomly from the customer population instead of producing it by mutation. With the imprint, the evolution of the businessman population benefits from knowledge of the search space acquired by customers, and is not completely random. If the chosen customer can satisfy the aforementioned two conditions it replaces the businessman. To investigate if the selected customer is an improvement, the assignment of the customers to the businessmen must be repeated. To accomplish the assignment, the calculation of the customer's distance from the members of the new set of businessmen is required.

The sensitivity of CSN to d_{min} is less than the sensitivity of sharing techniques to σ_s. Nevertheless, choosing an appropriate d_{min} is still of considerable importance.

The CSN has been applied to a multi-objective softkill-scheduling problem with the imprint operation [110]. Rank-based selection, elitism, and non-dominated sorting are some of the prominent features of that work.

3.3 Adaptive Sharing to Solve the IK Problem

To solve the IK problem, the algorithm must be fast enough to evaluate the solution for a very large space (e.g. a six dimensional space for a 6-DOF PUMA or general purpose articulated robots). Furthermore, the algorithm must be capable of not only finding multiple solutions for all the poses of the end-effector, but also solving the IK for any KC. In this section, the proposed algorithm for solving the IK problem is introduced.

3.3.1 Niching GA for Solving the Inverse Kinematic Problem

An overview of the proposed algorithm is displayed in Table 3.1. A detailed explanation of each step follows.

1. Initialization: Two independent populations of customers and businessmen are randomly created. Each individual, Θ_i, consists of n joint angles corresponding to the n DOF of the robot such that

$$\Theta_i = [\theta_i^1 \ \theta_i^2 \ \cdots \ \theta_i^n] \,, \tag{3.5}$$

where $\theta_i^1,\ \theta_i^2,\ \cdots,\ \theta_i^n$ are real numbers.

To allow more individuals to be associated with the IK solutions which are close to the reachable joint space borders, $[\theta_{min}$,$\theta_{max}]$, an extended range of permissible angles, $[\theta_{min} - \psi$,$\theta_{max} + \psi]$, is used. θ_{min} and θ_{max} are the joints' rotational limitations which are usually dictated by the mechanical design and manufacturing.

After the customer and businessmen populations are randomly generated, the initial d_{min} is calculated as follows

$$d_{min_{start}} = \frac{\kappa(\,\theta_{max} - \theta_{min})}{1 + \sqrt[n]{b}} \,, \tag{3.6}$$

where n, b, and κ correspond to the DOF (i.e., number of joints), the number of businessmen, and the fitting index, respectively. Equation (3.6) uses $\kappa > 1$, multiplied by the distance between any two businessmen, if they are spread equidistantly over the n dimensional joint space, that is, $d_{min_{start}}$ should be greater than the average distance between any two businessmen.

2. Fitness Value Calculation: The fitness values, $F_{fv}(c)$ and $F_{fv}(b)$, of the customers and businessmen are calculated by using (3.2). Then, the customers are assigned to the closest

businessman, where the closeness is measured by using the Euclidean distance between the customers and businessmen. If, at any generation g, customer c is served by businessman b who is the closest businessman, and that b is serving a total $C_{b,t}$ customers, the shared fitness of c, $F_{shared}(c)$, is calculated by

$$F_{shared}(c) = F_{fv}(c) \cdot C_{b,g} \ . \tag{3.7}$$

This equation is the counterpart of (3.3) in the original maximization CSN algorithm, which has been modified for minimization. By using the niched fitness value, $F_{shared}(c)$, in the GA, the tendency of the selection operator is to form the parent pool from individuals with a smaller $C_{b,g}$, that is, individuals in less dense locations. The niched selection scheme preserves the diversity of the individuals, and prevents the individuals from converging to one single solution of the problem.

3. Selection: A tournament selection is adopted to create the parent pool. From the customers, n_{tour} individuals are selected at random. From this subset, the customer with the least fitness value (error) is transferred to the parent pool. A Binary Tournament Selection, in which $n_{tour} = 2$, is used for the proposed algorithm.

4. Elitism: It has been demonstrated that elitism can speed up the performance of the GA significantly [111, 112]. Also, elitism can help prevent the loss of good solutions once they have been found. A drawback of using elitism is a premature convergence of the algorithm due to one elite, which is not the global optimum, dominating the parent pool, and therefore after a few generations the entire population.

In simple GAs, the elitism is performed by transferring a number of the customers with the best fitness values to the next generation directly. But in niching GAs, by applying the same elitism method, there is the chance that all the elites are chosen from only a few well developed niches. This can decrease the diversity of the population quickly and cause premature convergence of the algorithm. In this thesis, a method is developed to profit from elitism while avoiding its potential drawbacks. The proposed elitism scheme is performed as follows. Instead of simply choosing the best customers from the entire pool of customers, one customer, belonging to each businessman, is selected. This customer has the lowest fitness value over all the other customers belonging to the same businessman. As a result, each businessman contributes one elite, unless the businessman does not have any customers. If the number of elites is odd, a randomly selected customer is added to the list of elites. Then, the elite list is transferred to the next generation population. The crossover operator is used to find the rest of the customers in the next generation population.

5. Customer Crossover: In this step, a new generation of customers is produced from the parent pool of the previous step. Simulated Binary Crossover (SBX)[113] is adopted for the crossover operation. In Section 3.3.2 the SBX and the proposed modifications are presented to accommodate joint angles with physical limits.

6. Businessmen Imprint: Each businessman is compared with an individual, randomly selected from the newly formed parent pool. If this individual is an improvement over the businessman and is d_{min} away from all the other businessmen, the individual replaces the corresponding businessman. For each businessman, the process of comparison is repeated n_{limit} times, or until the businessman is replaced by a better candidate. Here n_{limit} is a multiple of the population number of businessmen.

7. Updating d_{min} : The value of d_{min} is closely related to the accuracy of the end solutions. Lower values of d_{min} bring flexibility to the businessmen to locate regions with better fitness values and higher concentration of solutions. The drawback of having a small d_{min} is an increase in the probability of losing some niches, because of the tendency of the businessmen to converge at the regions with the highest customer concentration.

In the initialization of the algorithm, d_{min} is set at its maximum value to prevent the GA from converging immaturely on only one niche. As the iterations continue, the niches begin to establish themselves around the solution points. Then, the difference between their fitness values and the number of individuals in different niches decreases.

In this step of the algorithm, d_{min} is decremented in small step sizes until it reaches a certain lower limit. In the GA proposed here, the following function is used for updating d_{min} :

$$d_{min} = d_{min_{start}}(1 - \lambda \frac{g}{g_{max}}) \,, \tag{3.8}$$

where g and g_{max} correspond to the current iteration and maximum iteration number. λ is the coefficient that defines how small d_{min} can become.

8. Checking the Termination Criteria: In this stage, the output of the algorithm is compared to a termination criteria. If the average fitness value of the businessmen population, $F_{avg,g}(b)$, in generation g can satisfy the following criterion, the algorithm is interrupted, and the results are entered into the post-processing phase. $F_{avg,g}(b)$ is computed as follows

$$\begin{cases} F_{avg,g}(b) & \leq & \mu F_{limit} \\ \\ F_{limit} & = & w_p \,.\|\mathbf{P}_{des}\| + w_o \,.\mathcal{Q}\left(\mathbf{R}_{des}\right) \end{cases} , \tag{3.9}$$

Table 3.1: Niching GA for solving the IK problem

Step	Description
1	Randomly Initialize the Customer Population Randomly Initialize the Businessman Population Initialize d_{min} with $d_{min_{start}}$
2	Customers Raw Fitness Value Calculation Businessmen Raw Fitness Value Calculation Assignment of Customers to the Closest Businessman Customers Shared Fitness Value Calculation
3	Forming the Customers Parent Pool by Tournament Selection
4	Transferring the Elite Customers to the Next Generation
5	Customer Crossover
6	Businessmen Imprint
7	Updating d_{min}
8	If the Termination Criterion Is Not Reached Return to Step 2

where $\|P_{des}\|$ and $\mathcal{Q}(\mathbf{R}_{des})$ represent the norm of the position and the quaternion representation of the orientation of the task T_{des}, respectively. F_{limit} is considered as an upper bound on the fitness value, F_{fv}, that depends only on the position and orientation of the task point. $0 \leq \mu \leq 1$ is a coefficient that represents how small the average businessmen fitness value should be before terminating the algorithm. According to the termination criteria, the algorithm is stopped when the businessmen produce robot postures with end-effector positions and orientations that can reach an average distance of μF_{limit} from the task point.

In the first generations of the algorithm, the businessmen are randomly spread over the search space, and their average fitness value is high. With the progress of the algorithm and the decrease in d_{min}, the businessmen begin to converge at the locations with better fitness values, while keeping their distance. This results in the decrease in their average fitness value. Further decrease in d_{min} and discovery of better locations decreases the average fitness value of the businessmen even more. Therefore, with the termination criterion of (3.9), it is guaranteed that the algorithm stops only when the businessmen have positioned themselves in the vicinity of the solutions of the IK problem.

3.3.2 Continuous Crossover Operation

The crossover operation randomly selects two parents, P_1 and P_2, from the parent pool, and produces two children, C_1 and C_2 from them. It has been shown that for continuous search spaces, real coded GAs are more suitable than binary coded algorithms [113]. In the proposed algorithm, an SBX crossover [113] is chosen to apply the variable-by-variable crossover. The idea behind the SBX is to create a random distribution of offsprings in the domain of real numbers. This distribution matches the distribution of the common binary crossovers, that is, the SBX uses a randomly generated number, $u_\beta(i)$, to produce a random expansion ratio $\beta(i)$ that defines how similar the offsprings are to their parents, and is represented by

$$\beta(i) = \left| \frac{C_2(i) - C_1(i)}{P_2(i) - P_1(i)} \right| . \tag{3.10}$$

In order to incorporate the joint angle's mechanical limitations, the crossover is carried out as follows

1. From the n joints, l joints ($l \leq n$) are randomly selected for the crossover operation. The rest of the joint angles are transferred from the parents to the children, unchanged. In the implementation, $l = 0.5n$ [114].

2. For each of the l joint angles selected in the last step, a random number, $u_\beta(i)$, is generated. The expansion ratio, β, is then calculated by using

$$\beta(i) = \begin{cases} (2u_\beta(i))^{\frac{1}{\eta+1}} & \text{if } u_\beta(i) \leq 0.5 \\ \left(\frac{1}{2(1-u_\beta(i))}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} , \tag{3.11}$$

   where η denotes the distribution index and can be any nonnegative real number. For the small values of η, points far away from the parents, have higher probability of being chosen, whereas with large values of η, points closer to the parents are more likely to be chosen. A value of 2–5 produces a good estimate of the binary coded crossover [114]. In the proposed algorithm, η initially has a small value (equal to two). With the progress of the algorithm η increases (to around 5) so that the solutions can fine-tune to the center of each solution region.

   When the joint angles have physical limits (as commonly found in industrial robots) (3.11) must be modified to produce offsprings that are located inside the joint limits. To accomplish this, the following method is developed. First, the lower and upper bound of the

expansion ratio, β_L and β_H, are calculated from the following equation for each of the joint variables as follows

$$\begin{cases} \beta_L(i) &= \frac{0.5(P_1(i)+P_2(i))-q_{min}}{|P_1(i)-P_2(i)|} \\ \\ \beta_H(i) &= \frac{-0.5(P_1(i)+P_2(i))+q_{max}}{|P_1(i)-P_2(i)|} \end{cases} , \tag{3.12}$$

where $P_1(i)$ and $P_2(i)$ denote the i-th variable of the two parents. Since the expansion ratio of the children to parents is limited by the joint variable limits, (3.12) calculates the maximum allowable value of this parameter, corresponding to each limit. The value of u_β is then updated as

$$u_\beta(i) = \frac{u_\beta(i)}{k(i)} , \tag{3.13}$$

where

$$k(i) = \frac{1}{1 - \frac{0.5}{\beta_{limit}(i)^{\eta+1}}} , \tag{3.14}$$

and

$$\beta_{limit}(i) = \begin{cases} \beta_L(i) & \text{if } \beta_L \leq \beta_H \\ \\ \beta_H(i) & \text{otherwise} \end{cases} . \tag{3.15}$$

By changing the probability distribution of $\beta(i)$, the modification guarantees that the children are within the variable range. u_β is modified by (3.13), (3.14), and (3.15) such that the probability of choosing a β less than $\beta_{limit}(i)$, is equal to one, that is, for an arbitrary u_β, the resultant children are in the range $[q_{min}\ ,\ q_{max}]$.

Finally, $\beta(i)$ is calculated from (3.11) by using the updated $u_\beta(i)$.

3. In the last step, the children are produced from the following equation:

$$\begin{cases} C_1(i) &= 0.5[\,(1+\beta(i))\,P_1(i) + \,(1-\beta(i))\,P_2(i)\,] \\ \\ C_2(i) &= 0.5[\,(1-\beta(i))\,P_1(i) + \,(1+\beta(i))\,P_2(i)\,] \end{cases} , \tag{3.16}$$

and are then placed in the new generation population.

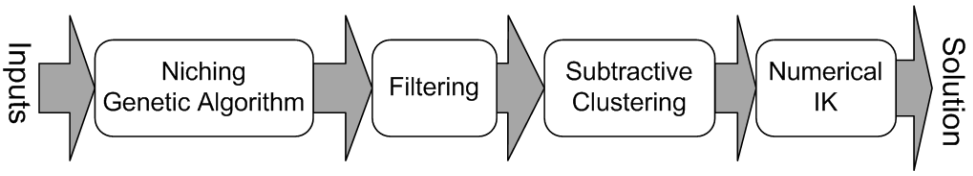


Figure 3.1: Block diagram of the proposed GA post-processing algorithm

3.4 Processing the Output

The output of the GA is a set of n dimensional vectors, representing the joint angles, with high population density around the regions with high fitness values and lower concentration in the rest of the search space. Since the local optima have higher fitness values than the regions around them, the local optima also attract a concentration of the individuals. In order to distinguish the solution regions (Global optima) from local optima, a mechanism is necessary to detect the regions with a high concentration of individuals and low error.

If the robot has 2-DOF, identifying the results in the 2D space is accomplished by observation, which is not convenient if the GA is intended to be used as a subroutine in a larger algorithm or software. For robots with more DOFs (for example, a 6-DOF PUMA), identifying these solution regions must be done in a six dimensional space, which is not possible by visual inspection. Consequently, a robust algorithm is required for clustering the results. In addition, a method is needed to increase the accuracy and resolution of the solutions for more precise applications. When the solutions converge within the desired tolerance, the local optimums are easily identified from the global ones.

In this section, the filtering, clustering, and the numerical IK post-processing stages are explained. Fig. 3.1 shows a block diagram of the proposed post processing algorithm.

3.4.1 Filtering

The fitness function of each individual is the orientation/position error from the desired value. It is easy to use the fitness function as a measure of filtering the results before the clustering.

In the filtering phase, the individuals with high fitness values (errors) are rejected and individuals with lower fitness values are transferred to the clustering step. The threshold of the filtering is represented by ε_{filter}. It should be noted that choosing a small ε_{filter} can lead to selecting fewer more accurate individuals. Also, some of the less established niches might be lost.

3.4.2 Clustering

Since no prior knowledge of the number of solutions of the IK exists, the number of solution niches or clusters is unknown. To deal with this issue, subtractive clustering [115] is employed to find the niches. Subtractive clustering is a one-pass algorithm for estimating the number of clusters and their centres for a set of data, when the number of the clusters is unknown.

In subtractive clustering, each point in the data set is a potential cluster centre. This algorithm measures the potential of each of the points, according to the density of the data set around it and assigns the point with the highest potential a cluster centre. Then, removes all the other points in the $r_{cluster}$ from the cluster centre, and repeats the process until all the points in the data set are within the radius of a cluster.

Choosing $r_{cluster}$ has a substantial effect on the number of clusters that are detected by the algorithm. The larger the $r_{cluster}$, the fewer the clusters detected. Because the GA has filtering and runs until a relatively good convergence is achieved, $r_{cluster}$ can be set to small values to detect all the solution regions with a good precision.

In the GA, all the joint angles are mapped into $[-\pi, \pi]$. If some of the solutions of the IK are close to the border limits, $-\pi$ or π, two concentrations of the individuals that represent only one solution forms close to both of the border limits. The reason is that, although the angles close to these border are at the two limits, from a mechanical point of view the angles are close to each other. As a result, the clustering stage detects two different clusters close to the borders, both belong to the same cluster. To overcome this problem, in the proposed algorithm, the clustering is applied to the joint angles, where they are mapped into $[-\pi, \pi]$. Then the solutions of the previous clustering are mapped into $[0, 2\pi]$ and undergo subtractive clustering again.

3.4.3 Numerical Improvement of the Inverse Kinematic Solutions

The outputs of the previous two stages are the centres of the niches, detected by the clustering algorithm. Although these centres represent the location of the solutions in the joint angle space, the centres might not have the required accuracy and resolution for a precision control of the robotic manipulator. As a result, the necessity of improving the accuracy of the solutions arises.

To increase the accuracy of the solutions to any desired order, a Quasi-Newton algorithm is utilized. The numerical IK uses the outputs of the clustering stage as the initial search point and then converges to within the desired positioning and orienting tolerances.

If a solution in the output of the numerical error still yields errors greater than the admitted tolerances, the solution is identified as a local optimum and eliminated from the final set of solutions. However, to clarify the achieved results, the local optima are not eliminated.

Adopting numerical methods after the GA, has the additional benefit that the GA, which is computationally demanding, can be stopped even when the niches are formed on the approximate location of the solutions, that is, the GA can be stopped earlier, and the responsibility of further convergence to the solutions can be transferred to the much faster numerical algorithm. Consequently, the overall speed of the convergence of the IK algorithm is enhanced.

3.5 Results

Four distinct KCs, each with different number of joints, are applied for validating the proposed algorithm.

In the rest of this section, the results of running the algorithm for the tested KCs are presented. First, the results of running the algorithm for an $\mathbf{m}_3$ manipulator to visualize the output of each post processing stage is presented. Then, the performance of the algorithm in finding all the IK solutions for an $\mathbf{m}_4$ and an $\mathbf{m}_6$ PUMA robot for two different task points are analyzed. Finally, the solutions that are found by the algorithm for an $\mathbf{m}_7$ robot are given. The results are achieved on Matlab on a 2GHz AMD64 processor with 1.5GB of RAM.

The GA is a stochastic method such that numerous runs of the algorithm are required to examine the performance and the repeatability of the algorithm. Consequently, for each of the selected task points, the algorithm is executed five times. The parameters that are used in the algorithm remain fixed for all the runs and all of the configurations. The parameters are summarized in Table 3.2. It is noteworthy that the expression for the population size of the businessmen and the customers represents an approximate upper bound. Therefore, the algorithm appears to perform well, even with smaller sizes of the population. In the table, n represents the DOF of the robot.

3.5.1 3-DOF Planar Robot

Fig.3.2 depicts the $\mathbf{m}_3$ planar robot when a task point is reached with two distinct IK solutions. The matrix representation of the tested 3-DOF manipulator is

$$\mathbf{m}_3 = \begin{bmatrix} 0 & 0 & 60 \\ 0 & PR & 30 \\ 0 & P & 30 \\ 0 & P & 30 \end{bmatrix}. \tag{3.17}$$

Table 3.3 summarizes the results of running the algorithm for the tested 3-DOF KC. In the table, N_{GA} and N_{NM} are the number of solutions before and after the numerical IK, respectively.

Table 3.2: Parameters used in the Genetic Algorithm

Parameter	Value
Businessman Population (b)	$2n.2^{n-2}$
Customer Population (c)	$2n.b$
κ	1.2
λ	0.5
θ_{min}	$-\pi$
θ_{max}	π
ψ	π
η	2-5
$r_{cluster}$	0.0796
ε_{filter}	μF_{limit}
n_{limit}	$3.b$
n_{tour}	2
μ	0.5
g_{max}	500

(a) (b)

Figure 3.2: Solutions of the $\mathbf{m}_3$ planar KC

Table 3.3: Output of the algorithm for the $\mathbf{m}_3$ planar. N represents the number of niches. E_P, E_O, and $\mathbf{E}_J$ represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	7	11	7	6	5
N_{NM}	2	2	2	2	2
$\mathbf{E}_P^{GA}(cm)$	0.1490	0.0997	0.4320	4.0835	0.5998
$\mathbf{E}_O^{GA}(degree)$	1.4957	0.0197	0.0716	4.7478	0.0231
$\mathbf{E}_P^{NM}(cm)$	0.0001	0.0001	0.0548	0.0002	0.0214
$\mathbf{E}_O^{NM}(degree)$	0.0000	0.0000	0.1496	0.0000	0.0470

$\mathbf{E}_P^{GA}$ and $\mathbf{E}_P^{NM}$ are the minimum positioning error (cm) before and after the numerical IK, and $\mathbf{E}_O^{GA}$ and $\mathbf{E}_O^{NM}$ are the minimum orientation error (degrees) before and after the numerical IK, respectively.

The algorithm is stopped in generation 484, when the termination criteria is satisfied. As can be seen from the table, the algorithm can find the two solutions in all of the runs. It should be noted that the number of clusters are higher before entering the numerical algorithm phase. The reason is that the algorithm is stopped before all the niches converge at the exact solution locations. By doing so, the GA is interrupted earlier and further convergence is demanded from the numerical method in order to achieve a better computational efficiency. If the GA is allowed to run for more generations, the number of niches found after the clustering phase are likely to be closer to the number of niches after the numerical method's phase.

To further investigate the effect of the post processing phase, the joint angles are used to visualize the outputs of each stage. Fig.3.3(a) reflects the output of the niching GA, which consists of the customer and businessman populations. At this stage, the individuals are scattered around the entire search space with high concentrations around the solution locations. Furthermore, the population of the businessmen is concentrated around the solution locations, while keeping the distance d_{min} from each other. As shown in Fig.3.3(b), only those from the original population that have fitness values better than the threshold of filtering remain after the filtering phase. Clustering phase is then applied to the filtered population to find the niche centres. A numerical method further converges the solutions of the clustering phase to the actual solutions. The output of the clustering and the numerical method is shown in Fig.3.3(c). Here, more than one niche can converge to the same numerical solution.

It is evident that the post processing stage accepts the original population of the GA and then extracts the solutions of the IK with the required degree of accuracy.

3.5.2 4-DOF Spatial Manipulator

The tested $\mathbf{m}_4$ KC has two distinct IK solutions for any reachable task point. The 4-DOF manipulator is a planar manipulator, attached to a rotational joint at the base to produce spatial movements. The matrix representation of the manipulator is as

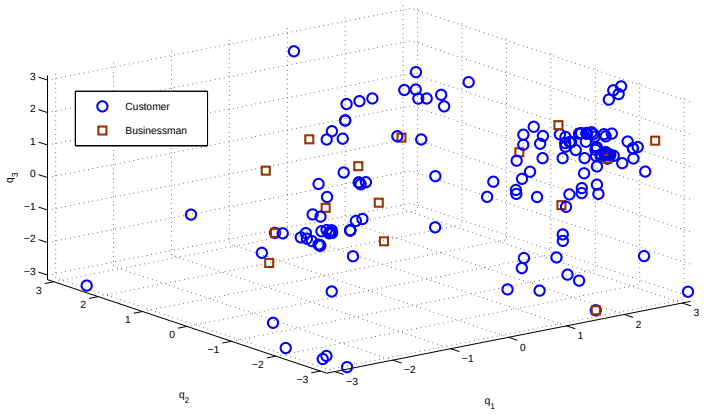
$$\mathbf{m}_4 = \begin{bmatrix} 0 & 0 & 90 \\ 0 & R & 45 \\ 0 & P & 40 \\ 0 & P & 40 \\ 0 & P & 30 \end{bmatrix} . \tag{3.18}$$

Fig.3.4(a) and Fig.3.4(b) show $\mathbf{m}_4$ when it reaches the first analyzed task point with two distinct IK solutions, and Fig.3.4(c) and Fig.3.4(d) portray the distinct solutions for the second task point. Table 3.4 and Table 3.5 lists the results of 5 runs of the algorithm for each of the task points. $\mathbf{E}^J$ is the per joint difference of the joint angles before the numerical IK to the respective value after applying the numerical method and reaching a solution. t is the run time of the algorithm for each experiment.

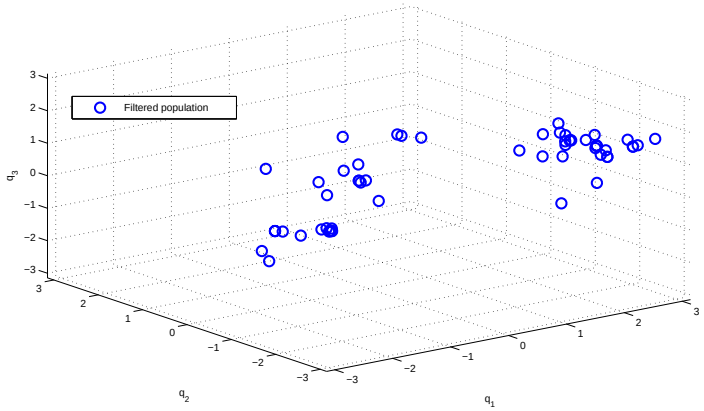
For the first task point, the algorithm runs for an average of 82 iterations and 40 seconds before convergence. The corresponding values for the second task point are 24 iterations and 19 seconds. In all of the runs, all of the solutions of the IK are found. Typically, the GA stage ends with more niches than the actual number of the solutions. After the numerical method is applied to the solutions, more than one of the niche centres converge at the same solution. As a result, at the output of the numerical stage, the final number of the solutions is typically less than the number of niche centres.

3.5.3 6-DOF PUMA

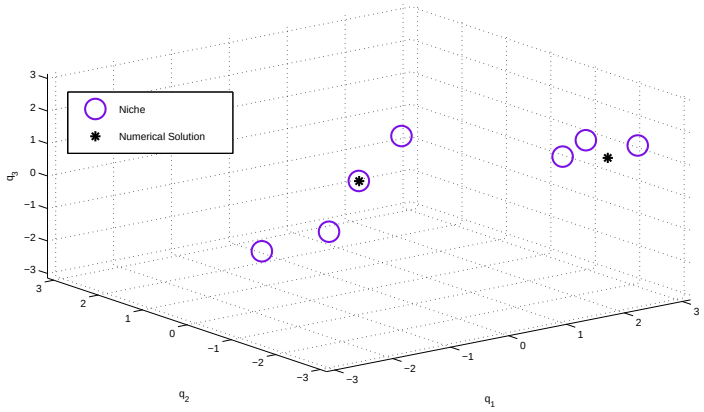
An $\mathbf{m}_6$ manipulator, resembling the KC of a PUMA manipulator is chosen as the next test bed of the algorithm. $\mathbf{m}_6$ is represented by the following



(a)

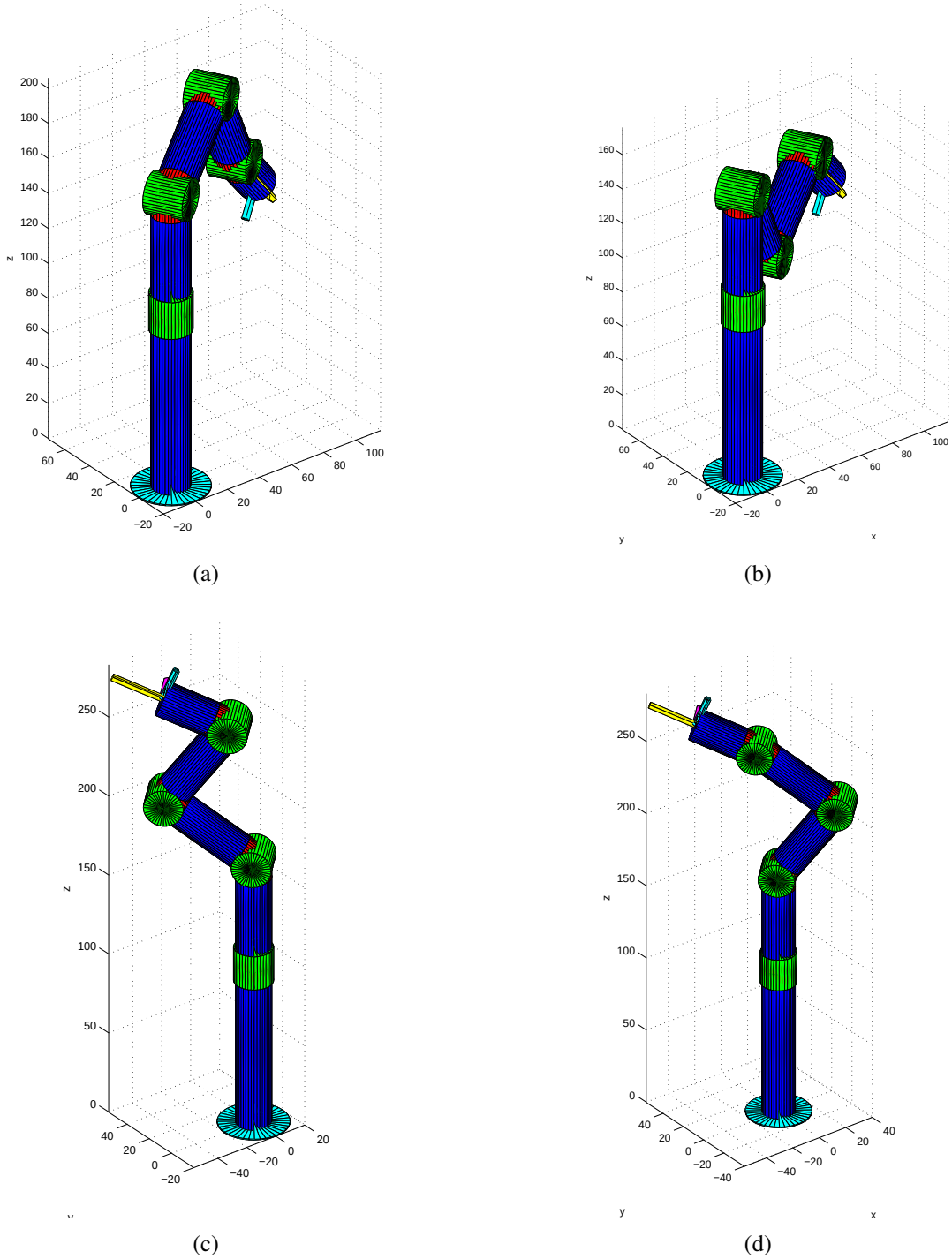


(b)



(c)

Figure 3.3: Output of the post processing stages for the $\mathbf{m}_3$ planar KC: (a) customer and businessman populations; (b) filtered population; and, (c) detected niches and numerical algorithm solutions



(a) (b)

(c) (d)

Figure 3.4: Solutions of the $\mathbf{m}_4$ spatial KC: (a) and (b) IK solutions for reaching task point 1; (c) and (d) IK solutions for reaching task point 2

Table 3.4: Output of the algorithm in reaching task point 1 for the $\mathbf{m}_4$ spatial. N represents the number of niches. E_P, E_O, and $\mathbf{E}_J$ represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	18	30	15	17	21
N_{NM}	2	2	2	2	2
$\mathbf{E}_P^{GA}(cm)$	1.2184	10.4490	12.5175	15.5401	23.3084
$\mathbf{E}_O^{GA}(degree)$	1.9098	0.4613	0.3554	1.0181	1.9905
$\mathbf{E}_P^{NM}(cm)$	0.0000	0.0001	0.0001	0.0001	0.0000
$\mathbf{E}_O^{NM}(degree)$	0.0001	0.0001	0.0001	0.0000	0.0000
$\mathbf{E}^J(degree)$	3.7007	3.5954	3.2532	1.9500	7.1817
$t(seconds)$	39	30	45	36	48

Table 3.5: Output of the algorithm in reaching task point 2 for the $\mathbf{m}_4$ spatial. N represents the number of niches. E_P, E_O, and $\mathbf{E}_J$ represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	47	17	30	19	10
N_{NM}	2	2	2	2	2
$\mathbf{E}_P^{GA}(cm)$	17.6113	7.0731	4.0406	1.0563	34.3000
$\mathbf{E}_O^{GA}(degree)$	0.6621	5.0517	0.5007	2.3897	1.3140
$\mathbf{E}_P^{NM}(cm)$	0.0002	0.0001	0.0002	0.0001	0.0001
$\mathbf{E}_O^{NM}(degree)$	0.0002	0.0001	0.0003	0.0001	0.0000
$\mathbf{E}^J(degree)$	7.8336	3.8397	3.6875	4.8561	10.3207
$t(seconds)$	18	14	21	27	16

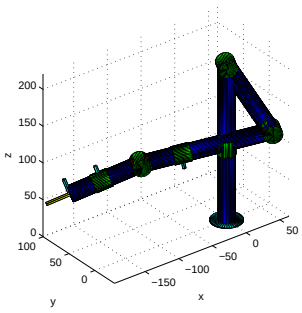
Table 3.6: Output of the algorithm in reaching task point 1 for the $\mathbf{m}_6$ PUMA. N represents the number of niches. E_P, E_O, and $\mathbf{E}_J$ represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	216	214	214	219	229
N_{NM}	8	9	8	8	8
$\mathbf{E}_P^{GA}(cm)$	6.2119	9.4829	9.3211	7.6261	12.0384
$\mathbf{E}_O^{GA}(degree)$	3.3087	3.9755	3.9423	3.4848	1.8421
$\mathbf{E}_P^{NM}(cm)$	0.0001	0.0000	0.0000	0.0001	0.0000
$\mathbf{E}_O^{NM}(degree)$	0.0000	0.0001	0.0001	0.0001	0.0000
$\mathbf{E}^J(degree)$	7.2211	4.7227	7.6433	9.7832	5.5340
$t(seconds)$	1638	2240	2295	3263	1536

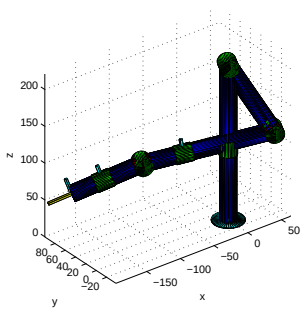
$$\mathbf{m}_6 = \begin{bmatrix} 0 & 0 & 90 \\ 0 & R & 90 \\ 0 & P & 90 \\ 0 & P & 90 \\ 0 & R & 30 \\ 0 & P & 30 \\ 0 & R & 30 \end{bmatrix}. \tag{3.19}$$

Fig.3.5 and Fig.3.6 display all the solutions of the IK, when the robot reaches the two task points. In the figures, visual handles are attached to links connected to the output of joints 1, 4 and 6 to differentiate between the two possible values of joint angles q_1, q_4 and q_6 in producing multiple IK solutions. As observed in Fig.3.5, since the only reason for running the algorithm is to find multiple solutions of the IK, all of the solutions of IK, regardless of self collision, are shown. The results of five runs of the algorithm are summarized in Table 3.6 and Table 3.7. The algorithm achieves these results after 61 and 86 iterations (36 and 42 minutes) for task points 1 and 2, respectively.

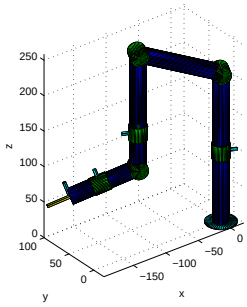
For the first task point, although the correct number of the IK solutions is eight, the algorithm reports nine distinct solutions in one of the runs. Further investigation reveals that two of the solutions differ only in the third joint value, which is $\theta_3 = 0$ in one of the solutions and $\theta_3 = 2\pi$ for the other solution.



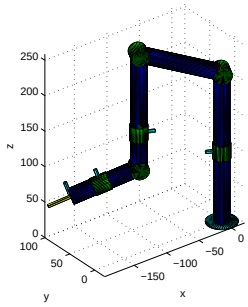
(a)



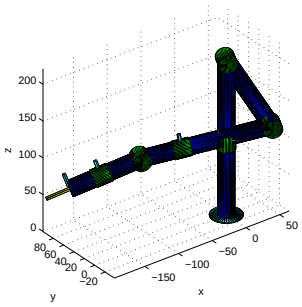
(b)



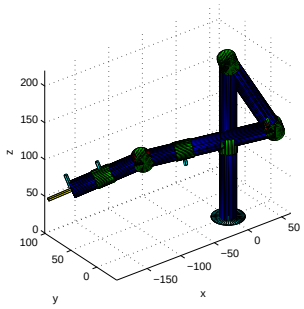
(c)



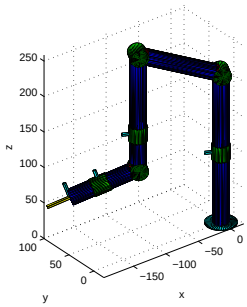
(d)



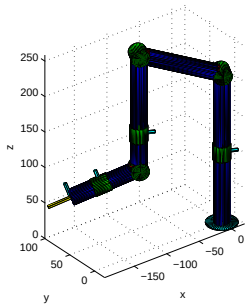
(e)



(f)



(g)



(h)

Figure 3.5: Solutions of the $\mathbf{m}_6$ PUMA KC for task point 1

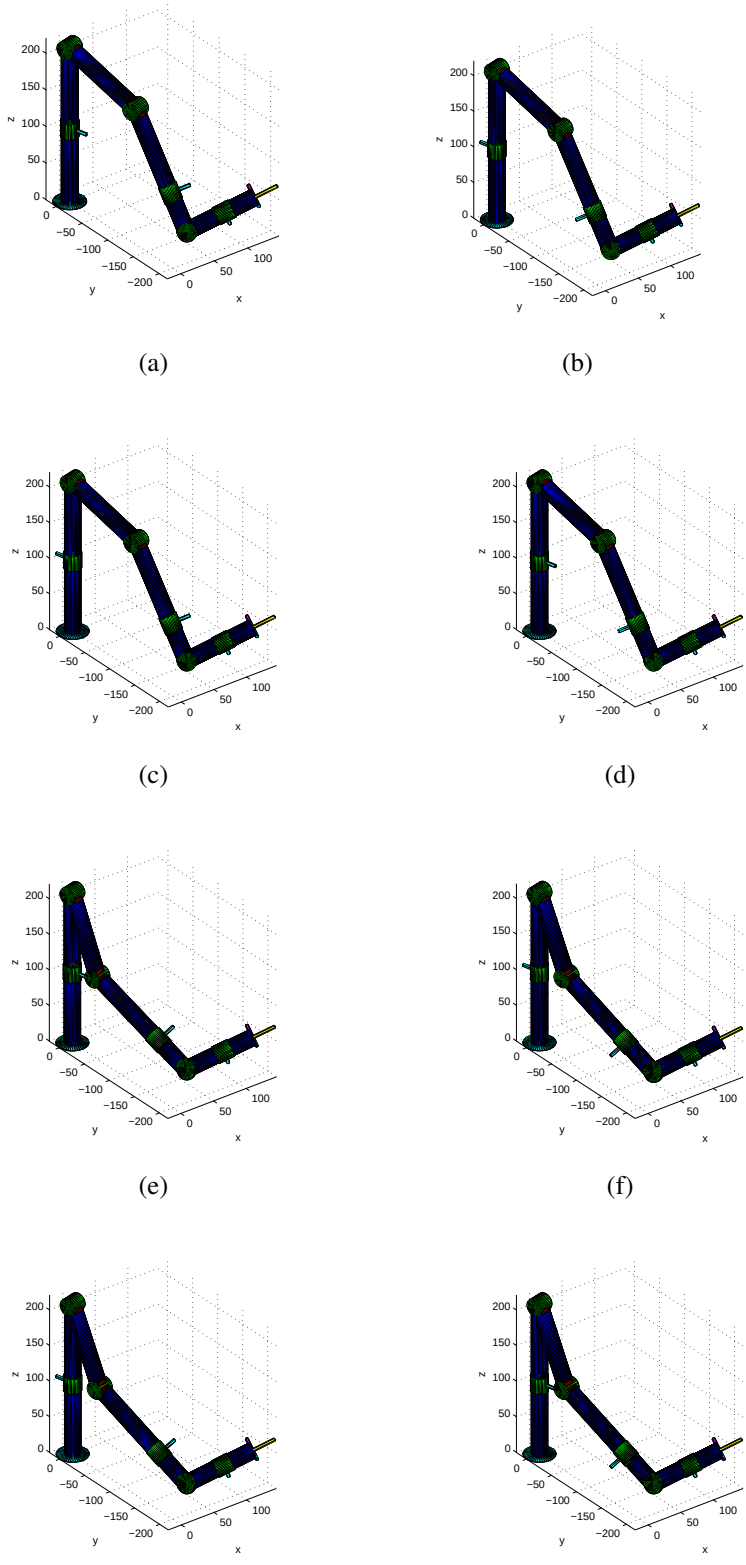


Figure 3.6: Solutions of the $\mathbf{m}_6$ PUMA KC for task point 2

Table 3.7: Output of the algorithm in reaching task point 2 for the 6-DOF PUMA. N represents the number of niches. E_P, E_O, and $\mathbf{E}_J$ represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	107	154	146	59	124
N_{NM}	8	8	8	8	8
$\mathbf{E}_P^{GA}(cm)$	3.9693	3.6407	3.1639	4.1705	4.8595
$\mathbf{E}_O^{GA}(degree)$	2.4588	6.8048	3.8925	1.7206	3.0407
$\mathbf{E}_P^{NM}(cm)$	0.0000	0.0006	0.0000	0.0001	0.0004
$\mathbf{E}_O^{NM}(degree)$	0.0000	0.0001	0.0001	0.0001	0.0001
$\mathbf{E}^J(degree)$	1.4597	3.9952	4.7653	5.8982	6.4299
$t(seconds)$	5421	5453	5276	5381	5695

3.5.4 7-DOF Spatial Robot

An $\mathbf{m}_7$ spatial manipulator is chosen for further verification of the algorithm in finding multiple solutions of the IK. The matrix representation of the manipulator is as follows:

$$\mathbf{m}_7 = \begin{bmatrix} 0 & 0 & 20 \\ 0 & R & 0 \\ 0 & P & 0 \\ \pi/2 & P & 200 \\ 0 & P & 200 \\ 0 & P & 0 \\ \pi/2 & P & 0 \\ 0 & R & 20 \end{bmatrix}. \tag{3.20}$$

In Fig.3.7, it is evident that the KC of the robot closely resembles that of the Canada Arm 2 of the International Space Station. The length of the links 1, 2, 5 and 6, are assumed to be zero, indicating that the joints are directly connected without any links between them. The algorithm achieves the termination criteria in iteration 530, after running for about 42 hours. The algorithm converges slowly in this example due to the fact that the 7-DOF spatial manipulator is a redundant robot. Thus, the objective function has numerous local and global optima. In addition, the complexity of performing a search in the solution space of an $\mathbf{m}_7$ manipulator

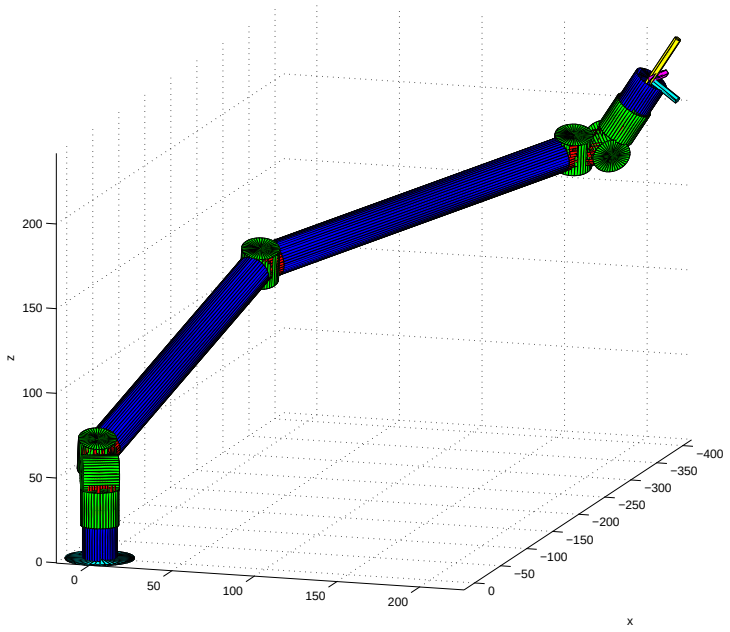


Figure 3.7: $\mathbf{m}_7$ spatial robot

Table 3.8: Output of the algorithm in reaching the task point for the $\mathbf{m}_7$ Spatial Robot. N represents the number of niches. E_P, E_O, and $\mathbf{E}_J$ represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm.

N_{GA}	N_{NM}	$\mathbf{E}_P^{GA}(cm)$	$\mathbf{E}_O^{GA}(deg)$	$\mathbf{E}_P^{NM}(cm)$	$\mathbf{E}_O^{NM}(deg)$	$\mathbf{E}^J(deg)$	$t(hour)$
69	28	2.87	1.33	0.0003	0.0000	7.79	42

increases exponentially, compared to that of the $\mathbf{m}_6$ system of the previous example.

The output of the algorithm after the numerical stage consists of 28 niches. From these 28 niches, 16 were local solutions of the problem, and 12 are the global and correct solutions of the IK. The summary of the results of the run for the $\mathbf{m}_7$ KC is provided in Table 3.8.

3.6 Summary

In this chapter, an enhanced version of an adaptive niching strategy is proposed and applied to extract multiple solutions of the IK problem for spatial robots. Since this algorithm uses few preset parameters, it can be generalized to solve the IK of a robot with an unknown number of DOF and kinematic configuration. The algorithm incorporates real coding, adaptive minimum

distance for businessmen, elitism, and adaptive real coding distribution index. To process the results, a subtractive clustering algorithm is chosen. It is demonstrated that the algorithm performs with superior precision for random task points, independent of the KC of the robot.

By using the GA niching algorithm in conjunction with a numerical method, the resolution and precision of the results are significantly improved. The niche centres that are detected in the GA are applied in the numerical method as the initial search points to achieve solutions with the required level of precision.

The convergence speed of the algorithm , especially for 6-DOF manipulators, can prove to be an obstacle in using the algorithm effectively in the practical applications. Although the proposed method is slower than some multi-solution IK solvers, it is more general, that is, a wide range of constraints can be used, and the algorithm provides a solution which is still valid [116].

In the next chapter, a multi-solution IK solver is proposed which is faster than the niching GA IK solver.

Chapter 4

A Multi-Solution Inverse Kinematics Solver Based on the Joint Reflection Operator

4.1 Introduction

In this chapter, a method is proposed to efficiently obtain multiple solutions of the IK problem. The method is developed for non-redundant MRRs, yet, the approach is generic enough to be used for all $\mathbf{m}_n$, where $n = 1 \cdots 6$. The algorithm requires no symbolic or analytical modifications, when the robot configuration changes. Also, the novel algorithm is proven to be computationally efficient and fast.

The approach is a generalization of numerical IK solvers, designed to incorporate the results of analytical methods for $\mathbf{m}_2$ manipulators to obtain multiple solutions of the IK problem. The solver operates by finding all the positioning solutions and utilizing these solutions to determine the desired positioning/orienting IK solutions. The performance of the algorithm is verified by solving for multiple solutions of the IK problem for three distinct 6-DOF KCs. For one of the tested KCs, no closed-form IK solutions exists.

The proposed method has the following features.

- The proposed algorithm can be used for all the possible KCs, which can be assembled from the MRR standard modular joint set.
- The algorithm obtains all the solutions of the IK problem.
- The only requirement of the algorithm are the FK equations of the manipulator. No prior knowledge of the number and/or proximity of the solutions is required.

- No symbolic or analytical modification is needed when the algorithm is applied for solving the IK problem of a new KC.
- The proposed algorithm is proven to be computationally more efficient than some of the existing IK solvers in the literature.

This chapter is divided into four sections. Section 4.2 provides the background for the rest of chapter. In Section 4.3, the Joint Reflection Operator, the cornerstone of the proposed method, is explained. In Section 4.4, the results of applying the proposed algorithm to three 6-DOF manipulators are presented. Section 4.5 summarizes the findings of this chapter.

4.2 Background

4.2.1 Approach

According to (2.36) and (2.37), a manipulator pose Θ^s is an IK solution, if and only if the following conditions are satisfied:

$$\mathbf{E}_P\left(\Theta^s, T_{des}\right) = 0\,, \tag{4.1}$$

and

$$\mathbf{E}_O\left(\Theta^s, T_{des}\right) = 0\,. \tag{4.2}$$

A set which includes all the IK solutions of a certain n-DOF manipulator, $\mathbf{m}_n$, for a specific task are represented by $\mathbb{Q}_s\left(\mathbf{m}_n\right)$. If the set of all the manipulator poses, in which $\mathbf{E}_p(\Theta) = 0$ and its counterpart, in which $\mathbf{E}_o(\Theta) = 0$ are defined as $\mathbb{Q}_p\left(\mathbf{m}_n\right)$ and $\mathbb{Q}_o\left(\mathbf{m}_n\right)$ respectively, $\mathbb{Q}_s\left(\mathbf{m}_n\right)$ can be obtained by $\mathbb{Q}_s\left(\mathbf{m}_n\right) = \mathbb{Q}_p\left(\mathbf{m}_n\right) \cap \mathbb{Q}_o\left(\mathbf{m}_n\right)$, by solving such that

$$\mathbb{Q}_p\left(\mathbf{m}_n\right) = \{\Theta | \mathbf{E}_P\left(\mathbf{m}_n, \Theta, T_{des}\right) = 0\} \tag{4.3}$$

and

$$\mathbb{Q}_o\left(\mathbf{m}_n\right) = \{\Theta | \mathbf{E}_O\left(\mathbf{m}_n, \Theta, T_{des}\right) = 0\} \tag{4.4}$$

and

$$\mathbb{Q}_s\left(\mathbf{m}_n\right) = \{\Theta | \Theta \in \mathbb{Q}_P \ \wedge \ \Theta \in \mathbb{Q}_O\} \tag{4.5}$$

To find all the solutions to the IK problem, the approach of the Joint Reflection Operator-based method is to find all the members of $\mathbb{Q}_p$. Since they all satisfy the condition of (4.3), they can be examined for satisfying (4.4). The set of joint angles which satisfies both (4.3) and (4.4) then constitutes $\mathbb{Q}_s$.

Consequently, the proposed method is used to find all the members of the set in (4.3). By using (2.34), each of the solutions are then used as the initial search point for a numerical algorithm that finds $\arg\min F_{obj}\left(\Theta, T_{des}\right)$. If a Θ which minimizes $F_{obj}\left(\Theta, T_{des}\right)$ has $\mathbf{E}_P$ and $\mathbf{E}_O$ within an acceptable tolerance from zero, Θ is added to $\mathbb{Q}_s$.

The goal of the newly developed algorithm in this chapter is to find all the solutions of the positioning and orienting IK for serial multi-DOF robot manipulators $\mathbb{M}_n$. It is assumed that by using any current IK solver, one solution of the IK, Θ^s, is found. The proposed algorithm uses Θ^s as an initial point for finding the remaining IK solutions.

4.2.2 Classification of the $\mathbf{m}_2$ Manipulators

Since the cornerstone of the proposed method is reducing $\mathbf{m}_n$ to $\mathbf{m}_2$ manipulators, the $\mathbf{m}_2$ KCs are examined in more detail. In Fig.4.1(a), the joints of an $\mathbf{m}_2$ manipulator are illustrated. z_1, z_2, are the axes of rotation of the joints, and P_{12} is the vector representing the relative position of the two joints in the Cartesian space. n_1 and n_2 are the common normals of vectors z_1 and P_{12}, and z_2 and P_{12}, respectively, such that

$$\begin{cases} n_1 = P_{12} \times z_1 \\ n_2 = z_2 \times P_{12} \end{cases} . \tag{4.6}$$

According to the relative position and orientation of the joint axes, z_1 and z_2, 2-DOF manipulators of $\mathbb{M}_2$ are grouped into three distinct classes.

Class 1: Intersecting Joint Axes: This class is represented by $\mathbb{M}_{\times,2}$. In the members of this class, n_1 and n_2 are parallel, whereas a non-zero angle exists between z_1 and z_2. A sample manipulator for this case together with the joint axes and corresponding norms, are shown in Fig. 4.1(b). The condition that all the members of this class should have is written as

$$\mathbb{M}_{\times,2} = \{\ \mathbf{m}_2 \mid \ (n_1 \times n_2 = 0) \ \wedge \ (z_1 \times z_2 \neq 0)\} \ . \tag{4.7}$$

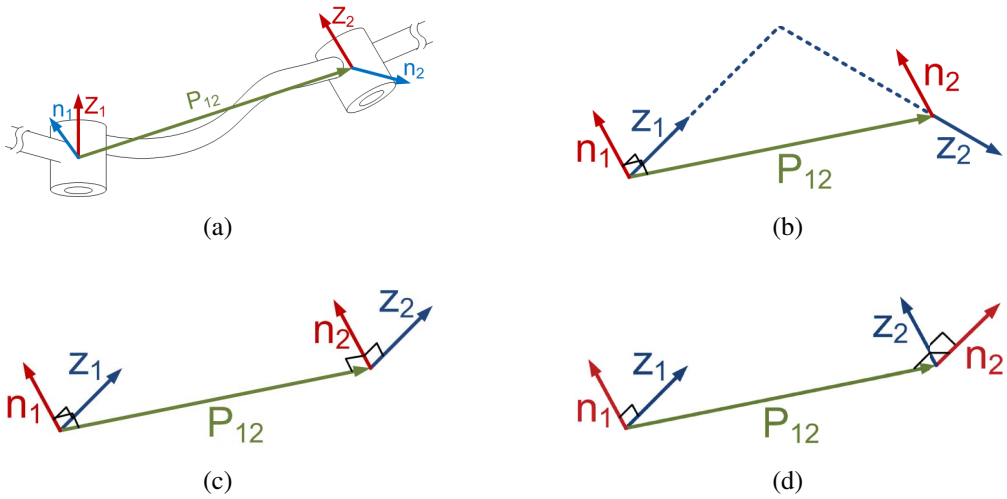


Figure 4.1: Relative position and orientation of joint axes of a $\mathbf{m}_2$: (a) $\mathbf{m}_2 \in \mathbb{M}_2$; (b) $\mathbf{m}_2 \in \mathbb{M}_{\times,2}$; (c) $\mathbf{m}_2 \in \mathbb{M}_{\shortparallel,2}$; and, (d) $\mathbf{m}_2 \in \mathbb{M}_{\shortparallel\!\vdash,2}$

Table 4.1: Denavit-Hartenberg parameters of a 2-DOF manipulator with two intersecting joints

Joint	α_i	a_i	d_i	θ_i
1	α_1	0	0	θ_1
2	0	l_2	d_2	θ_2

According to the modified DH convention, when two joint axes are intersecting, the origin of the link frames should be moved to the point of intersection, i.e, $a_1 = 0$, $d_1 = 0$, and $\alpha_2 = 0$. Table 4.1 shows the DH parameters of a manipulator in this class. Fig. 4.2 shows all the configurations of $\mathbb{M}_{\times,2}$.

Class 2: Parallel Joint Axes: $\mathbb{M}_{\shortparallel,2}$ represents this class of 2-DOF manipulators. The joint axes and the corresponding norms of a member of $\mathbb{M}_{\shortparallel,2}$ are illustrated in Fig.4.1(c). For a 2-DOF manipulator to be a member of this group, in addition to n_1 and n_2 being parallel, z_1 and z_2 should also be parallel. Consequently, the condition that all the members of this class should have is written as

$$\mathbb{M}_{\shortparallel,2} = \{\, \mathbf{m}_2 \mid z_1 \times z_2 = 0 \,\}. \tag{4.8}$$

For all the manipulators of this class, $\alpha_1 = 0$, $d_1 = 0$ and $a_1 \neq 0$. The DH parameters of the members of $\mathbb{M}_{\shortparallel,2}$ are listed in Table 4.2. In Fig.4.3, the configurations in this group are illustrated.

Class 3: Non-Intersecting and Non-Parallel Joint Axes: This class is represented by $\mathbb{M}_{\shortparallel\!\vdash,2}$. The joint axes and the corresponding norms of a member of $\mathbb{M}_{\shortparallel\!\vdash,2}$ are shown in Fig.4.1(d).

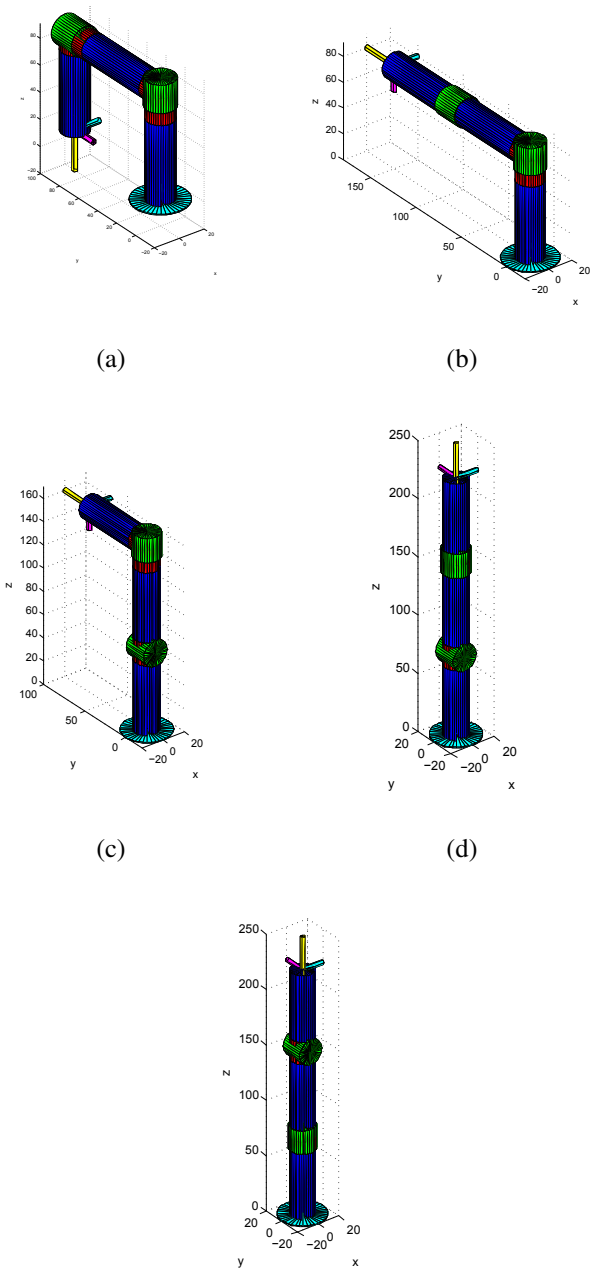
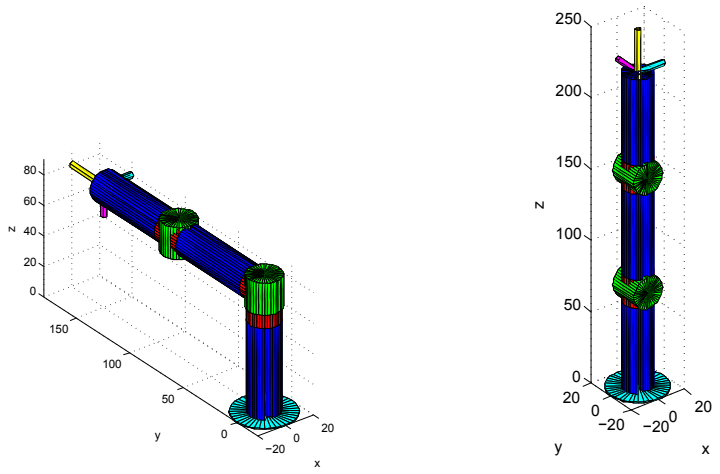


Figure 4.2: Non-redundant 2-DOF manipulators with intersecting joint axes which can be assembled from the considered joint types

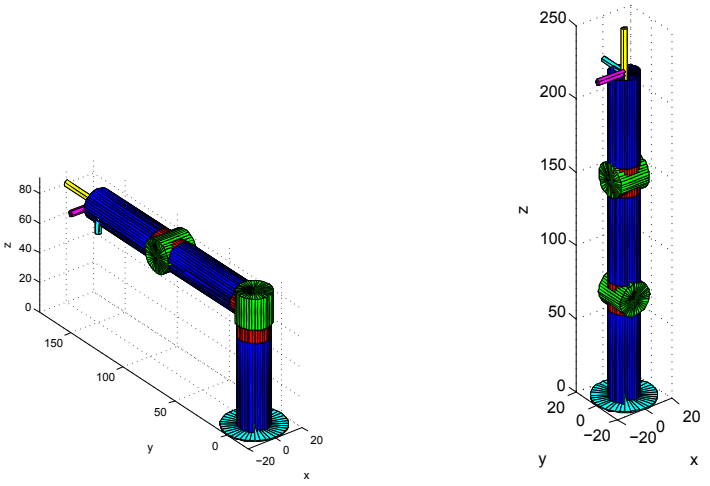


(a) (b)

Figure 4.3: Non-redundant 2-DOF manipulators with parallel joint axes which can be assembled from the considered joint types

Table 4.2: Denavit-Hartenberg parameters of a 2-DOF manipulator with two parallel joints

Joint	α_i	a_i	d_i	θ_i
1	0	l_1	0	θ_1
2	0	l_2	0	θ_2



(a) (b)

Figure 4.4: Non-redundant 2-DOF manipulators with no parallel nor intersecting joint axes which can be assembled from the considered joint types

Table 4.3: Denavit-Hartenberg parameters of a 2-DOF manipulator with joint axes which are not intersecting nor parallel

Joint	α_i	a_i	d_i	θ_i
1	$\frac{\pi}{2}$	0	d_1	θ_1
2	0	l_2	0	θ_2

For a 2-DOF manipulator to be a member of this group, n_1 and z_2 or n_2 and z_1 should be parallel. Therefore, the condition that all the members of this class is written as

$$\mathbb{M}_{\Vdash,2} = \{\, \mathbf{m}_2 \mid (n_1 \times n_2 \neq 0) \wedge (z_1 \times z_2 \neq 0) \} \ . \tag{4.9}$$

Fig.4.4 exhibits all the manipulators of this class. The joint axes in the manipulators of this class lie in two parallel planes, but an angle $\alpha_1 \neq 0$ is between them. It is observed from Fig.4.4, that for all the members of this class, the joint axes are perpendicular. Therefore, only the case, in which $\alpha_1 = \frac{\pi}{2}$ is considered. As a result, the manipulators of this group share these DH parameters: $\alpha_1 = \frac{\pi}{2}$, $\alpha_2 = 0$, and $d_2 = 0$. The complete list of the DH parameters of manipulators of this group is shown in Table 4.3.

4.3 Joint Reflection Operator

In this section, the *Joint Reflection Operator (JRO)*, which is the basis of the proposed multi-solution IK solver, is explained. First, the approach to find all the members of $\mathbb{Q}_p$ for $\mathbb{M}_2$, $\mathbb{M}_3$, and $\mathbb{M}_n$ manipulators is explained. Then, the approach is generalized to find the members of $\mathbb{Q}_s$, that is, the solutions to the combined positioning/orienting IK problem.

4.3.1 Finding Multiple Positioning IK Solutions

Finding Multiple Positioning IK Solutions for $\mathbb{M}_2$ Manipulators

Assume a 2-DOF manipulator with the link frames assigned according to the DH convention. According to (2.22), the Homogenous Transformation of the end-effector T_{ee}, with respect to the reference frame at the base of the robot, is written as the following:

$$\begin{aligned} \mathbf{T}_{ee}\left(\theta_1,\ \theta_2\right) &= \left(\begin{array}{c|c} \mathbf{R}_{ee}(\theta_1,\theta_2) & \mathbf{P}_{ee}(\theta_1,\theta_2) \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right) \\ &= \mathbf{T}_{0,1}\left(\theta_1\right).\mathbf{T}_{1,2}\left(\theta_2\right)\ . \end{aligned} \tag{4.10}$$

Since the DH parameters are chosen according to the modified DH convention, the last frames of the manipulator (the frame of the end-effector) are always chosen such that $\alpha_2 = 0$. If the Homogenous Transformation of the desired frame is represented by (2.26), in which the desired end-effector position of the task $\mathbf{P}_{des}$ is expressed as

$$\mathbf{P}_{des} = \left[\begin{array}{c} P_x \\ P_y \\ P_z \end{array}\right]\ . \tag{4.11}$$

Then, the set of the solutions of the positioning problem $\mathbb{Q}_P$ is written as:

$$\mathbb{Q}_P = \{\Theta = [\theta_1\ \ \theta_2] \mid \mathbf{P}_{ee}(\theta_1,\theta_2) = \mathbf{P}_{des}\}\ . \tag{4.12}$$

As a result, to find the members of $\mathbb{Q}_P$, the following equation should be solved for $[\theta_1\ \ \theta_2]$ as follows

$$\mathbf{P}_{ee}(\theta_1, \theta_2) = \mathbf{P}_{des} \, . \tag{4.13}$$

For all the manipulators of $\mathbb{M}_2$, an operator, called the *Joint Reflection Operator (JRO)* represented by Ψ_{JRO}, can be found such that an unknown positioning solution $\Theta^* = [\theta_1^* \ \theta_2^*]$ can be reached from a known one $\Theta^s = [\theta_1^s \ \theta_2^s]$ through $\Theta^* = \Psi_{JRO}(\Theta^s)$. Ψ_{JRO} is applied in two steps:

1. θ_i^*, the angle of one of the joints of Θ^* is determined by linear operator $\kappa(\Theta^s)$ from the known solution Θ^s by computing

$$\theta_i^* = \kappa(\Theta^s) = a \cdot \theta_i^s + b \, . \tag{4.14}$$

2. A θ_j^* is obtained by minimizing $\mathbf{E}_P$ when θ_i has a constant value equal to θ_i^* as follows

$$\theta_j^* = \arg\min_{\theta_j} \ \mathbf{E}_p(\theta_1, \theta_2)|_{\theta_i = \theta_i^*} \, . \tag{4.15}$$

The minimization problem for finding θ_j^* is solved with any of the existing minimization algorithms such as Gradient descent, pattern search, or GAs. In this research, a pattern search algorithm is used for this stage [117, 118].

The existence of the linear operator (4.14) is proven in Theorem.1. The uniqueness of the solution of (4.15) is confirmed in Theorem.2.

Theorem 1. Existence of the linear operator κ: *Given a manipulator $\mathbf{m}_2 \in \mathbb{M}_2$, a known solution of the IK problem Θ^s, a linear operator $\kappa(\Theta^s)$ exists such that:*
I. $\theta_i^* = \kappa(\Theta^s) = a \cdot \theta_i^s + b$.
II. $\theta_i^* = \kappa(\Theta^s) \quad \Leftrightarrow \quad \theta_i^s = \kappa(\Theta^*)$.
where θ_i^* is the angle of one the joints of another solution Θ^*.

Proof. To prove the theorem, (4.13) is solved in each of the three defined classes of $\mathbb{M}_2$ for θ_1 and θ_2 to identify the corresponding operator $\kappa(\Theta^s)$.

Class 1: Intersecting Joint Axes: The result of solving (4.13) for the manipulators of this group is written as the following two solution sets:

$$
\begin{aligned}
\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}_1 &= \begin{bmatrix} \arctan\left(\frac{\sqrt{-{d_2}^2+{P_x}^2+{P_y}^2}P_y+P_x\,d_2}{{P_x}^2+{P_y}^2}, \frac{\sqrt{-{d_2}^2+{P_x}^2+{P_y}^2}P_x-P_y\,d_2}{{P_x}^2+{P_y}^2}\right) \\ \arctan\left(\frac{P_z}{a_2}, \frac{\sqrt{P_x^2+P_y^2-d_2^2}}{a_2}\right) \end{bmatrix}, \\
&= \begin{bmatrix} f_1^{\times}(\mathbf{P}_{des}) \\ f_2^{\times}(\mathbf{P}_{des}) \end{bmatrix} \\
\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}_2 &= \begin{bmatrix} \arctan\left(-\frac{\sqrt{-{d_2}^2+{P_x}^2+{P_y}^2}P_y-d_2P_x}{{P_x}^2+{P_y}^2}, -\frac{P_y\,d_2+\sqrt{-{d_2}^2+{P_x}^2+{P_y}^2}P_x}{{P_x}^2+{P_y}^2}\right) \\ \pi-\arctan\left(\frac{P_z}{a_2}, \frac{\sqrt{P_x^2+P_y^2-d_2^2}}{a_2}\right) \end{bmatrix} \\
&= \begin{bmatrix} f_3^{\times}(\mathbf{P}_{des}) \\ \pi - f_2^{\times}(\mathbf{P}_{des}) \end{bmatrix}. \qquad (4.16)
\end{aligned}
$$

Table 4.4 summarizes the solutions in (4.16) for the manipulators of this class. The non-linear trigonometric terms of the solutions are represented by $f_j^i(\mathbf{P}_{des})$ to show that they are functions of the desired end-effector position of the manipulator. The table indicates a linear relationship $\theta_2^* = a.\theta_2^s + b$, when $a = -1$ and $b = \pi$. Therefore, for the manipulators of this class, to find a new solution Θ^* from an already known solution Θ^s, operator Ψ_{JRO} is applied as follows:

1.
$$\theta_2^* = \kappa(\Theta^s) = a.\theta_2^s + b = -\theta_2^s + \pi\,. \qquad (4.17)$$

2.
$$\theta_1^* = \arg\min_{\theta_1} \mathbf{E}_p(\theta_1, \theta_2)|_{\theta_2=\theta_2^*}\,. \qquad (4.18)$$

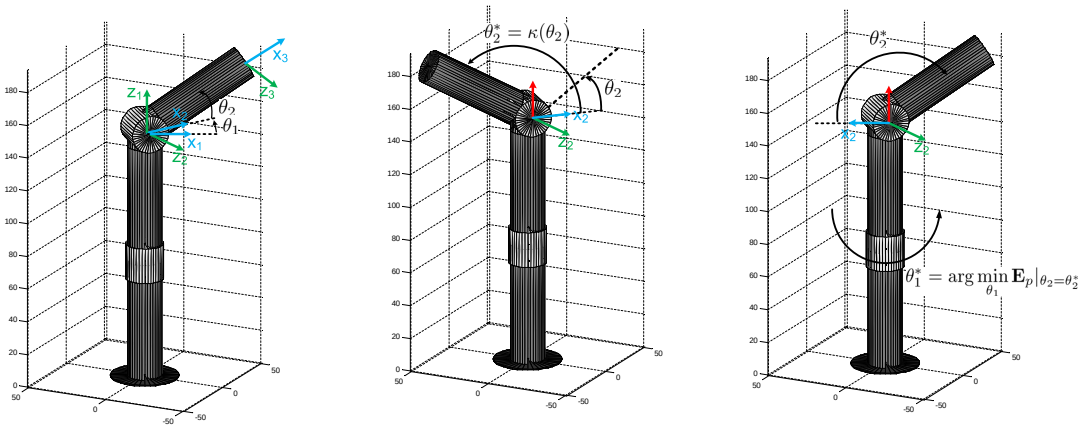
A new solution for the positioning problem is achieved, and κ satisfies the second condition of the theorem.

For example, Fig.4.5(a) depicts a $\mathbf{m}_2$ manipulator with two intersecting joint axes. Also, the corresponding DH frame assignment is illustrated in the figure. Here, the frame of the first joint is moved to the point of intersection of the two joint axes, and therefore, $a_1 = 0$. Fig.4.5(b) illustrates the pose of the manipulator after (4.17) is applied. Fig.4.5(c) shows the pose produced in the second stage of when (4.18) is applied.

Class 2: Parallel Joint Axes: The solution of (4.13) for the manipulators of this group are expressed as

Table 4.4: Positioning IK solutions of a $\mathbf{m}_2$ with intersecting joint axes

	θ_1	θ_2	Stationary Point Type
1	$f_1^{\times}(\mathbf{P}_{des})$	$f_2^{\times}(\mathbf{P}_{des})$	Solution
2	$f_3^{\times}(\mathbf{P}_{des})$	$\pi - f_2^{\times}(\mathbf{P}_{des})$	Solution



(a) (b) (c)

Figure 4.5: Steps involved in operator Ψ_{JRO} in an $\mathbf{m}_2$ with intersecting joint axes: (a) original pose at joint angle Θ^s; (b) second joint at θ_2^*; and, (c) pose with both joints reflected at joint angle Θ^*

Table 4.5: Positioning IK solutions of a $\mathbf{m}_2$ with parallel joint axes

	θ_1	θ_2	Stationary Point Type
1	$f_1^{\shortparallel}(\mathbf{P}_{des})$	$f_2^{\shortparallel}(\mathbf{P}_{des})$	Solution
2	$f_3^{\shortparallel}(\mathbf{P}_{des})$	$-f_2^{\shortparallel}(\mathbf{P}_{des})$	Solution

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} f_1^{\shortparallel}(\mathbf{P}_{des}) \\ \arccos\left(\frac{1}{2}\frac{P_x^2+P_y^2-a_1^2-a_2^2}{a_1.a_2}\right) \end{bmatrix} = \begin{bmatrix} f_1^{\shortparallel}(\mathbf{P}_{des}) \\ f_2^{\shortparallel}(\mathbf{P}_{des}) \end{bmatrix}, \tag{4.19}$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} f_3^{\shortparallel}(\mathbf{P}_{des}) \\ -\arccos\left(\frac{1}{2}\frac{P_x^2+P_y^2-a_1^2-a_2^2}{a_1.a_2}\right) \end{bmatrix} = \begin{bmatrix} f_3^{\shortparallel}(\mathbf{P}_{des}) \\ -f_2^{\shortparallel}(\mathbf{P}_{des}) \end{bmatrix},$$

where $f_{\shortparallel}^1$ and $f_{\shortparallel}^2$ represent the non-linear trigonometric expressions of the solutions of θ_1. Table 4.5 summarizes the positioning solutions of the manipulators. A linear relationship, $\theta_2^* = a.\theta_2^s + b$ with $a = -1$ and $b = 0$, exists. For the manipulators of this class, ψ is performed as the follows

1.
$$\theta_2^* = \kappa(\Theta^s) = a.\theta_2^s + b = -\theta_2^s\,. \tag{4.20}$$

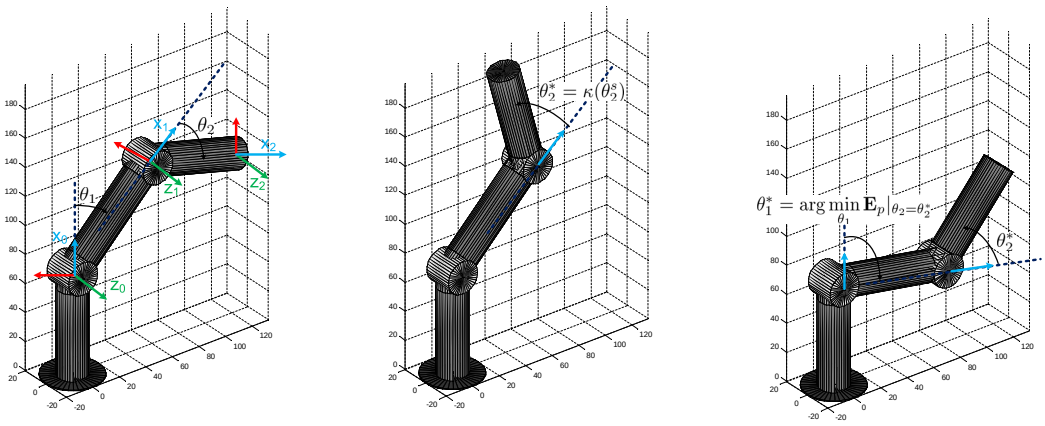
2.
$$\theta_1^* = \arg\min_{\theta_1} \mathbf{E}_p(\theta_1, \theta_2)|_{\theta_2=\theta_2^*}\,. \tag{4.21}$$

For this class of 2-DOF manipulators, operator κ exists and satisfies the second condition of the theorem.

For example, consider $\mathbf{m}_2$ manipulator shown in Fig.4.6(a) and its corresponding DH frames. To find a new positioning solutions for the illustrated pose, θ_2^* is determined by (4.24) to reach the pose of Fig.4.6(b). In the second step, by using (4.25), θ_1^* is calculated.

3. Class 3: Non-Intersecting and Non-Parallel Joint Axes: Solving (4.13) for this class results in four distinct sets of joint angles. However, unlike the previous cases, only one of the joint angle sets yields the positioning error zero. This joint angle is

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{P_y}{P_x}\right) \\ \arctan\left(\frac{P_z-d_1}{a_2}, \frac{+\sqrt{P_x^2+P_y^2}-a_1}{a_2}\right) \end{bmatrix} = \begin{bmatrix} f_1^{\shortparallel\vdash}(\mathbf{P}_{des}) \\ f_2^{\shortparallel\vdash}(\mathbf{P}_{des}) \end{bmatrix}. \tag{4.22}$$



(a) (b) (c)

Figure 4.6: Steps involved in operator Ψ_{JRO} in an $\mathbf{m}_2$ with parallel joint axes: (a) original pose at joint angle Θ^s; (b) second joint at θ_2^*; and, (c) pose with both joints reflected at joint angle Θ^*

Further investigation of the rest of the results of (4.13) indicates that in addition to the mentioned solution in which $\mathbf{E}_P$ becomes zero, a local minimum exists. In the local minimum, although $\mathbf{E}_P$ is not zero, it has a value which is less than all the other points in its vicinity. The joint angle of the local minimum is found by computing

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \pi + \arctan\left(\frac{P_y}{P_x}\right) \\ \arctan\left(\frac{P_z - d_1}{a_2}, \frac{-\sqrt{P_x^2+P_y^2}-a_1}{a_2}\right) \end{bmatrix} = \begin{bmatrix} \pi + f_1^{\shortmid\!\vdash}(\mathbf{P}_{des}) \\ f_3^{\shortmid\!\vdash}(\mathbf{P}_{des}) \end{bmatrix} . \quad (4.23)$$

The solution and the local minimum are summarized in Table 4.6. As can be seen, a linear relationship, $\theta_2^* = a.\theta_2^s + b$ with $a = +1$ and $b = \pi$, exists. Thus, the steps involved to find a local optimum from a solutions and vice-versa through operator ψ are written as follows

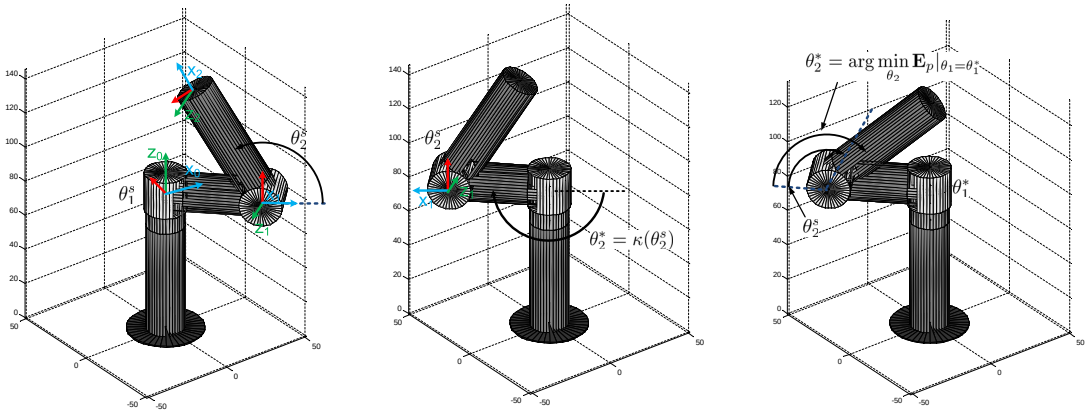
1.
$$\theta_1^* = \kappa(\Theta^s) = a.\theta_1^s + b = \theta_1^s + \pi \, . \quad (4.24)$$

2.
$$\theta_2^* = \arg\min_{\theta_2} \mathbf{E}_p(\theta_1, \theta_2)|_{\theta_1=\theta_1^*} \, . \quad (4.25)$$

Fig.4.7(a) shows an $\mathbf{m}_2$ with joint axes which are not parallel nor intersecting with the link frames, assigned according to the DH convention. To reach the local minimum of the positioning error, first the angle of joint 1 is moved by π radians as in Fig.4.7(b). Then an

Table 4.6: Positioning IK solutions of a $\mathbf{m}_2$ with joint axes which are not intersecting nor parallel

	θ_1	θ_2	Stationary point type
1	$f_1^{\shortmid\vdash}(\mathbf{P}_{des})$	$f_2^{npi}(\mathbf{P}_{des})$	Solution
2	$\pi + f_1^{\shortmid\vdash}(\mathbf{P}_{des})$	$f_3^{npi}(\mathbf{P}_{des})$	Local Minimum



(a) (b) (c)

Figure 4.7: Steps involved in operator Ψ_{JRO} in an $\mathbf{m}_2$ with joint axes which are not intersecting nor parallel: (a) original pose at joint angle Θ^s; (b) first joint at θ_1^*; and, (c) pose with both joints reflected at joint angle Θ^*

angle for the second joint in which the positioning error is minimum is found to achieve the pose in Fig.4.7(c).

Table 4.7 shows operator Ψ_{JRO} and the linear operator κ for all the three possible cases of the manipulators of class $\mathbb{M}_2$. There is a linear operator κ for each $\mathbb{M}_2$ class manipulator.

□

The next theorem guarantees that the second step of the operator Ψ_{JRO}, defined in (4.15), always has a unique solution reachable by any local search algorithm.

Theorem 2. *Uniqueness of the minimum of* $\mathbf{E}_p$*: Given the class of manipulators* $\mathbf{m}_2 \in \mathbb{M}_2$*, a*

Table 4.7: Summary of the ψ operator for the considered cases

	class	θ_1^*	θ_2^*	a	b
1	$\times$	$\arg\min_{\theta_1} \mathbf{E}_p(\theta_1,\theta_2)\|_{\theta_2=\theta_2^*}$	$\kappa(\Theta^s)$	-1	π
2	$\shortmid\shortmid$	$\arg\min_{\theta_1} \mathbf{E}_p(\theta_1,\theta_2)\|_{\theta_2=\theta_2^*}$	$\kappa(\Theta^s)$	-1	0
3	$\vdash$	$\kappa(\Theta^s)$	$\arg\min_{\theta_2} \mathbf{E}_p(\theta_1,\theta_2)\|_{\theta_1=\theta_1^*}$	+1	π

known solution Θ^s, and $\theta_i^* = \kappa(\Theta_s)$ with κ defined as in Theorem.1, $\arg\min_{\theta_j}\ \mathbf{E}_p(\theta_1, \theta_2)|_{\theta_i=\theta_i^*}$ *has only one minimum.*

Proof. When joint i of a 2-DOF manipulator is considered fixed, the manipulator is transformed into a 1-DOF manipulator with only joint j providing the DOF. To prove that $\mathbf{E}_P$ has only one minimum, the critical points of $\mathbf{E}_P$ are calculated by equating the derivative of $\mathbf{E}_P$ with respect to θ_j, to zero and solving for θ_j. The second derivative test at the critical points can then be used to determine if a critical point is a maximum or a minimum. If only one minimum exists, the theorem is proven.

According to which of the two joints are assumed to be fixed at the corresponding θ_i^*, the relationship of the end-effector of the manipulator, as a function of the joint angles (defined in (4.10)), is written in one of the following forms

$$\mathbf{T}_{0,1}(\theta_1^*) = \mathbf{T}_{des}.\mathbf{T}_{1,2}(\theta_2^*)^{-1} = \mathbf{T}_{des}^f\ , \tag{4.26}$$

or

$$\mathbf{T}_{1,2}(\theta_2^*) = \mathbf{T}_{0,1}(\theta_1^*)^{-1}.\mathbf{T}_{des} = \mathbf{T}_{des}^f\ , \tag{4.27}$$

where $\mathbf{T}_{des}^f$ is a newly desired position in the task space, including the Homogenous Transformation of the fixed joint i, and is written as

$$\mathbf{P}_{des}^f = \begin{bmatrix} P_x^f \\ P_y^f \\ P_z^f \end{bmatrix}\ . \tag{4.28}$$

The left sides of (4.26) and (4.27) represent a 1-DOF manipulator with one joint. According to (2.21), if the angle of the single joint of this manipulator is represented by θ_j, the position of the end-effector of the resulting 1-DOF manipulator is then

$$\mathbf{P}_{ee} = \begin{bmatrix} a_1.\cos(\theta_j) \\ a_1.\sin(\theta_j) \\ d_1 \end{bmatrix}\ . \tag{4.29}$$

By using (2.28), the positioning error of the manipulator, with respect to the newly defined task $\mathbf{T}_{des}^f$ is expressed as follows

$$\begin{aligned}\mathbf{E}_p\left(\theta_j\right) &= \left\|\mathbf{P}_{des}^{f} - \mathbf{P}_{ee}\left(\Theta\right)\right\| \\ &= \sqrt{-2\,a_1\cos\left(\theta_j\right)P_x^f + {P_x^f}^2 - 2\,a_1\sin\left(\theta_j\right)P_y^f + {P_y^f}^2 + {d_1}^2 - 2\,d_1 P_z^f + {P_z^f}^2 + {a_1}^2}\,.\end{aligned} \tag{4.30}$$

To find the critical points of $\mathbf{E}_p$, with respect to θ_j, the following is solved

$$\begin{aligned}\frac{d\,\mathbf{E}_p}{d\theta_j} &= \frac{1}{2}\,\frac{2\,a_1\sin\left(\theta_j\right)P_x^f - 2\,a_1\cos\left(\theta_j\right)P_y^f}{\sqrt{-2\,a_1\cos\left(\theta_j\right)P_x^f + {P_x^f}^2 - 2\,a_1\sin\left(\theta_j\right)P_y^f + {P_y^f}^2 + {d_1}^2 - 2\,d_1 P_z + {P_z^f}^2 + {a_1}^2}} \\ &= 0\,,\end{aligned} \tag{4.31}$$

where (4.31) has two solutions, $\theta_{j,1}$ and $\theta_{j,2}$, written as

$$\theta_{j,1} = \arctan\left(\frac{P_y^f}{P_x^f}\right)\,, \tag{4.32}$$

$$\theta_{j,2} = \pi + \arctan\left(\frac{P_y^f}{P_x^f}\right)\,. \tag{4.33}$$

Since $\mathbf{E}_p$ at $\theta_{j,1}$ and $\theta_{j,2}$ can be a maximum as well as a minimum, the second derivative test is utilized to determine the type of critical point. The second derivative of $\mathbf{E}_p$ at $\theta_{j,1}$ and $\theta_{j,2}$ is denoted by

$$\frac{d^2\mathbf{E}_p}{d{\theta_j}^2}(\theta_{j,1}) = \frac{a_1.P_{xy}^f\sqrt{P_{xy}^f}}{\sqrt{2\,{P_x^f}^2 a_1 + {P_x^f}^2 P_{xy}^f + 2\,a_1 {P_y^f}^2 + {P_y^f}^2 P_{xy}^f + {d_1}^2 P_{xy}^f - 2\,d_1 P_z^f\,P_{xy}^f + {P_z^f}^2 P_{xy}^f + {a_1}^2 P_{xy}^f}}\,, \tag{4.34}$$

$$\frac{d^2\mathbf{E}_p}{d{\theta_j}^2}(\theta_{j,2}) = \frac{-a_1.P_{xy}^f\sqrt{P_{xy}^f}}{\sqrt{2\,{P_x^f}^2 a_1 + {P_x^f}^2 P_{xy}^f + 2\,a_1 {P_y^f}^2 + {P_y^f}^2 P_{xy}^f + {d_1}^2 P_{xy}^f - 2\,d_1 P_z^f\,P_{xy}^f + {P_z^f}^2 P_{xy}^f + {a_1}^2 P_{xy}^f}}\,, \tag{4.35}$$

in which

$$P_{xy}^f = \sqrt{{P_x^f}^2 + {P_y^f}^2}\,. \tag{4.36}$$

It is obvious that the second derivative at $\theta_{j,1}$ and $\theta_{j,2}$ is positive and negative, respectively. As a result, $\theta_{j,1}$ is a minimum whereas $\theta_{j,2}$ is a maximum.

Therefore, by keeping one of the joint angles fixed, $\mathbf{E}_p$ has only one minimum. In addition, (4.15) with θ_i^* defined as in Theorem 1 has a unique solution.

□

The process of finding new solutions for $\mathbb{M}_2$ class manipulators is called the *Joint Reflection Operator*, and is represented by Ψ_{JRO}. JRO consists of two stages; applying the linear operator κ, and the minimization problem for finding the combined positioning and orienting minimum. Finding a new solution Θ^* from a known solution Θ^s, through the Joint Reflection Operator of $\mathbb{M}_2$ manipulators, is expressed as follows for manipulator $\mathbf{m}_2$

$$\Theta^* = \Psi_{JRO}^{\mathbb{M}_2}(\mathbf{m}_2, \Theta^s)\ . \tag{4.37}$$

Part II of Theorem 1 states that applying the κ operator twice to pose Θ^s, consecutively, results in the same pose Θ^s. When the Ψ_{JRO} is applied to pose Θ^s two times, joint θ_i which has undergone the κ operator stays the same. According to Theorem 2, the angle of the other joint θ_j is unique for each θ_i, that is, θ_j has a one to one relationship with θ_i. When two Ψ_{JRO} are

applied to pose Θ^s, consecutively, θ_i does not change, and because of the one to one relationship, θ_j remains the same. Thus, the following characteristic for Ψ_{JRO} is deducted:

$$\Theta^* = \Psi_{JRO}^{\mathbb{M}_2}(\mathbf{m}_2, \Theta^s) \quad \Leftrightarrow \quad \Theta^s = \Psi_{JRO}^{\mathbb{M}_2}(\mathbf{m}_2, \Theta^*) \ . \tag{4.38}$$

Finding Multiple Positioning IK Solutions for $\mathbb{M}_3$ Manipulators

Any member of $\mathbb{M}_3$ can be reduced to a member of $\mathbb{M}_2$, when one joint is assumed to be fixed and only two of the joints are rotating. An operator $\Lambda(\mathbf{m}_3, \Theta^s, i, j)$, is defined for the conversion from an $\mathbf{m}_3$ manipulator at pose $\Theta^s = [\theta_1^s \ \theta_2^s \ \theta_3^s]$ to a $\mathbf{m}_2$, when all the joints, except the i-th and j-th joints, are fixed. Then,

$$\Lambda(\mathbf{m}_3, \Theta^s, i, j) = \{ \mathbf{m}_3 \,|\theta_k = \theta_k^s \ \text{ for all } \ k \neq i, j \} \ . \tag{4.39}$$

$\mathbb{M}_2^{\mathbf{m}_3}$ represents the set of $\mathbf{m}_2$ manipulators which can result from applying $\Lambda(\mathbf{m}_3, \Theta^s, i, j)$ to manipulator $\mathbf{m}_3$ at pose Θ^s for all i and j, and is expressed as

$$\mathbb{M}_2^{\mathbf{m}_3}(\Theta^s) = \bigcup_{\substack{i=1,2,\cdots 3 \\ j=1,2,\cdots 3 \\ j \neq i}} \Lambda(\mathbf{m}_3, \Theta^s, i, j) \ . \tag{4.40}$$

For example, Fig.4.8 shows an $\mathbf{m}_3$ at pose Θ^s and its corresponding $\mathbb{M}_2^{\mathbf{m}_3}(\Theta^s)$. In Fig.4.8(b), the first joint is assumed to be fixed and the manipulator is reduced to a $\mathbb{M}_{\shortparallel,2}$ class manipulator. In Fig.4.8(c), by keeping the second joint fixed, the resulting $\mathbf{m}_2$ is a $\mathbb{M}_{\shortparallel\vdash,2}$ class manipulator. In Fig.4.8(d), the third joint is assumed to be fixed and the resulting manipulator is a class $\mathbb{M}_{\times,2}$. As is illustrated, the last link of $\Lambda(\mathbf{m}_3, \theta^s, 1, 2)$ is modified such that the resulting $\mathbf{m}_2$ can reach the same position in the task space as the original $\mathbf{m}_3$.

In the previous section, an operator $\Psi_{JRO}^{\mathbb{M}_2}$ for $\mathbb{M}_2$ manipulators is introduced to find a new solution Θ^* from a known solution Θ^s. In $\mathbf{m}_3$, any two joints, i and j, belong to one of the three classes of $\mathbb{M}_2$ manipulators, including $\mathbb{M}_{\times,2}$, $\mathbb{M}_{\shortparallel,2}$, and $\mathbb{M}_{\shortparallel\vdash,2}$. It is found that a new positioning solution Θ^* (or a local minimum in the case of $\mathbb{M}_{\shortparallel\vdash,2}$ class) is reachable from Θ^s by the $\Psi_{JRO}^{\mathbb{M}_2}$ operator when it is applied to $\Lambda(\mathbf{m}_3, \Theta^s, i, j)$. Therefore, by keeping one joint angle constant and allowing motion in the i-th and j-th joints (i, j), a new solution, represented by Θ_{ij}^* can be calculated by using the $\Psi_{JRO}^{\mathbb{M}_2}$ operator of $\mathbb{M}_2$ manipulators as follows

$$\Theta_{ij}^* \ = \ \Psi_{JRO}^{\mathbb{M}_2}(\Lambda(\mathbf{m}_3, \Theta^s, i, j), \Theta^s) \tag{4.41}$$

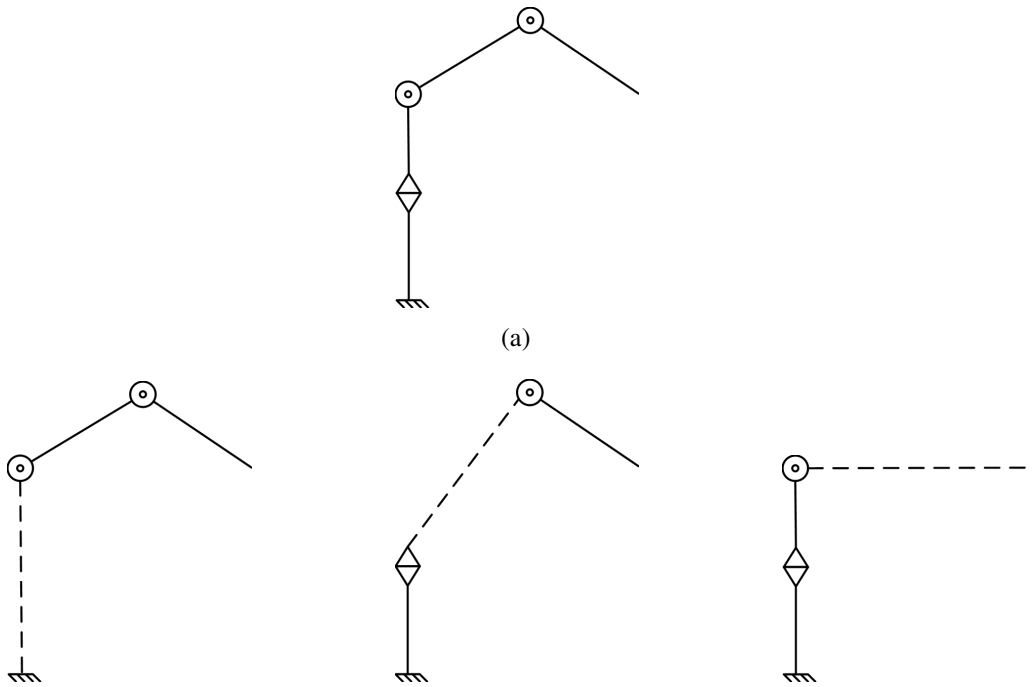


Figure 4.8: Example 1: Sample $\mathbf{m}_3$ and corresponding $\mathbb{M}_2^{\mathbf{m}_3}$ manipulators: (a) original $\mathbf{m}_3$ at pose Θ^s; (b) $\Lambda(\mathbf{m}_3, \Theta^s, 2, 3)$; (c) $\Lambda(\mathbf{m}_3, \Theta^s, 1, 3)$; and, (d) $\Lambda(\mathbf{m}_3, \Theta^s, 1, 2)$

with the following characteristics.

1. Characteristic 1: When the Ψ_{JRO} operator is applied to a solution Θ^s twice, according to part.II of Theorem.1, the angle of the joint which undergoes the κ operator twice remains the same. As for the angle of the other joint, since the solution of the minimization problem is unique (according to theorem.2) it does not change either. Consequently, the following characteristics is defined

$$\begin{aligned} \Theta_{ij} &= \Psi_{JRO}^{\mathbb{M}_2}\left(\Lambda\left(\mathbf{m}_3, \Theta^s, i, j\right), \Theta^s\right) \\ &\Updownarrow \\ \Theta^s &= \Psi_{JRO}^{\mathbb{M}_2}\left(\Lambda\left(\mathbf{m}_3, \Theta_{ij}, i, j\right), \Theta_{ij}\right) . \end{aligned} \tag{4.42}$$

2. Characteristic 2: When the Ψ_{JRO} operator is applied to the joint couple (i, j), according to the definition of Ψ_{JRO}, the i-th and j-th joint angles are determined such that the position of the end-effector remains the same and $\Lambda\left(\mathbf{m}_3, \Theta^s, k, l\right)$ for all $(k, l) \neq (i, j)$ does not change. Therefore, when a new Ψ_{JRO} operator, with respect to a new joint couple (k, l) is applied, it does not affect the result of the first Ψ_{JRO}. As a result, in the Ψ_{JRO} operator the

order which the joint couples (i, j) and (k, l) undergo the Ψ_{JRO} operator does not change the final result. It is concluded that the Ψ_{JRO} operator is a commutative operator, such that

$$\begin{cases} \Theta_{ij} &= \Psi_{JRO}^{\mathbb{M}_2}\left(\Lambda\left(\mathbf{m}_3, \Theta, i, j\right), \Theta\right) \\ \Theta_{ij,kl} &= \Psi_{JRO}^{\mathbb{M}_2}\left(\Lambda\left(\mathbf{m}_3, \Theta_{ij}, k, l\right), \Theta_{ij}\right) \end{cases}, \tag{4.43}$$

and:

$$\begin{cases} \Theta_{kl} &= \Psi_{JRO}^{\mathbb{M}_2}\left(\Lambda\left(\mathbf{m}_3, \Theta, k, l\right), \Theta\right) \\ \Theta_{kl,ij} &= \Psi_{JRO}^{\mathbb{M}_2}\left(\Lambda\left(\mathbf{m}_3, \Theta_{kl}, i, j\right), \Theta_{kl}\right) \end{cases}, \tag{4.44}$$

then

$$\Theta_{ij,kl} = \Theta_{kl,ij}\ . \tag{4.45}$$

If a new solution, Θ_{ij}^*, is achieved by using (4.41), Θ_{ij}^* can also undergo the $\Psi_{JRO}^{\mathbb{M}_2}$ operator. By considering another 2-DOF manipulator of $\mathbf{m}_3$, consisting of a new joint couple, (k, l), to reach yet another new positioning solution (or local minimum)

$$\Theta_{ij,kl}^* = \Psi_{JRO}^{\mathbb{M}_2}\left(\Lambda\left(\mathbf{m}_3, \Theta_{kl}^*, k, l\right), \Theta_{ij}^*\right)\ . \tag{4.46}$$

This process continues until all of the joint couples that can form a $\mathbf{m}_2$ from the three classes of $\mathbb{M}_2$, is examined. This process is converted to and implemented as a tree structure with the original solution Θ^s as the root. Fig.4.9 depicts such a tree for an $\mathbf{m}_3$ manipulator. In the tree, each node represents a new positioning solution. An edge from node Θ_{ij}^* to node $\Theta_{ij,kl}^*$ is obtained by (4.41). In the tree, by iteratively applying (4.41) each time considering a different joint couple, new solutions can be reached. The greyed out nodes represent the solutions which have already been found through the other branches of the tree due to characteristics 1 or 2 of (4.41). Therefore, in the $\mathbf{m}_3$ manipulators, finding the new solutions facilitates as a search in the branches of the shown tree.

Finding Multiple Positioning IK Solutions for $\mathbb{M}_n$ Manipulators

Any manipulator with more than 3-DOF is considered redundant for positioning tasks. For redundant manipulators, a common practice is to assume that the redundant joints lack any

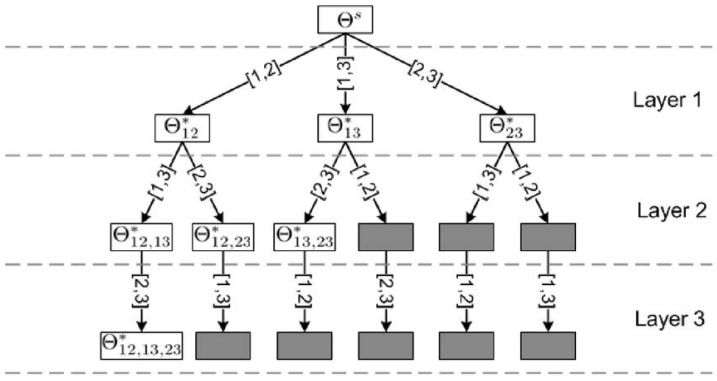


Figure 4.9: Tree structure resulting from applying $\Psi_{JRO}^{\mathbb{M}_2}$ operator to all the joint couples

DOF, reducing the manipulator to a non-redundant manipulator, and then solving the resulting IK problem [119].

Since the cornerstone of the joint reflection operator is to find the positioning solutions, the same method can be used to reduce the positioning-redundant n-DOF manipulators to a positioning-non-redundant 3-DOF manipulator, and then applying the operator to the reduced manipulator. Therefore, in $\mathbf{m}_n$ manipulators, by using a similar method to $\mathbf{m}_3$ manipulators, new solutions are extracted from a known solution. Specifically, the tree of the joint couples which can produce new positioning solutions is created layer by layer. Then, the $\Psi_{JRO}^{\mathbb{M}_2}$ is applied to each of the nodes.

4.3.2 Finding Multiple Positioning and Orienting IK Solutions

When a new positioning solution is found using the operator $\Psi_{JRO}^{\mathbb{M}_2}$, a numerical algorithm with the objective function of (2.32) is used to achieve convergence to the minima of the combined positioning/orienting IK problem. For $\mathbf{m}_2$ and non-redundant $\mathbf{m}_3$ manipulators, only one positioning/orienting IK solution exists, and the rest of the solutions of $\mathbf{E}_P$ cannot satisfy the second condition of the IK solution, $\mathbf{E}_O = 0$. However, in manipulators with a DOF higher than three, the newly found positioning solution can be used as an initial point for a numerical search for a positioning/orienting solution. The result of such a search includes joint angles capable of satisfying both $\mathbf{E}_P = 0$ and $\mathbf{E}_O = 0$.

In this thesis, the numerical search algorithm is implemented by a BFGS Quasi-Newton method [120, 121, 122, 123] with a mixed quadratic and cubic line search procedure in conjunction with a Jacobian-based IK solver [73, 74].

Table 4.8 shows the pseudo-code of the proposed algorithm for finding the positioning/orienting IK solutions. The inputs of the algorithm consist of the manipulator $\mathbf{m}_n$, the desired task in the

Table 4.8: Pseudocode of the algorithm for finding all the solutions of the IK

$\Theta_{all} = Find\_all\_IK\_solutions(\mathbf{m}_n, T_{des}, \Theta^s)$

1 IF (Θ^s *is already processed*)
2 END
3 ENDIF

4 FOR ($i = 1$ to $(dof - 1)$)
5 FOR ($j = i$ to dof)

6 IF ((i, j) *unprocessed, based on characteristics* 1 *and* 2)

7 $\Theta^* = \Psi^{\mathbb{M}_2}_{JRO}(\Lambda(\mathbf{m}_n, \Theta^s, i, j), \Theta^s)$
8 $\Theta_{LS} = Numerical\_IK(\mathbf{m}_n, T_{des}, \Theta^*)$

9 IF (Θ_{LS} *is a solution and not in* Θ_{all})
10 $\Theta_{all} = add\_to\_solution\_list(\Theta_{all}, \Theta_{LS})$
11 ENDIF

12 $\Theta_{all} = Find\_all\_IK\_solutions(\mathbf{m}_n, T_{des}, \Theta^*)$

13 ENDIF

14 ENDFOR
15 ENDFOR

Cartesian space, defined as a Homogenous Transformation T_{des}, and a known IK solution Θ^s, where the output is a list of all the IK solutions Θ_{all}. In lines 1–3, Θ^s is checked to confirm that it has not already undergone the joint reflection operator. Lines 4–5 are a nested loop which iterate through all the joint couples (i, j) of the manipulator. In line 6, the selected joint couple (i, j) are examined, by using characteristics 1 and 2, in relation to the joint couples already processed. Only if the couple (i, j) can produce new solutions, are they allowed to undergo the rest of the nested loops. With the progress of the algorithm, the number of unexplored joint couples decreases, and the conditional statement of this line expresses the termination criteria of the recursive algorithm. In line 7, $\Psi^{\mathbb{M}_2}_{JRO}$ is applied to Θ^s to reach a new positioning solution. In line 8, the newly found positioning solution Θ^* is employed as the initial point for obtaining the combined positioning/orienting IK by a Numerical IK algorithm. In lines 9–11, Θ_{LS} is added to the list of the found solutions, Θ_{all}, if it is a solution (and not a local minimum), and has not already been found. Line 12, calls the $Find\_all\_IK\_solutions$ algorithm, recursively, to find new solutions from the newly found positioning solution Θ^*.

4.4 Results

In this section, the results of applying the proposed Joint Reflection Operator based IK solver for solving the IK problem of three distinct 6-DOF KCs are given. In the numerical analysis, since the intention is to verify the ability of the algorithm in finding multiple-solutions of IK, no collision detection or joint angle limits are incorporated. In practical applications, the IK solutions, which violate collision detection or joint angle limitations, can be easily separated after all the IK solutions are found.

The processing time of the multi-solution IK solver of Chapter 3 is approximately 40 minutes for a 6-DOF PUMA manipulator. The run time of the same experiment, under the computer setup used for the analysis of this chapter is around 25 minutes. The worst processing time, encountered in the numerical analysis of the algorithm of this chapter, is less than 30 seconds. Therefore, the proposed algorithm is, by far, more time efficient than the algorithm of the previous chapter.

As mentioned in Section 2.6.2, continuation methods (CMs) are proven to be efficient and robust multi-solution IK solvers. Moreover, continuation methods are more accurate and faster than elimination methods, when the complexity of the manipulators increases. Therefore, CM is selected as one of the comparison references for JRO-based IK solvers. One of the subroutines of the Matlab toolbox implementation of CM, called HomLab 10, which is specifically developed to solve the IK problem is utilized [124]. HomLab 10 has been developed as companion software of [125]. For solving the IK problem, CM is utilized in two fashions. In the first, which is used in this analysis, the algorithm follows 320 distinct paths. This method is suitable for cases in which no prior knowledge of the solutions of the manipulator exists. On the contrary, if enough information about the characteristics of the IK solutions exists prior to running the CM algorithm, the known information is applied to initialize the algorithm and limit the number of considered trajectories in order to significantly enhance the speed of the algorithm.

As another comparison reference for the convergence speed of the proposed approach, a single solution IK solver, based on numerical iterative IK solvers is utilized. To enable the single solution algorithm to find multiple solutions of IK, it is iteratively initialized with a random starting point, until all the solutions of the IK are found. No guarantee exists that the randomly initialized algorithm can find all the solutions within a reasonable time or iteration number. Therefore, if the algorithm cannot find all the solutions within 1000 iterations, the run is terminated under the assumption that the algorithm cannot to find all the solutions. Because of the stochastic nature of the randomly initialized local search, the algorithm is run five times for each test, and the minimum, maximum, and average time of the runs are recorded for comparison. The randomly initialized local search algorithm is referred to as the RI.

It is noteworthy that both the JRO and CM algorithms search the entire joint space for all

the possible IK solutions without requiring the exact number of solutions. However, the RI algorithm needs the number of solutions, as the termination criteria. Therefore, in the conducted numerical analysis, the RI has the advantage of termination of the operation as soon as all the solutions were found, while the other two methods performed their search regardless of the number of solutions found and until no further search options exist.

4.4.1 6-DOF Kinematic Configuration 1

The matrix representation of the first tested 6-DOF KC is

$$\mathbf{m}_6{}^1 = \begin{bmatrix} 0 & 0 & 60 \\ 0 & PR & 60 \\ 0 & P & 60 \\ 0 & R & 60 \\ 0 & P & 60 \\ 0 & P & 0 \\ 0 & R & 0 \end{bmatrix}. \tag{4.47}$$

Since the manipulator does not have three intersecting joint axes, there is no guarantee that closed-form IK solutions exist [61, 62]. Depending on the desired position and orientation of the end-effector, the manipulator can have either four or eight IK solutions, making the interval methods difficult to apply. Fig.4.10 reflects all the solutions, where four IK solutions exist, and Fig.4.11 shows the solutions where eight IK solutions exist. Ten desired end-effector position/orientation, reachable by the manipulator, are generated randomly. The RI, CM, and JRO algorithms are applied to find the IK solutions corresponding to each test case. In all the cases, all the algorithms can find all the IK solutions. The number of IK solutions s, run times of CM and JRO, in addition to the maximum, minimum, and average run times of RI are shown in Table 4.9. As seen from the table, the run time of the RI algorithm is less than that of the JRO algorithm. In all of the tested cases, the run time of JRO is significantly less than that of the CM. In the 7-th test case, in which the JRO algorithm exhibits its worst performance, the run time is still 24.6% of the run time of the CM.

4.4.2 6-DOF Kinematic Configuration 2

In this numerical analysis, a 6-DOF manipulator with the following KC is tested

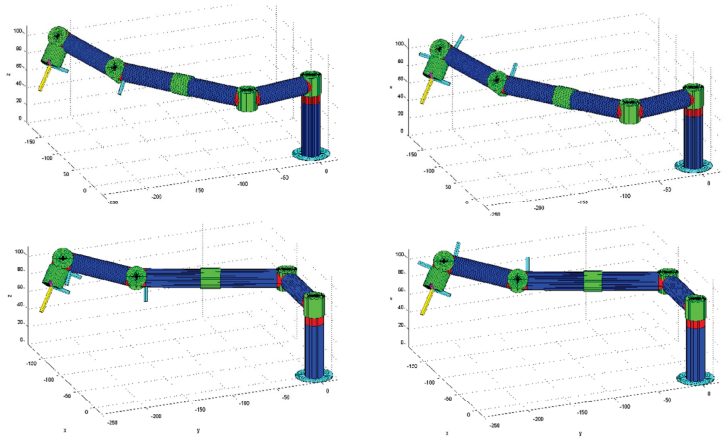


Figure 4.10: Tested $\mathbf{m}_6{}^1$ in a pose with four IK solutions.

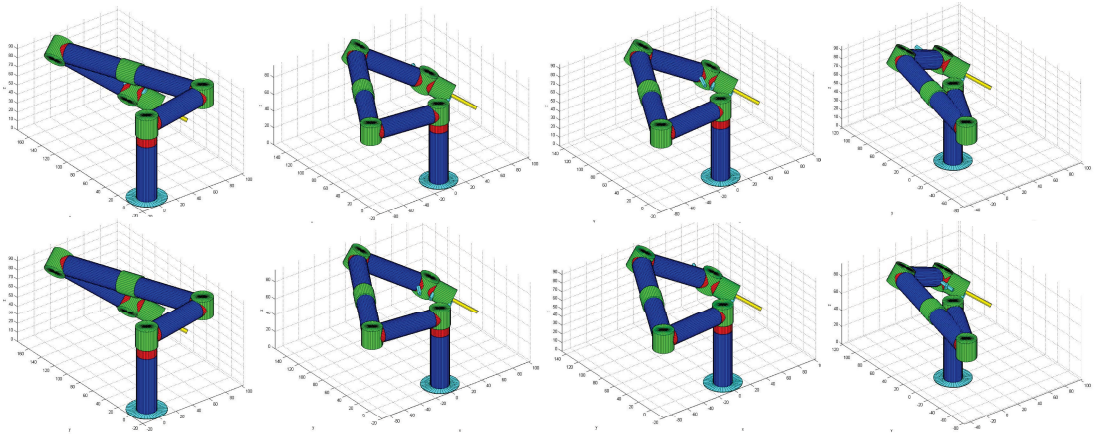


Figure 4.11: Tested $\mathbf{m}_6{}^1$ in a pose with eight IK solutions.

Table 4.9: Time comparison of the RI, CM, and JRO methods for $\mathbf{m}_6{}^1$. Unit of time is seconds.

Processing Time (s)	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
s	4	4	4	4	4	8	4	4	8	4
RI (min.)	5.0	3.2	2.8	2.3	5.3	13.2	9.9	1.5	7.0	9.0
RI (avg.)	7.7	5.6	5.0	4.9	6.0	16.6	15.6	5.1	19.4	10.2
RI (max.)	13.4	8.3	8.6	11.6	7.1	20.2	20.5	12.9	26.3	12.9
CM	143.5	141.9	143.7	143.7	136.4	138.0	147.0	129.5	143.8	121.5
JRO	21.29	22.4	14.1	18.4	33.1	29.5	36.2	20.9	33.6	23.3

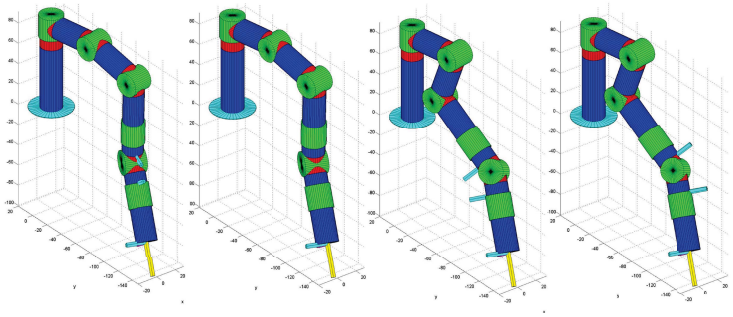


Figure 4.12: Tested ${\mathbf{m}_6}^2$ in a pose with four IK solutions.

Table 4.10: Time comparison of the RI, CM, and JRO methods for ${\mathbf{m}_6}^2$. Unit of time is seconds.

Processing Time (s)	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
s	4	4	8	4	4	4	8	4	4	8
RI (min.)	3.1	2.5	4.1	8.3	4.7	4.0	14.5	2.6	1.8	14.2
RI (avg.)	6.1	4.4	18.1	16.4	10.3	10.0	23.3	4.6	2.2	22.3
RI (max.)	9.6	6.3	34.9	21.5	21.5	13.4	30.1	7.0	2.8	25.8
CM	118.6	131.6	130.9	131.5	139.9	132.6	131.9	124.0	131.1	135.8
JRO	18.6	13.7	31.6	25.2	15.4	24.0	49.6	14.7	15.8	45.9

$$
{\mathbf{m}_6}^2 = \begin{bmatrix} 0 & 0 & 60 \\ 0 & PR & 30 \\ \pi/2 & P & 30 \\ 0 & P & 30 \\ 0 & R & 10 \\ 0 & P & 10 \\ 0 & R & 30 \end{bmatrix}. \tag{4.48}
$$

Depending on the desired position and orientation of the end-effector, either four or eight IK solutions for this manipulator can be found. Fig.4.12 and Fig.4.13 illustrate the two cases in which the manipulator has four and eight IK solutions, respectively. In Table 4.10, a time comparison of RI, CM and JRO is shown. The JRO algorithm seems to be slower than the RI method and faster than the CM in all the test cases. In the 12-th case, in which the JRO has its worst performance, the run time is still 36% of the run time of the CM.

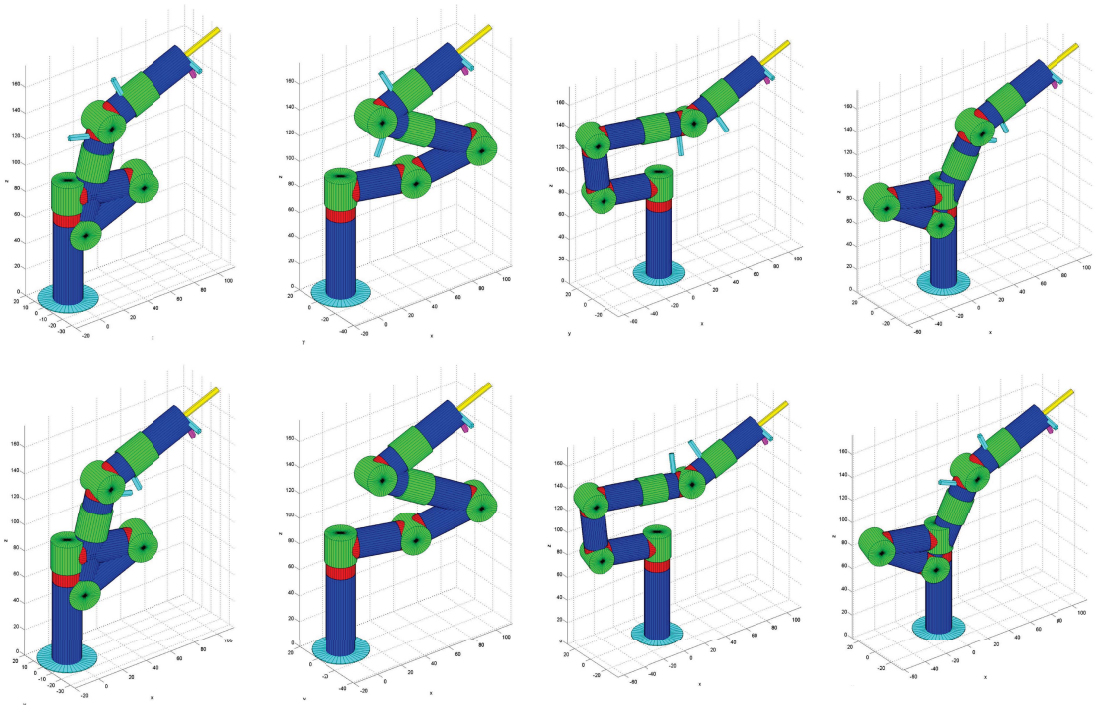


Figure 4.13: Tested ${\mathbf{m}_6}^2$ in a pose with eight IK solutions.

4.4.3 6-DOF PUMA Manipulator

In this experiment, the JRO-based algorithm is tested on the 6-DOF PUMA manipulator in Fig.4.14. The KC representation of the PUMA is

$$ {\mathbf{m}_6}^P = \begin{bmatrix} 0 & 0 & 60 \\ 0 & PR & 10 \\ 0 & PR & 60 \\ 0 & P & 60 \\ 0 & R & 0 \\ 0 & P & 0 \\ 0 & R & 0 \end{bmatrix} . \tag{4.49} $$

In Fig.4.14, the four IK solutions of a test case for PUMA type manipulator are shown. The other four solutions of the PUMA are similar to the shown solutions in the first three joint angles, and differ only in the angle value of the three distal joints. For the PUMA experiment, 40 randomly generated task points are created. In all of the test cases JRO, CM, and RI can find all the IK solutions. The processing time of the JRO, CM as well as the minimum, maximum, and average run time of the RI are plotted in Fig.4.15. Fig.4.16 reflects the scaled version of the same graph to clarify the comparison of JRO and RI. In Fig.4.15, the run time, averaged over all

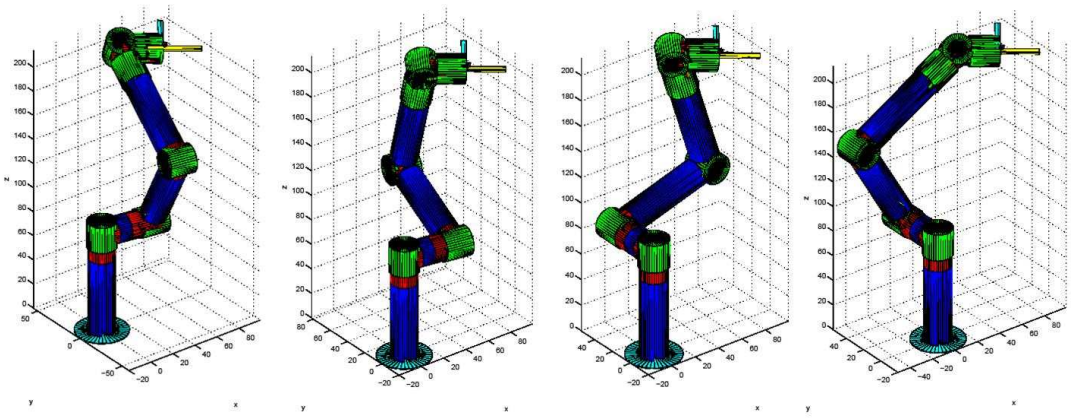


Figure 4.14: Tested 6-DOF PUMA type manipulator at four of the possible eight IK solutions

Tested 6-DOF PUMA type manipulator at four of the possible eight IK solutions - The other four solutions are similar to the shown poses with different the wrist angles

the tested cases, is around 11 seconds for the JRO. The same measure for the CM is 130 seconds. Fig.4.16 shows that although the minimum run time of the RI for each case is lower than the run time of the JRO, the differences between the run times of the JRO and the average run time of the RI are not considerable. Although the RI proves to be slightly faster than the JRO, due to it requiring a priori knowledge of the number of IK solutions, the RI is not suitable for solving the IK problem of the manipulators in which the number of the IK solutions is unknown or variable with respect to the coordinates of the end-effector.

4.5 Summary

In this chapter, a multi-solution IK solver for MRRs is proposed. The algorithm is capable of finding multiple positioning solutions of the manipulator. The positioning solutions are used as initial points for solving the combined positioning/orienting IK problem. To find the multiple positioning solutions, the algorithm iteratively applies the Joint Reflection Operator to the joint pairs, capable of producing multiple solutions, until all the positioning solutions are found.

The performance of the algorithm is compared with a continuation method and a randomly initialized local search algorithm. It is shown that, although the run time of the proposed algorithm is in the same range as of the average run time of the randomly initialized local search algorithm, the run time of the proposed algorithm was considerably less than the run time of the continuation method.

The JRO-based algorithm searches all the possible joint combinations, i.e., it has a predefined termination criteria. Thus, when the termination criteria is reached, it can be concluded that no more solutions exists. Therefore, similar to the continuation method, the proposed algorithm

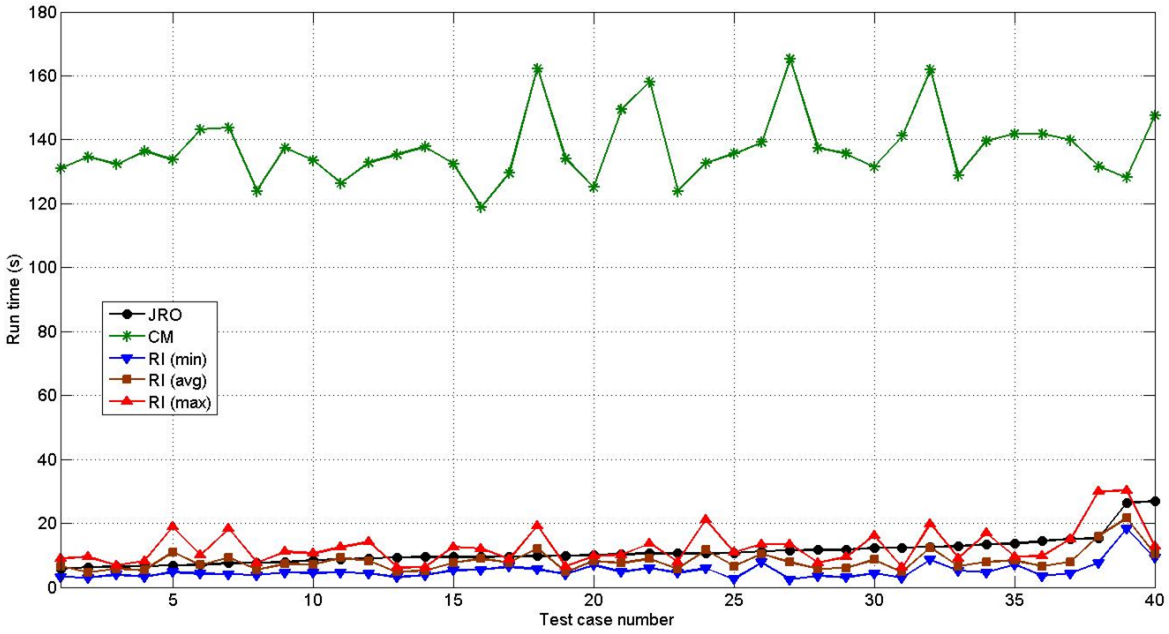


Figure 4.15: Comparison of the JRO, CM and RI for $\mathbf{m}_6{}^P$

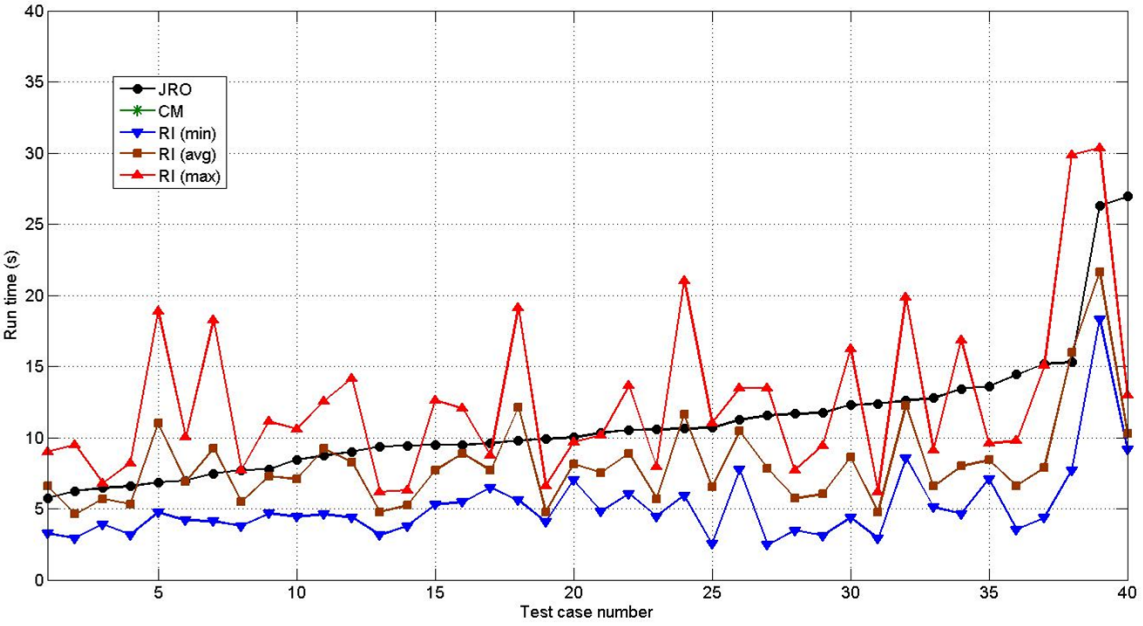


Figure 4.16: Comparison of the JRO and RI for $\mathbf{m}_6{}^P$

can be used for the manipulators in which the number of the IK solutions is unknown or changes with respect to the desired position/orientation of the task. However, in the case of the randomly initialized local search algorithm, a prior knowledge of the number of solutions is needed in order to terminate the algorithm when and if this number is reached. This can be viewed as a disadvantage of the RI algorithm. Therefore, the JRO is a more advantageous tool for solving the IK problem of MRRs.

Chapter 5

Trajectory Optimization Using Multiple Solutions of the Inverse Kinematic Problem

5.1 Introduction

As shown in Section 2.6.4, dynamic performance of manipulators changes, when the utilized IK solution in reaching a task point varies. This characteristic implies that by choosing a suitable IK solution for performing each task point, the performance of a robotic manipulator can be enhanced.

The approaches for solving the trajectory optimization problem can be categorized into two categories. In the first group, the point-to-point trajectory optimization, the goal is to find the most optimized trajectory in the joint angle space to connect two task points in the cartesian space, i.e., the objective is to find the least costly trajectory in the joint space from task point T^i_{des} to T^j_{des}. In the following section, the most prominent works are discussed.

Algorithm have been developed to find the minimum time trajectory of a manipulator to move from one task point to the next [126, 127, 128]. A modified version of the aforementioned algorithm has been used for the trajectory planning with moving obstacles [129]. Joint torque limitations have been considered as the constraints of the algorithm. Saramago has proposed strategies for both moving through a certain set of task points, and for moving from a starting point to an end point [130]. The goal of the algorithm is to find the minimum time and minimum required energy trajectories in the presence of obstacles. An algorithm has been developed for trajectory optimization with the path following and obstacle avoidance, as constraints [131]. Also, algorithms for trajectory optimization in kinematically redundant manipulators have been

presented [132, 133]. In the former, the goal is to minimize the consumed energy, and in the latter, the idea is to avoid the workspace singularities. Furthermore, a GA has been utilized for obstacle avoidance in the trajectory optimization of a robotic manipulator [134].

To minimize the cost function, the methods in the point-to-point trajectory optimization group search for a trajectory in the *joint angle space* of the manipulator to connect two task points $\mathbf{T}_{des}^{1}$ and $\mathbf{T}_{des}^{2}$. They are defined in the *Cartesian space* for a one-to-one mapping between the two spaces. However, most robotic manipulators have multiple IK solutions. As a result, the mapping from each task point in the Cartesian space into the joint angle space is not unique, that is, the problem of connecting a start point to an end point in the cartesian space should be mapped into a problem in the joint angle space in which the start and end points can be chosen from any of their corresponding IK solutions.

In addition, the operation of the industrial manipulators consists of repetitively reaching numerous task points. In a wide range of applications such as spot welding, inspection, and measurement the sequence that the robot reaches the task points is of no importance [135]. Consequently, another addressable problem in trajectory optimization is how to arrange the sequence of task points $\mathbf{T}_{des}^{i}$ to achieve a performance boost.

The second group of algorithms, multi-goal path planners, addresses both. In multi-goal trajectory optimization, the goal is to find a sequence of reaching the task points, coupled with the respective IK solution, such that the result is a collision free shortest path. An algorithm is proposed to solve for approximate solutions of the multi-goal path planning [136]. The multi-goal path planning algorithm considers multiple solutions of the IK. Collision free point-to-point paths are generated with a best-first search on a regular grid in the joint angle space. A GA solves the resulting Traveling Salesman Problem (TSP). Since the algorithm tries to avoid solving the point-to-point path planning problem for as long as possible, the result of algorithm improves, when the time allocated to it increases. An algorithm for solving for near-optimal solutions of the multi-goal path planning problem is also proposed in the literature [135]. A bi-directional tree-expansion PRM planner is used for the point-to-point path planning. The approach is based on the assumption that finding a point-to-point optimized trajectory is more expensive than finding a complete tour of the task points. Thus, a greedy algorithm is devised to prevent solving the point-to-point trajectory until it is absolutely necessary. The aim of both multi-goal path planners is finding a near-optimal solution for the collision free shortest path problem, and no other optimization criteria is considered.

In multi-goal path planners, the point-to-point trajectory optimization should be performed for calculating the cost of moving from one task point to the other. Therefore, point-to-point trajectory optimization can also be considered as a special case of the multi-goal path planners, in which the number of the considered task points is two.

In this chapter, a formulation for finding the exact solution of the trajectory optimization problem is proposed. The formulation incorporates all the solutions of the IK problem, and can be used in conjunction with any of the existing point-to-point trajectory optimization algorithms.

The rest of this chapter is organized as follows. Section 5.2 introduces the new formulation of trajectory optimization into a Generalized Traveling Salesman Problem (GTSP). Section 5.3 explains the proposed algorithm for solving the formulated problem. In Section 5.4, the employed method for converting the resulting GTSP to a TSP is introduced. Finally, in Section 5.5, the result of using the algorithm for trajectory optimization of a 3-DOF planar and a 6-DOF PUMA manipulator, is presented. Section 5.6 summarizes the achievements of this chapter.

5.2 Trajectory Optimization with Multiple IK Solutions

5.2.1 Problem Definition

In many applications, the sequence of reaching the task points is not important. Thus, the sequence of reaching the task points can also be considered as an optimization variable, i.e., the trajectory optimizer finds the sequence which the task points are reached.

Assume a task $\mathbb{T}_t = \{\mathbf{T}^1_{des}\, \mathbf{T}^2_{des}\, \cdots \mathbf{T}^t_{des}\}$ consists of t task points in which each task point has s distinct IK solutions. s depends only on the manipulator's kinematic characteristics and remains the same for all possible $\mathbf{T}^i_{des}$. S^i_j denotes the j-th IK solution for the i-th task point.

In this chapter, the problem of selecting the most suitable IK solution for each task point, $\mathbf{T}^i_{des}$, coupled with the sequence of performing the tasks, $\mathbf{T}^i_{des},\ i \in [1, t]$, to minimize the cost of reaching the entire task points is visited. Mathematically, this problem is expressed as

$$\text{argmin}_{[S^{i_1}_{j_1}, S^{i_2}_{j_2}, \cdots S^{i_n}_{j_n}]} C(S^{i_1}_{j_1}, S^{i_2}_{j_2}) + C(S^{i_2}_{j_2}, S^{i_3}_{j_3}) + \cdots + C(S^{i_{n-1}}_{j_{n-1}}, S^{i_n}_{j_n}) \tag{5.1}$$
$$s.t. \begin{cases} i_1 \cup i_2 \cup \cdots \cup i_n = \mathbb{T}_t \\ j_1,\, j_2,\, \cdots, j_n \ \in [1, s] \end{cases},$$

where $C(x, y)$ represents the cost of moving from pose x to pose y.

Fig.5.1 is a schematic of the multi IK solution trajectory optimization. The top block, which consists of the inputs, includes the IK solutions of the manipulator for all the task points. The multi solution trajectory optimization algorithm (TOA) is depicted by an arrow that connects the input and output blocks. The outputs of the algorithm are a sequence of the desired task points, and the corresponding IK solution that should be used in accomplishing each task.

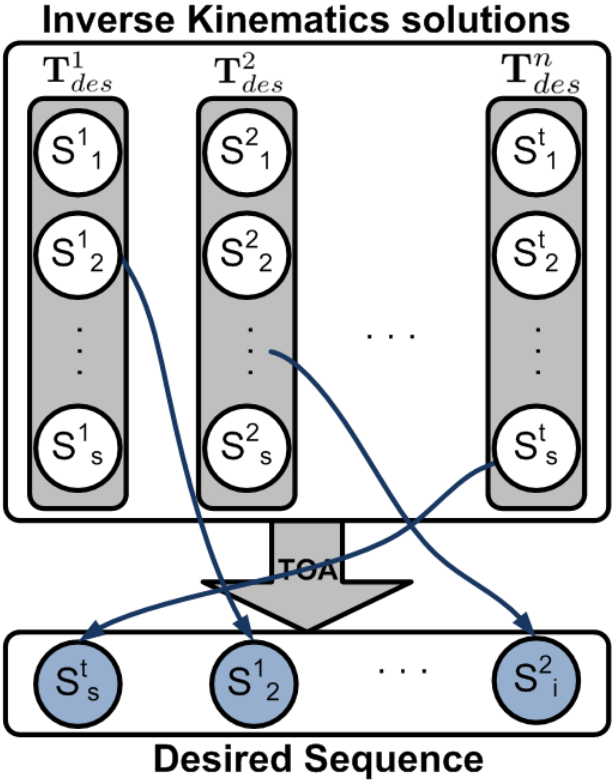


Figure 5.1: Conceptual block diagram of the Trajectory Optimization Algorithm

5.2.2 Size of the Search Space

To develop a method for solving the multi-goal path planning problem, it is beneficial to have an understanding of the size of the search space. For a task set with t task points $t!$ ways of performing the tasks in different sequences exist. If a manipulator has s distinct IK solutions for each of the task points, there are s^t ways of choosing the IK solutions for the task sequence. As a result, there are $t!s^t$ distinct sequences of task points with the corresponding multiple IK solutions. For instance, a PUMA type robot with 6-DOF produces eight distinct IK solutions. If a simple operation of performing six task points is considered, then the size of the search space is 188,743,680. For the same manipulator and for ten task points, the search space size is in the order of 3.8964e+015. The sheer size of the search space calls for an efficient and fast method that replaces the impractical exhaustive search method.

5.2.3 Optimization Criteria

The optimization criteria are the operational metrics of the manipulator and can be translated to a cost function that should be minimized or maximized. The optimization criteria can include one or more of the following measures: shortest collision-free path, minimum required torque, minimum required power, fastest time of performing the task, the most dexterous path of performing the task, and so on.

The cost of moving from the j_1-th IK solution of the i_1-th task to the j_2-th solution of the i_2-th task is denoted as $C(S_{j_1}^{i_1}, S_{j_2}^{i_2})$. $C(S_{j_1}^{i_1}, S_{j_2}^{i_2})$ should be calculated prior to running the proposed trajectory optimization algorithm for all $i_1, i_2 \in [1, t]$ and $j_1, j_2 \in [1, s]$. For each pair of task points, the cost of moving from all of the IK solutions of the first task point to all of the IK solutions of the next should be determined. Thus, for each pair of task points, the cost calculations should be performed s^2 times. The cost calculation is performed after finding the IK solutions of the manipulator for all task points by using one of the algorithms of Chapters 3 or 4.

If the transition costs are stored in an array $\mathbf{C}$ for a task consisting of t task points in a manipulator with s IK solutions, $\mathbf{C}$ have $((t-1).s)^2$ cells and the same number of cost calculations are needed (the cost is not required for moving among the different IK solutions of the same task).

5.2.4 Formulating the Problem

Fig.5.2 is a visual representation of a task with four task points in a manipulator with three distinct IK solutions. Each ellipse represents a task point, and each circle inside the ellipse

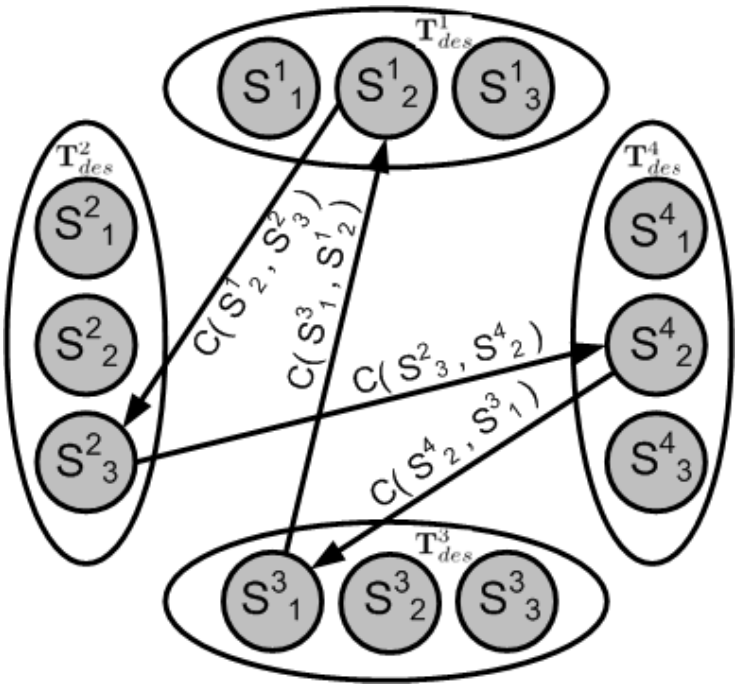


Figure 5.2: Solution of the algorithm

represents a solution of the IK for the task point.

The path, indicated by the arrows, represents a solution of the trajectory optimization problem. In this particular example, the solution path is shown as $[S_2^1\ S_3^2\ S_2^4\ S_1^3]$, i.e., the manipulator should start from the second IK solution of the first task point and move to the third IK solution of the second task point, and so forth. The cost of moving along the arrows is indicated by $C(S_{j_1}^{i_1}, S_{j_2}^{i_2})$ for moving from $S_{j_1}^{i_1}$ to $S_{j_2}^{i_2}$.

It can be observed that the solution path has the following characteristics.

- The path is a closed cycle.
- There is a cost allocated to each move.
- The path passes through all of the task points only once, and on passing through each task point, only one of the IK solutions is visited.
- The desired path has the minimum total cost, $\sum_{C_i \in cycle} C_i$, among all of the possible paths.

By considering these characteristics and assuming that each of the IK solutions is a node in a graph, the problem is reformulated as a graph $G = (V, A)$, where V is the set of nodes and A is the set of arcs. V is clustered into subsets $V_1, V_2, \cdots, V_t$, where V_i is the set containing all of the IK solutions for task point i. Each arc $(S_{s_i}^{t_i}, S_{s_j}^{t_j})$ represents a trajectory from the s_i-th member

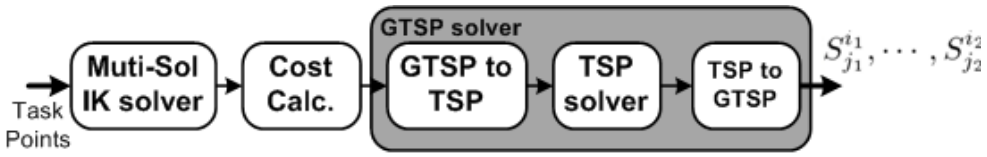


Figure 5.3: Architecture of the proposed algorithm

of V_{t_i} to the s_j-th member of V_{t_j}, i.e., from the s_i-th solution of t_i-th task to the s_j-th solution of t_j-th task. Associated with each arc $(S_{s_i}^{t_i}, S_{s_j}^{t_j})$ is a cost (or distance) defined as $c_{t2,s2}^{t1,s1}$. The goal is to find a closed path (tour) that visits all V_i just once, and upon visiting each V_i passes through just one of the nodes of V_i.

The new formulation and representation of the trajectory optimization problem is now similar to the *Generalized Traveling Salesman Problem (GTSP)* in the combinatorial optimization and is a well researched variant of the famous *Traveling Salesman Problem (TSP)*. To reach the solution of the trajectory optimization problem, the resulting GTSP should be solved.

5.3 Trajectory Optimization Algorithm

In this section, a method for solving the trajectory optimization problem in the newly formulated GTSP form is described. Fig.5.3 reflects the block diagram of the algorithm. The inputs to the algorithm are a set of desired task points. The algorithm consists of three stages.

1. **Multi Solution IK Solver:** The first step in multiple IK solution trajectory optimization is to find the IK solutions. In the first block, the IK problem of the manipulator for all the task points is solved. If there is a closed-form IK solution for the robotic manipulator, all of the solutions can be easily extracted. If the closed-form IK solutions are not conveniently accessible, any of the multi-solution IK solvers in Chapters 3 or 4, can be applied.

   It is noteworthy that even though all of the solutions of the IK are more desirable, by no means, are they all needed. If not all of the solutions of the IK are used, the algorithm finds the best path by using only the available IK solutions.

2. **Cost Calculator:** The solutions of the IK problem are entered into the cost calculator. At this stage, the costs of moving from the IK solutions of all the task points to the IK solutions of the rest are calculated, i.e., $C(S_{j_1}^{i_1}, S_{j_2}^{i_2})$ for all $i_1, i_2 \in [1, t]$ and $j_1, j_2 \in [1, s]$ is calculated. The costs are calculated according to the selected optimization criteria and constitute the matrix $\mathbf{C}$.

3. **GTSP Solver:** By forming the C matrix, the foundation for solving the GTSP is laid. In this stage, the formulated GTSP is be solved by using one of the many current methods.

   The GTSP solvers are categorized into two groups. In the first group, the idea is to solve the GTSP directly. An exact algorithm has been developed for the GTSP by formulating it into an integer programming problem and finding the shortest Hamiltonian circuit [137]. A Lagrangian relaxation algorithm has also been proposed for solving the GTSP [138]. A branch and cut algorithm for the asymmetric version of the problem has also been developed in [139].

   The approach of the second group in solving the GTSP is to transform the GTSP to a TSP, and then solve the resulting TSP by one of the current algorithms. Therefore, a GTSP solver of this approach consists of three phases. In the first phase, the GTSP is transformed to a TSP. Next, the TSP is solved by one of the existing methods such as linear programming [140], Genetic Algorithms [141], Simulated Annealing [142], or Tabu search [143] to name a few. The results of the TSP solver is then converted back to the original GTSP. The GTSP to TSP transformation was first introduced by Lien, in which the number of nodes of the transformed TSP is relatively large, in fact, more than three times the number of nodes in the associated GTSP [144]. Later, another transformation to decrease the size of the corresponding TSP has been presented. In the method, the number of nodes of the TSP is twice the number of the nodes of the original GTSP [145]. Finally, Behzad proposed a transformation in which the number of the nodes are kept constant, hence making the complexity of the transformed GTSP equal to a TSP of the same number of nodes [146].

   The transformation of a GTSP to a TSP with Behzad's method needs negligible processing power and does not change the number of nodes. Furthermore, considering the wealth of easily accessible code for solving TSP, the transformation approach is more attractive. As depicted in Fig.5.3, in the proposed algorithm the second approach is adopted.

5.4 GTSP to TSP Conversion

In this section, Behzad's transformation method, chosen for solving the trajectory optimization problem, is described. As mentioned in Section 5.2.4, $G = (V, A)$ represents a graph with nodes clustered into t distinct subsets of V. Each subset V_i includes the distinct IK solutions of the i-th task point. To convert the original GTSP problem to a TSP, a directed graph G', associated with G, is defined as follows.

1. The nodes of G and G' are the same.

2. All the IK solutions of a certain task t_i in G are connected in a closed cycle in G'. In the directed closed cycle of each task t_i, the successor to the s_i-th IK solution of task t_i, $S^{t_i}_{s_i}$, is represented by $S^{t_i}_{s_i\,(s)}$.

3. The elements of the cost matrix of G', represented by C', are extracted from the original cost matrix C of G as follows

$$\begin{cases} C'\left(S^{t_i}_{s_i}, S^{t_i}_{s_j\,(s)}\right) = 0 \\ C'\left(S^{t_i}_{s_i\,(s)}, S^{t_i}_{s_j}\right) = \infty \\ C'\left(S^{t_i}_{s_i}, S^{t_j}_{s_j}\right) = C\left(S^{t_i}_{s_i\,(s)}, S^{t_j}_{s_j}\right) + M \end{cases} , \tag{5.2}$$

where

$$M > \sum_{\forall t_i, t_j, s_i, s_j} C\left(S^{t_i}_{s_i}, S^{t_j}_{s_j}\right) . \tag{5.3}$$

When a minimum cost TSP tour is found on G' by an existing solver, the solution of the desired GTSP is extracted by connecting the first solution in each cluster which the tour visits. In Fig.5.4, the TSP tour that represents the GTSP solution in Fig.5.2 is illustrated.

5.5 Results

The proposed algorithm is tested on a 3-DOF planar and a 6-DOF PUMA type manipulator. As a reference for validating the results and comparing the performance of the algorithm in term of speed, an exhaustive search method is considered. The exhaustive algorithm finds the minimum cost path by examining all the possible cases one-by-one. The algorithm proposed in Chapter 4 is used as the IK multi-solution solver.

The power required to move from an IK solution of a task point to the next is considered as the cost function. To calculate the cost, seven degree polynomial trajectories are used to connect the starting pose to the end pose. The trajectories are created such that the duration of the motion from the initial point to the end point is constant for all the task point pairs. With the recursive Newton-Euler formulation, the inverse dynamic problem is solved to reach the required torque for the manipulator to follow the trajectory. Finally, from the required torque and the trajectory acceleration, the required power, and thus the cost is calculated.

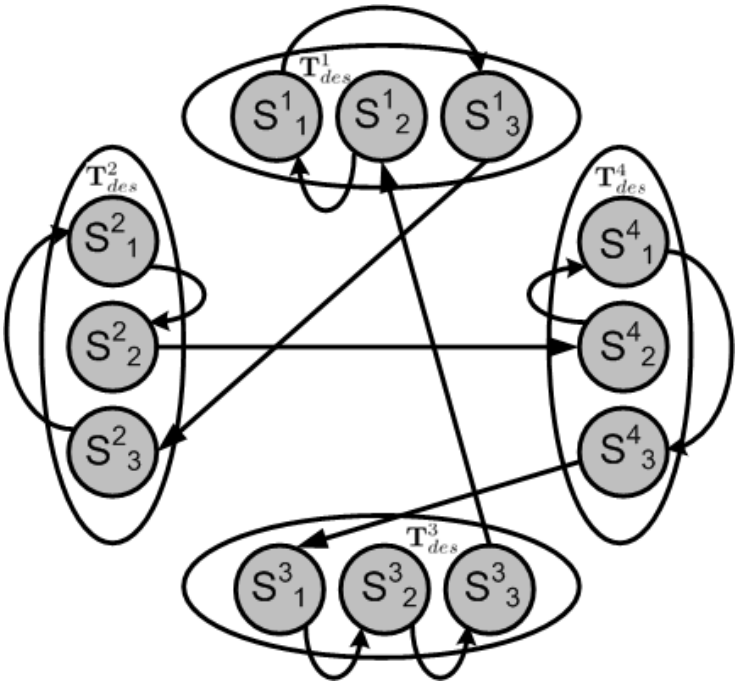


Figure 5.4: TSP tour of G', corresponding to the GTSP tour of G shown in Fig.5.2

To solve the transformed GTSP problem, the TSP is first formulated as an integer programming problem and then a subroutine of the GNU Linear Programming Kit (GLPK) [147] is utilized to solve the resulting problem. GLPK uses the branch-and-bound algorithm as well as Gomory's mixed integer cuts for integer problems [148]. Finally, the results of the TSP are transformed back into the GTSP form, and the final solution of the trajectory optimization is extracted.

5.5.1 3-DOF Planar Manipulator

The 3-DOF planar manipulator is shown in Fig.5.5. A robot with 3-DOF is selected to allow the exhaustive search to perform the search completely and within a reasonable time. The planar robot has two distinct IK solutions for each task point in the Cartesian space. Three distinct tasks, each with three task points are considered. The first four columns in Table 5.1 display the results of the proposed algorithm in relation to the exhaustive search. The end results of both methods are the same, whereas the proposed algorithm is considerably faster. The last two columns indicate how poor performance can be, if multiple IK solutions are not considered. If the absolute worst combination of IK solutions and task sequences are used, the power requirement to complete the task set is at a maximum (column five). Averaged over all the possible IK solution combinations and task sequences, the expected power requirements are provided in column six. The algorithm selects a combination of IK solutions, coupled with task sequences

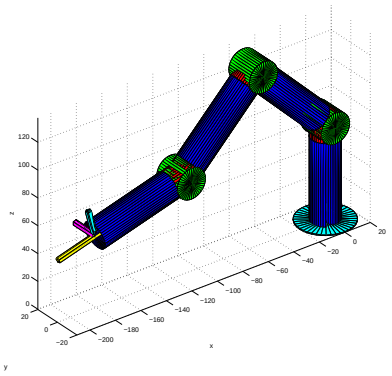


Figure 5.5: Tested 3DOF planar robot

Table 5.1: Comparison between the Exhaustive search and the proposed algorithm

	Proposed Algorithm		Exhaustive Search		Possible Cases	
	Time (s)	Power (KW)	Time (s)	Power (KW)	Maximum Power (KW)	Average Power (KW)
1	6.42	53	44.62	53	181	104
2	6.53	48	47.51	48	150	105
3	7.87	27	44.56	27	139	75

that result in much lower power requirements than both the average and worst case.

The time indicated in the table does not include the time spent on solving the IK for multiple solutions, and only consists of the time spent on calculating the cost and solving the resultant GTSP.

5.5.2 6-DOF PUMA Robot

A 6-DOF PUMA robotic manipulator, similar to the manipulator in Fig.2.12, is considered for a performance comparison of the proposed algorithm, and the exhaustive search when the number of the task points increases. A PUMA robot has eight distinct IK solutions. Table 5.2 lists the run time of the proposed algorithm, and the exhaustive search, in conjunction with the size of the search space as the number of task points increases. The proposed algorithm is significantly faster than the exhaustive search, and the difference between the two methods increases exponentially with the increase in the dimension of the problem. In cases of four or more task points, the exhaustive search algorithm fails to reach a solution within a reasonable time and is stopped.

Table 5.2: Comparison of the run time for the proposed algorithm and the exhaustive search

Number of tasks	Run time (minutes)		Size of search space
	Proposed Algorithm	Exhaustive Search	
2	1.3	2.7	128
3	4.3	103.5	3072
4	8.8	>420	98304
5	14.8	–	3e+6
6	23.5	–	188e+6

5.6 Summary

In this chapter, a method, based on the GTSP model for incorporating multiple IK solutions into the trajectory optimization problem, is presented. The resulting problem is converted to the TSP and solved. It is shown that for a manipulator with eight IK solutions and three task points, the proposed algorithm takes 4% the processing time of an exhaustive search and provides the same solution. In the case of a 3-DOF planar manipulator, a 60% reduction in the average power consumption is achieved compared to the case where only one IK solutions is considered.

All the experiments are performed on Matlab on a 2.4GHz pentium mobile platform. The speed of the algorithm can significantly be improved by implementing the algorithm in a lower level programming language such as C/C++.

To incorporate different optimization criteria, the proposed method is used in conjunction with any point-to-point trajectory optimization algorithm. By formulating the multi IK solution trajectory optimization problem into a GTSP, it benefits from numerous existing TSP solver algorithms. For instance, the resulting TSP can be solved with approximate methods instead of the exact solution method, presented in this work, for a higher speed. The necessity of such an approach is conceivable, since the size of the search space increases exponentially with the increase in the number of task points and IK solutions.

The total cost of the optimized trajectory in performing a certain task is used as a performance measure to compare the two manipulators. Therefore, the optimized total cost can be applied as an optimization criterion in the TBCO in order to seek manipulators which can perform a task with lower costs.

Chapter 6

A Memetic Algorithm for Solving the Task-Based Configuration Optimization Problem

6.1 Introduction

In this chapter, a Memetic Algorithm (MA) for solving the TBCO problem is proposed. In addition to using a numerical local search method, the proposed MA utilizes an elitism and a kinematic structure-aware restarting scheme to further enhance the efficiency of the search. Furthermore, a local search algorithm is proposed to iteratively search for the dimensions of a manipulator with a certain kinematic structure that is capable of performing a certain task. The local search algorithm uses the new Jacobian matrix notation, called the Task Embedded Jacobian, to produce the results.

The rest of this chapter is organized as follows. In Section 6.2, MAs are introduced. Section 6.3 defines the framework of the proposed MA for solving the TBCO problem. A local search algorithm for finding the link dimensions of the manipulators to enable them to perform a set of tasks is proposed in Section 6.4. The proposed MA-based TBCO algorithm is explained in detail in Section 6.5. In Section 6.6, the results of utilizing the proposed TBCO algorithm are given. Section 6.7 summarizes the findings of the chapter.

6.2 Memetic Algorithms

As mentioned in Section 2.7, GAs work on the principle of natural evolution. Specifically, the individuals, which are the coded variables of the problem, evolve to new individuals with better

fitness values due to the operations of crossover, mutation, and selection.

GAs, complemented with the local search algorithms, create a new class of metaheuristics algorithms called Memetic Algorithms (MAs) [149]. The word, *Memetic*, comes from the term, *meme*, which was coined by Dawkins [150] to denote an analogous to *gene* in the context of cultural evolution [151]. The central philosophy of MAs is individual improvement as well as population cooperation and competition, as they are present in many social and cultural systems. In the literature, MAs can be found under a large variety of names such as Hybrid Genetic Algorithms, Genetic Local Searchers, Lamarckian Genetic Algorithms, Baldwinian Genetic Algorithms.

In numerous articles, it has been suggested that the efficiency of a search in pure GAs can be significantly improved by hybridizing them with other techniques [152, 153, 154]. The No-Free-Lunch (NFL) theorem [152], states that a search algorithm strictly performs in accordance with the amount and quality of the problem knowledge it incorporates. Consequently, it can be concluded that an MA, which incorporates the information of the landscape of the proximity of each of the individuals through a local search, performs better than a pure GA without a local search. In essence, the success of MAs can be seen as the tradeoff between the exploration abilities of the GA, and the exploitation abilities of the local search algorithms [155].

Fig. 6.1 and Fig. 6.2 show the pseudocode of a pure GA and an MA in the simplest form, respectively [156]. It is evident that one difference between the two algorithms lies in the fact that MAs have a *local search* stage. It accepts an individual as the input and produces a new individual in the neighbourhood of the original solution, provided that the new solution has a better fitness value. In the literature, a wide range of distinct local search algorithms have been reported. The application requirements and the problem characteristics affect, and sometimes dictates the algorithms for the local search. The wealth of different arrangements and methods for local search and genetic operators in the literature, have given rise to taxonomy methods for classifying the existing MAs [155].

Another difference between the GA and MA, as depicted in Fig. 6.1 and Fig. 6.2, is the *Restart population* element in the MA. Consider a case in which the population is not able to produce new individuals through the genetic operators. Such a situation might occur if the individuals of the current population are very similar to each other. In such a case, the restart stage introduces new individuals into the population. The method for the detection of such a situation and the process of introducing new individuals into the population are application dependent [151].

Although MAs have proven to be effective tools in solving some optimization problems, the process of designing efficient MAs, currently, remains fairly ad-hoc and application dependent [157]. The class of problems in which the fitness function is decomposable, in the sense that

```
// Initialization
Randomly generate a population;
while termination criteria is not reached do
    // Genetic operators
    Calculate fitness value ;
    Selection ;
    Crossover ;
    Mutation ;
end
```

Figure 6.1: Pseudocode of a Genetic Algorithm

```
// Initialization
Randomly generate a population;
while termination criteria is not reached do
    // Genetic operators
    Calculate fitness value ;
    Selection ;
    Crossover ;
    Mutation ;
    // Local search operator
    if population converged then
        Restart population ;
    end
    Local search ;
end
```

Figure 6.2: Pseudocode of a simple Memetic Algorithm

computing the fitness of a solution given the fitness of another solution that is close to it is significantly less computationally expensive than computing the fitness of a solution from scratch, are considered suitable for MAs. The measure of closeness between two individuals can, informally, be defined as the number of common genetic materials they share. Since, in most applications, the calculation of the fitness function accounts for almost all the time spent in a generation of a GA, it seems that MAs have the upper hand when the fitness function is decomposable. This holds true if the improvement of the individuals is performed gradually in small steps [158].

6.3 Framework of the Proposed MA for Solving TBCO

As mentioned in Section 2.5.1, the TBCO can be formulated as a minimization problem in which all the elements of the matrix representation of a manipulator, $\mathbf{m}_n$, should be determined. The solution $\mathbf{m}_n$ should minimize an objective function, and simultaneously satisfy a nonlinear constraint. Therefore, each individual of MA, which represents an $\mathbf{m}_n$ manipulator, can be expressed with the MRR matrix representation as follows

$$\mathbf{m}_n = \begin{bmatrix} 0 & 0 & l_0 \\ \phi_1 & m_1 & l_1 \\ \phi_2 & m_2 & l_2 \\ & \vdots & \\ \phi_n & m_3 & l_n \end{bmatrix} . \tag{6.1}$$

The MRR matrix representation can be decomposed into two characteristically distinct, but interconnected, segments to further investigate the most suitable elements of the matrix representation to undergo a local search. Consequently, the elements which should be identified through the genetic operators can be determined. The matrix is expressed as follows

$$\mathbf{m}_n = \left[\begin{array}{c|c} \Sigma & \Delta \end{array} \right] , \tag{6.2}$$

where Σ and Δ are the kinematic structure and the link dimension matrices of $\mathbf{m}_n$ respectively. The Kinematic structure and the link dimension matrices are defined as follows

$$\Sigma = \begin{bmatrix} 0 & 0 \\ \phi_1 & m_1 \\ \phi_2 & m_2 \\ & \vdots \\ \phi_n & m_3 \end{bmatrix}, \tag{6.3}$$

and

$$\Delta = \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ \vdots \\ l_n \end{bmatrix}. \tag{6.4}$$

In the TBCO, the primary goal is to find both kinematic structure and link dimension matrices, Σ and Δ respectively. Due to the distinctive characteristics of Σ and Δ, the TBCO can be decomposed into a search in two interconnected spaces with distinct characteristics as follows:

Σ search: The kinematic structure matrix, Σ, consists of the first two columns of the MRR representation and includes discrete variables, ϕ_i and m_i for $i = 1..n$. Σ specifies the kinematic structure of the manipulator. If one element of Σ changes, the resulting manipulator can display fundamentally different kinematic characteristics. The reason can be explained by using the concept of manipulator workspace. When the orientation of two consecutive joints or the type of the joints are altered, the workspace of the manipulator might change shape. Since the kinematic parameters of the manipulator changes when Σ changes, the IK of the manipulator, and consequently the optimization criteria and constraints should be computed from scratch.

Δ search: The link dimension matrix, Δ, comprises of the third column of (6.1) with variables l_i for $i = 0..n$ which are bounded continuous variables within the range, $l_i \in [l_{min}, l_{max}]$. Δ defines the dimensions of the links of the manipulator. When Δ of a manipulator is changed, the shape of the workspace is preserved and only the volume is altered, i.e., if Δ is slightly altered, the change in the kinematic parameters of the manipulator is small. As a result, the change in the solutions of the IK problem is small. Therefore, the IK solution of the pre-change manipulator can be used as an initial guess for the IK of the post-change manipulator, enabling the solver to converge faster.

Computing the fitness values, which consist of the optimization criteria and the constraints, requires solutions to the IK problem. Small changes in Δ result in small changes in the IK solu-

tions. Therefore, when Δ changes in a robot, the IK problem of the new manipulator is solved relatively fast by utilizing the IK solutions of the original manipulator as an initial guess. That is, solving the IK problem of a manipulator, given the IK solutions of another manipulator which has a similar Σ but different Δ, is less computationally expensive than computing the IK solutions of the manipulator from scratch. This implies that the fitness function in the TBCO can be considered *decomposable*. The decomposable fitness function entails that the efficiency of GAs can be improved in solving the TBCO when they are hybridized with local search algorithms.

In summary, a local search can provide an effective means for conducting the search in the continuous part of the TBCO, which is Δ. Therefore, in the proposed TBCO, the principal burden of the search in the continuous space of Δ is carried out by the local search operators. The intention of the Δ search is to enhance the reachability error of the manipulator, enabling it to satisfy the constraints. Primarily, the Σ search is conducted by the Genetic Operators. In the next section, the Δ search is mathematically formulated, and a method for solving it is proposed.

6.4 Local Search Operator

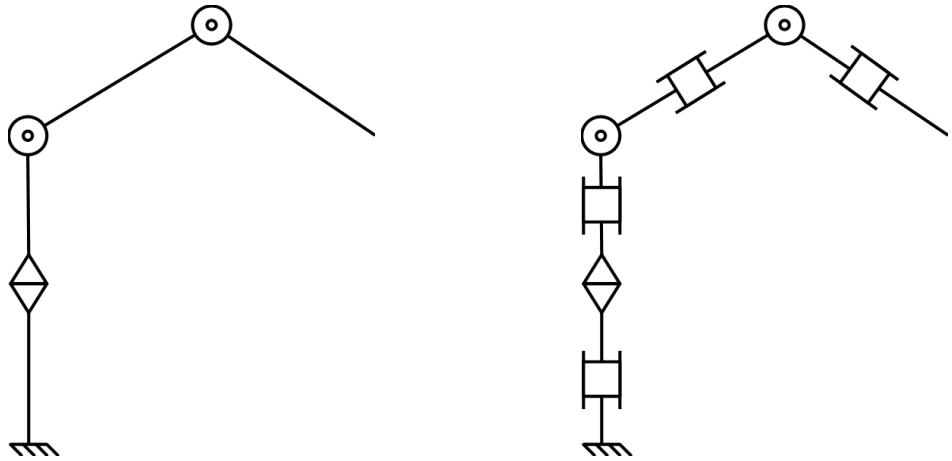
In MAs, the responsibility of the local search is to gradually improve the fitness value of an individual. In the TBCO, the local search modifies the dimension of the links of an individual (manipulator) in order to decrease its reachability error. Since the local search operates on the Δ part of the individuals, it is conducted in the continuous search space. The Δ search is mathematically expressed as follows

$$\begin{cases} \Delta_s &= \arg\min_{\Delta} f_{rch,\mathbb{T}}\left(\mathbf{m}_n, \mathbb{T}\right) \\ \mathbf{m}_n &= [\ \Sigma \mid \Delta\ ] \end{cases}, \tag{6.5}$$

where Δ_s represents the solution of the Δ search problem. Δ is the variable of the problem, and Σ and $\mathbb{T}$ are the parameters. In this section, a method for solving the Δ search problem is proposed.

6.4.1 Δ Local Search

The reason for applying a Δ search to a manipulator is to determine Δ, when Σ is fixed such that the resulting manipulator exhibits an enhanced capability in satisfying the reachability constraints for the desired task $\mathbb{T}$. The reachability error for the i-th task point is a function of the kinematic structure Σ, the link dimensions Δ, and the joint angles of the manipulator Θ_i when the manipulator reaches for the task point. The search for link dimensions is always entangled



(a) (b)

Figure 6.3: Sample $\mathbf{m}_3$ manipulator and the corresponding 7-DOF revolute-prismatic joint manipulator, where the links are considered as prismatic joints: (a) original manipulator; and, (b) resulting 7-DOF manipulator with 3 revolute joints and 4 prismatic joints

with the search for the set of joint angles required to reach the task points. Therefore, with the assumption that Σ is fixed, the variable vector of the Δ search, represented by $q_{\mathbb{T}}$, is defined as follows

$$q_{\mathbb{T}} = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \vdots \\ \Theta_t \\ \Delta \end{bmatrix} . \tag{6.6}$$

To solve for $q_{\mathbb{T}}$, the approach in this thesis is to model each link of the manipulator as a prismatic joint, and then determine the prismatic and rotational joint variables of the new manipulator that allow it to reach all the task points. With this assumption, an $\mathbf{m}_n$ manipulator which consists of n revolute joints is converted to a $(2n+1)$-DOF manipulator with n revolute and $(n+1)$ prismatic joints. In Fig. 6.3, an $\mathbf{m}_3$ manipulator and the corresponding post conversion 7-DOF manipulator with 3 revolute and 4 prismatic joints are shown.

Applying this transformation effectively converts the problem of finding the link dimensions and the corresponding t sets of joint angles in an n-DOF manipulator to the IK problem of a redundant $(2n+1)$-DOF manipulator for t tasks.

Although a wide range of approaches for solving the IK of redundant manipulators exists (Section 2.6.2 contains a review of the prominent methods) , none involves solving the IK problem for numerous task points simultaneously. The majority of methods rely on solving

the first order differential kinematic equations of the manipulator. In the next section, two of the most common approaches to solve this problem, when one task point is considered, are reviewed.

6.4.2 Solving the Inverse Kinematics of Redundant Manipulators

If a task is represented by the desired position of the end-effector in the Cartesian coordinates and a minimal representation of the orientation (such as the Euler angles) as follows

$$x_{des} = \begin{bmatrix} P_{x,des} \\ P_{y,des} \\ P_{z,des} \\ \phi_{x,des} \\ \phi_{y,des} \\ \phi_{z,des} \end{bmatrix} , \tag{6.7}$$

the mathematical formulation of the FK, expressed in (2.23), is written as follows

$$x = K\left(q\right) , \tag{6.8}$$

where q represents the joint variables of a manipulator with both prismatic and revolute joints. (6.8) closely resembles the general FK formulation of (2.23). By differentiating (6.8) with respect to time, the first order differential kinematics equations are obtained, and expressed as

$$\dot{x} = J\left(q\right)\dot{q} , \tag{6.9}$$

where $\dot{x}$ is the task-space velocity vector, $\dot{q}$ is the joint-space velocity vector, and $J\left(q\right) = \partial K / \partial q$ is the $6 \times n$ Jacobian matrix of the manipulator, where n is the number of the joints of the manipulator. For serial manipulators, the i-th and j-th columns of $J\left(q\right)$, corresponding to a revolute and a prismatic joint, respectively, are calculated as the follows

$$J\left(q\right) = \left[\begin{array}{c|c|c|c|c} \cdots & z_i \times P_{i,ee} & \cdots & z_j & \cdots \\ \hline \cdots & z_i & \cdots & \overline{0} & \cdots \end{array}\right] , \tag{6.10}$$

where z_i and z_j represent the axes of the i-th and j-th joint. $P_{i,ee}$ represents the vector which connects the coordinate frame of joint i to the end-effector. In Fig. 6.4, the z_2 and z_3 the joint axes of the second and third joint, rotational and prismatic, respectively, with $P_{2,ee}$ are illustrated.

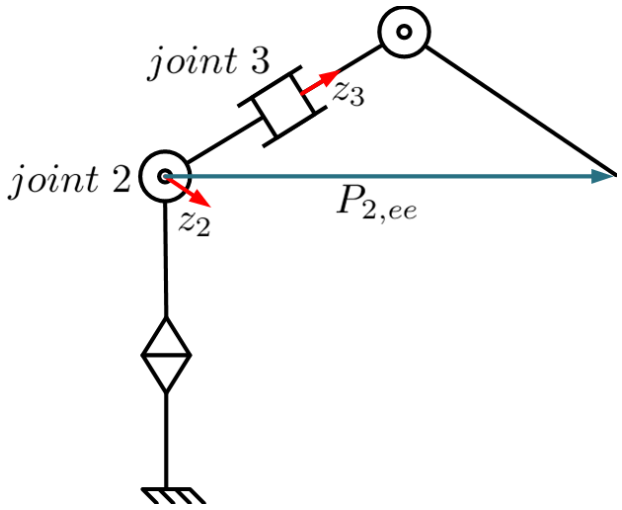


Figure 6.4: z_2, z_3, and $P_{2,ee}$ vectors illustrated on a 4-DOF manipulator with revolute and prismatic joints for the second and third joint

If J is nonsingular, (6.9) can be solved for $\dot{q}$ with the following equation:

$$\dot{q} = J(q)^{-1}\dot{x}\,. \tag{6.11}$$

Under the assumption that the manipulator is kinematically redundant, (6.9) can be solved by resorting to the pseudo-inverse $J^{\dagger}$ of the Jacobian matrix defined as follows [22]:

$$\dot{q} \;=\; J^{\dagger}(q)\,\dot{x} + \left(\mathbf{I} - J^{\dagger}(q)\,J(q)\right)\dot{q}_0\,. \tag{6.12}$$

The pseudoinverse, $J^{\dagger}$, is a unique matrix which satisfies the Moore-Penrose conditions[159]. $\left(\mathbf{I} - J^{\dagger}(q)\,J(q)\right)$ represents the orthogonal projection matrix into the null space of J, and $\dot{q}_0$ is an arbitrary joint space velocity. Consequently, the second part of solution is a null space velocity. The particular solution, in which $\dot{q}_0 = 0$ produces the pseudoinverse solution of (6.9) which is [78]

$$\dot{q} \;=\; J^{\dagger}(q)\,\dot{x}\,. \tag{6.13}$$

To solve (6.13) , a numerical method which updates the value of q in each iteration with the following rule is applied

$$q_k \;=\; q_{k-1} + J^{\dagger}\left(q_{k-1}\right)\ \triangle x_{k-1}\ , \tag{6.14}$$

where $J^{\dagger}\left(q_{k-1}\right)$ is the pseudoinverse of the Jacobian matrix at q_{k-1}, and $\triangle x$ is the distance of the position/orientation of the end-effector at q_{k-1} from the desired task.

Another method for solving the kinematic equations for redundant manipulators, which is more resilient to singular poses, is the damped-least-square method [80, 81]. In this method, instead of solving (6.13), the following first order differential equation is solved:

$$\dot{q} = J^T\left(J^T J + \lambda^2 I\right)^{-1}\dot{x}\ , \tag{6.15}$$

where J^T is the transpose of J, and I is the identity matrix. λ is called the damping factor. It can be recognized that if λ is zero, (6.15) and (6.11) become identical. The update formula for solving (6.15) iteratively is as follows

$$q_k \;=\; q_{k-1} + J^T\left(q_{k-1}\right)\left(J^T\left(q_{k-1}\right)J\left(q_{k-1}\right) + \lambda^2 I\right)^{-1}\ \triangle x_{k-1}\ . \tag{6.16}$$

When the links of a manipulator are converted to prismatic joints, their lengths are transformed into prismatic joint variables. The length of the links of the manipulator remain constant regardless of which task point the robot is reaching. Consequently, the prismatic joint variables which represent the length of the links should remain equal regardless of the task point. Therefore, in the utilized IK solver, an equality set of constraints for the prismatic joints should be considered. Furthermore, the pseudoinverse and damped-least-square methods are developed to solve the IK problem, when only one task point is considered. In the next section, a method for solving the multi-task IK problem with an implicit implementation of the prismatic joint constraints is described.

6.4.3 Task Embedded Jacobian Matrix

When the link dimensions are considered as prismatic joints, the joint variables of the IK of the resulting manipulator, in reaching the i-th task point, is expressed as

$$q_i = \begin{bmatrix} \Theta_i \\ \Delta_i \end{bmatrix}\ , \tag{6.17}$$

where Θ_i and Δ_i represent the revolute and prismatic joint angles of the converted manipulator, respectively. When only the i-th task point is considered, the pseudoinverse solution of (6.17) is written as

$$\dot{q}_i = J_i^{\dagger}(q_i)\,\dot{x}_i\,, \tag{6.18}$$

where J_i is the Jacobian of the converted manipulator at q_i, and is written as follows

$$J_i(q_i) = \left[\, J_i^R(\Theta_i, \Delta_i) \quad J_i^P(\Theta_i, \Delta_i) \,\right]\,, \tag{6.19}$$

where J_i^R represents the Jacobian matrix of the manipulator at q_i where only the revolute joints are considered, and is equal to the Jacobian matrix of the original n-DOF manipulator at Θ_i. J_i^P is the Jacobian matrix of the manipulator at q_i, when only the $(n+1)$ prismatic joints, representing the links of the manipulator, are considered.

For obtaining a solution of the Δ search problem, (6.18) should be concurrently solved for all q_i, when $i = 1..t$. i.e., the set of the following equations should be simultaneously solved

$$\begin{cases} \dot{q}_i = J_i^{\dagger}(q_i)\,\dot{x}_i & \text{for all } i = 1..t \\ \Delta_i = \Delta_j & \text{for all } i, j = 1..t \end{cases}\,, \tag{6.20}$$

where $\dot{q}_i$ and $\dot{x}_i$ represent the velocity vectors in the joint space and the task space, respectively. The second set of equations stipulates that the prismatic joint variables, which represent the link dimensions of the manipulator, should be the same for all the tasks.

To solve (6.20), a Jacobian matrix, called a Task Embedded Jacobian matrix, $J_{\mathbb{T}}$, is formed as follows

$$J_{\mathbb{T}} = \left[\begin{array}{cccc|c} J_1^R(\Theta_1, \Delta) & \overline{0} & \cdots & \overline{0} & J_1^P(\Theta_1, \Delta) \\ \overline{0} & J_2^R(\Theta_2, \Delta) & \cdots & \overline{0} & J_2^P(\Theta_2, \Delta) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \overline{0} & \overline{0} & \cdots & J_t^R(\Theta_t, \Delta) & J_t^P(\Theta_t, \Delta) \end{array}\right]\,. \tag{6.21}$$

By using $J_{\mathbb{T}}$, (6.20) is written as follows

$$\dot{q}_{\mathbb{T}} = J_{\mathbb{T}}^{\dagger}\, \dot{x}_{\mathbb{T}}\ , \tag{6.22}$$

where $q_{\mathbb{T}}$ is defined as (6.6). $\dot{x}_{\mathbb{T}}$ includes all the minimal representation task velocities of the t task points and is defined as follows:

$$\dot{x}_{\mathbb{T}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_t \end{bmatrix}\ . \tag{6.23}$$

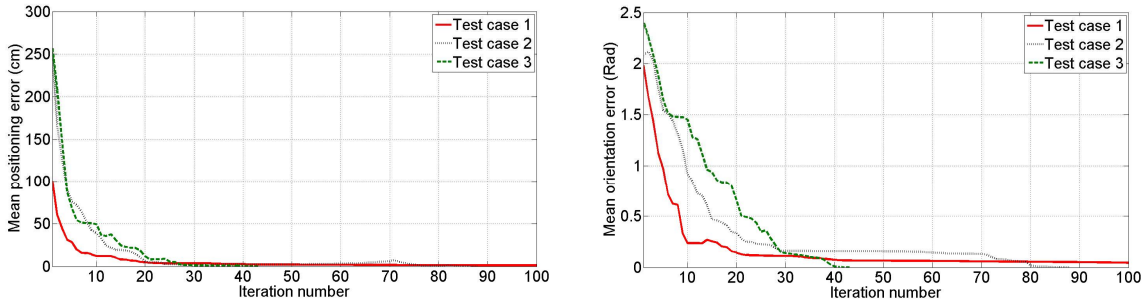
By solving (6.22) for $\dot{q}_{\mathbb{T}}$, the link dimensions Δ and the joint angles of the manipulator Θ_i when the i-th task is reached, are computed concurrently.

Although the Task Embedded Jacobian matrix is explained by applying the pseudoinverse method, to solve (6.22), any of the existing methods in the literature can be utilized. In this thesis, the damped-least-square method is chosen for solving (6.22), due to its resilience to matrix singularities. The Δ, found through the iterative Task Embedded Jacobian matrix method, is the closest solution to the initial guess with which the algorithm is initialized. In more specific terms, depending on $\mathbb{T}$, the Δ search might not have a unique solution. In such a case, the proposed Δ search algorithm converges to one of the solutions that is closest to the initially adopted guess.

6.4.4 Applying the Task Embedded Jacobian Method for Solving the Δ Search

To verify the proposed approach in Section 6.4.3 through numerical analysis, the dimensions of a 6-DOF PUMA type manipulator are determined by using the proposed Δ search. Three distinct tasks, each consisting of 50 task points, are randomly generated with a manipulator with the Σ of a PUMA, but a random Δ. The Task Embedded Jacobian method is selected to determine the dimensions of a manipulator capable of performing the tasks.

The positioning and orienting reachability errors are plotted with respect to the iteration number in Fig.6.5(a) and Fig.6.5(b), respectively. It is evident that, in all the test cases, the positioning and orientation errors converge to within a small margin of zero in the first 40 iterations. To converge to the desired accuracy of 0.1 cm and 0.05 radians, more iterations are needed.



(a) (b)

Figure 6.5: Mean position and orientation reachability errors of the test cases plotted with respect to the iteration number: (a) positioning reachability error (cm) with respect to the iteration number; and, (b) orientation reachability error (rad) with respect to the iteration number

6.5 Memetic Algorithm-Based Task Based Configuration Optimization

The pseudocode of the proposed algorithm is displayed in Fig. 6.6.

The algorithm starts by randomly initializing the population, R_{pop}, in line 1. In line 2, the high priority fitness values of the individuals, which consists of the constraints F_{cons}, are computed. The low priority fitness values, which are the optimization criteria F_{obj}, are calculated in the selection operator as needed. The $while$ loop of line 3 causes the algorithm to iterate between lines 4 to 14, when the termination criteria is not satisfied. In line 4, a kinematic structure-aware elitism method selects a fraction of the fittest individuals to form the population of elites E_{pop}. The population of the elites is added to the next generation of the population without any change. In line 5, the **Restart_Population** subroutine computes a measure of the occurrence of each individual. Then, the individuals with excessive occurrence measures are replaced with fresh randomly generated individuals in order to preserve the diversity of the population. In line 6, the parent pool, P_{pop}, is formed through the selection operator. A new generation is created in lines 7 – 8 from the parent pool by using the crossover and mutation operators. The population of the elites are then added to the new population in line 9. In lines 10 – 13, a part of the population, represented by N_{pop}, is selected to undergo the numerical improvement stage. The individuals, after being improved by the local search (Δ search), are added to the original population, R_{pop}. In line 14, the fitness value of the new population is computed.

Each stage of the pseudocode is explained in more detail in the following section. Table 6.1 lists the name and description of the variables used. Each individual of the population represents a manipulator coded into the matrix representation.

```
1  R_pop = Random_Population_Generation;
2  F_fv = Fitness_Calculation (R_pop) ;
3  while termination criteria is not reached do
4      E_pop = Elitism (R_pop, F_fv);
5      R_pop = Restart_Population (R_pop) ;
6      P_pop = Selection (R_pop, F_fv);
7      C_pop = Crossover (P_pop);
8      C_pop = Mutation (C_pop);
9      R_pop = C_pop ⋃ E_pop ;
10     N_pop = Selection_For_Local_Search (R_pop) ;
11     R_pop = R_pop − N_pop;
12     N_pop = Local_Search (N_pop) ;
13     R_pop = R_pop ⋃ N_pop ;
14     F_fv = Fitness_Calculation (R_pop) ;
15 end
```

Figure 6.6: Pseudocode of the proposed TBCO Memetic Algorithm

Table 6.1: Description of the the variables of the proposed MA

Variables	Description
R_{pop}	Population of manipulators
r	size of population R_{pop}
E_{pop}	Population of elites
e	number of elites
P_{pop}	Population of parents
p	size of the parent pool
C_{pop}	Population of children (Offsprings)
N_{pop}	Population of the selected individuals to undergo local search
g_{max}	Maximum number of iteration after satisfying the constraints
c_i	number of individuals with the same Σ as the i-th individual
Λ_i	Probability of the i-th individual to undergo local search
P_m	Mutation probability
P_c	Crossover probability
f^i_{cons}	Optimization constraint of the i-th individual
f^i_{obj}	Optimization criteria of the i-th individual

6.5.1 Random_Population_Generation

In this stage, a population of $\mathbf{m}_n$ manipulators is created randomly. This population, represented by R_{pop}, acts as the initial population of the MA. The number of individuals in R_{pop}, which is the population size r, for search in the space of n-DOF manipulators, is selected using the following equation:

$$r = \max \{ \; r_{min} \; , \; \kappa_r \; \mathbf{size} \left(\mathbb{M}_n \right) \; \} \; , \tag{6.24}$$

where the population size is set at a fraction $\kappa_r < 1$ of the total number of KCs in $\mathbb{M}_n$, stipulating that the population size should be, at least, r_{min}.

The initial population is formed such that it has the following characteristics.

- Each individual represents a feasible $\mathbf{m}_n$ manipulator. The relative orientation of the joints and the link dimensions are within the permissable ranges, and the joints are selected from the standard modular joint set.
- The population consists of only one individual from each kinematic structure. i.e., two individuals with the same kinematic structure, Σ, can not be found in the initial population. This feature provides the initial population with a more diverse sampling of the KC space.
- The individuals are created such that all are non-redundant. For instance, in an initial population consisting of $\mathbf{m}_4$ manipulators, no manipulator with four parallel joints axes exists.

6.5.2 Fitness_Calculation

In this stage, the fitness values of all the individuals are computed. Since, in the proposed algorithm, a priority-based selection scheme is adopted, the computation of the optimization criterion $f_{obj,\mathbb{T}}$, with lower priorities can be postponed until they are required by the selection scheme. On the contrary, the optimization constraints, $f_{cons,\mathbb{T}}$, should be calculated in each iteration of the algorithm.

6.5.3 While loop

The $while$ loop iterates, until the termination criterion is satisfied. In the proposed method, the loop is repeated for g_{max} generations, after the reachability error is decreased within the desired tolerance, ε_{cons}. i.e., after the first KC capable of performing the task is found, the

algorithm continues for g_{max} generations in order to reach a KC capable of performing the task and minimizing the optimization criteria.

6.5.4 Elitism

Elitism is the process in which the fittest individual(s) of a population are directly transferred to the next population. It has been shown that elitism can improve the efficiency of GAs significantly [111, 112]. Furthermore, for optimization problems in which prior information about the fitness value does not exist, the use of an elitism scheme in MAs has been recommended [160].

In the proposed TBCO algorithm, a kinematic structure-aware elitism scheme is employed. In the elitism scheme, a population E_{pop} with size e, consisting of the fittest individuals, is formed. The individuals of E_{pop} are selected such that E_{pop} would have the following characteristics.

- The members of E_{pop} are the fittest individuals of the population. That is, the individuals of E_{pop} have the lowest $f_{cons,\mathbb{T}}$ among the population. Between two individuals which satisfy the TBCO constraints, i.e., $f_{cons,\mathbb{T}} \leq \varepsilon_{cons}$, the one with a lower $f_{obj,\mathbb{T}}$ is adopted for E_{pop}. ε_{cons} represents the permissable tolerance for the constraint violation.
- E_{pop} consists of individuals with different kinematic structure, Σ. The fittest e individuals with distinct kinematic structure are preserved for the next population. This feature prevents individuals with the same Σ to be transferred to the next population. Consequently, the chance of the population being dominated by a few kinematic structures decreases.

6.5.5 Restart_Population

In MAs, keeping the population diverse is always an issue [157]. It has been suggested that by computing a measure of the diversity for the population and injecting new individuals into the population when the diversity drops below a certain threshold, the problem of maintaining the diversity can be addressed [151].

In the proposed TBCO, the inverse of the number of manipulators with a similar kinematic structure, Σ, is adopted as a diversity measure. i.e., when in a population, the number of the manipulators with the same Σ increases, the diversity of the population decreases.

Therefore, the following restart scheme is devised, where c_i (for $i = 1 \cdots r$) represents the number of manipulators with similar kinematic structures to that of the i-th individual.

1. A list of the individuals with $c_i \geq \kappa_c.r$, where $\kappa_c < 1$ is a constant, is created. Therefore, the list consists of the manipulators which have repeated occurrences, in terms of the kinematic structure Σ, with a total number of occurrences of $\kappa_c.r$ or more.

2. Half of the individuals on the list which are less fit are replaced by individuals created from scratch (using $\mathbf{Random\_Population\_Generation}$), and the other half which are fitter are returned to the population.

To summarize, the proposed $\mathbf{Restart\_Population}$ scheme can replace the more recurred individuals of the population in terms of the kinematic structure with randomly generated new individuals.

6.5.6 Selection

In the selection, a group of individuals are chosen to form a parent pool. The members of the parent pool then have the chance to produce a new population through crossover, mutation, and the local search operators.

In TBCO, the optimization criteria and the constraints are mapped into two sets of fitness values with different priorities. Then, the selection can be formulated into a priority-based selection in which two individuals are compared such that the constraints have more impact than the optimization criterion.

To implement a priority-based selection scheme, a binary tournament selection is chosen. In the tournament selection, to select a parent, n_{tour} individuals are selected randomly. From these individuals, the fittest individual is transferred to the parent pool. If the parent pool, P_{pop}, consists of p individuals, a total of p tournaments should be performed for forming P_{pop}.

An advantage of the binary tournament, in which $n_{tour} = 2$, is that it produces a larger selection variance than that of the ranking selection scheme, which is a more commonly used selection method in the GA literature. Choosing $n_{tour} \geq 2$ decreases the chance of weaker individuals being selected, and subsequently, decreases the diversity of the parent pool, but increases the convergence speed [161].

Another advantage of using the binary tournament selection in the TBCO is that it reduces the process of selection into a simple comparison between two individuals. The winner of the tournament can be decided by comparing any characteristic or measure of the individuals involved in the tournament. This feature can be exploited for developing priority-based selection operators. In such an operator, the individuals are compared according to a set of fitness values with different priorities.

Fig. 6.7 reflects the pseudocode of the proposed priority-based selection scheme, where δ_1, δ_2, and ζ are constants such that $\delta_1 < \delta_2$. R_1 and R_2 are two individuals randomly selected from the population for the binary tournament, and R_{out} is the winner of the tournament which is added to the parent pool. $F^1_{cons,\mathbb{T}}$ and $F^2_{cons,\mathbb{T}}$ represent the optimization constraint computed for the task, $\mathbb{T}$, and $F^1_{obj,\mathbb{T}}$ and $F^2_{obj,\mathbb{T}}$ represent the computed optimization criteria for R_1 and R_2, respectively.

According to the pseudocode, when R_1 and R_2 are being compared, three distinct cases occur.

1. If the optimization constraints of R_1 and R_2 are very close to each other, within δ_1, the one with the better $F_{obj,\mathbb{T}}$ is the winner.

2. If the optimization constraints of R_1 and R_2 are fairly close, within δ_2, the one with the better $F_{obj,\mathbb{T}}$ with a certain probability ζ is the winner.

3. If the difference between the optimization constraints of R_1 and R_2 is large, the one which can better satisfy the optimization constraints, with a smaller $F_{cons,\mathbb{T}}$, is selected as the winner.

It can be observed that the priority-based scheme considers $f_{obj,\mathbb{T}}$, only if R_1 and R_2 have approximately the same $f_{cons,\mathbb{T}}$, considering $f_{obj,\mathbb{T}}$ with a lower priority. Moreover, since the computation of $f_{obj,\mathbb{T}}$ is not required in the third case, its computation is postponed, until it is absolutely necessary.

6.5.7 Crossover

To produce two new individuals, called offsprings, two parents are randomly selected from the parent pool to undergo the crossover. The two parents are crossed with a probability of P_c, and otherwise, are transferred, unchanged, to the new population. This process is repeated until a new population with $(r - e)$ individuals is created.

In the TBCO, each individual consists of two segments, kinematic structure Σ and link dimension Δ, and three types of variables, corresponding to the three columns of the MRR matrix representation, namely the relative orientation of the joints ϕ_i, joint types m_i, and link lengths l_i. The methods by which the crossover is applied to each of the variables, differ due to the differences in the variable type and the permissible bounds. For ϕ_i and m_i, the crossover is applied in a discrete space with two distinct permissible bounds, and for l_i, the crossover is applied in a bounded continuous space.

Input: R_1 and R_2: Randomly selected individuals for tournament
Output: R_{out}: Winner of the tournament selection

```
if (|F^1_{cons,T} − F^2_{cons,T}| ≤ δ_1) then
    if F^1_{obj,T} ≤ F^2_{obj,T} then
        R_out = R_1
    else
        R_out = R_2
    end
else if (|F^1_{cons,T} − F^2_{cons,T}| ≤ δ_2) and rand ≤ ζ then
    if F^1_{obj,T} ≤ F^2_{obj,T} then
        R_out = R_1
    else
        R_out = R_2
    end
else
    if F^1_{cons,T} ≤ F^2_{cons,T} then
        R_out = R_1
    else
        R_out = R_2
    end
end
```

Figure 6.7: Pseudocode of the proposed priority-based selection scheme

Since GeneAs framework [114] provides a method for performing the crossover in individuals, consisting of distinct variable types, the framework is adopted as the crossover scheme. According to GeneAS, the crossover operator is applied to approximately 50% of the variables constituting the individuals. The utilized crossover operator is consistent with the corresponding variable type.

Fig. 6.8 denotes the pseudocode of the crossover scheme. P_1 and P_2 are the selected parents from the parent pool, whereas C_1 and C_2 are the offsprings produced from crossing P_1 and P_2. The variable at the i-th row and j-th column of the MRR matrix representation is expressed by $X\,(i,j)$, for all $i = 1..n$ and $j = 1..3$, where X can be P_1, P_2, C_1, or C_2. Each $X(i,j)$ has a 50% chance of undergoing the crossover. The crossover is applied to each element, according to the variable type of the element. The crossover operator used for the link lengths, l**_crossover** is the bounded SBX operator, as explained in Section 3.3.2, and ϕ**_crossover** and m**_crossover** uses a discrete version of the bounded SBX which creates only the permissible values of ϕ and m.

Input: P_1 and P_2: Randomly selected individuals for crossover
Output: C_1 and C_2: The offsprings

```
1  forall i = 1..n, j = 1..3 do
2      if rand ≤ 0.5 then
3          switch j do
4              case j = 1
                   // X(i,j) is joint relative orientation
5                  [C1(i,j)  C2(i,j)] = φ_crossover( P1(i,j)  P2(i,j) )
6              end
7              case j = 2
                   // X(i,j) is joint type
8                  [C1(i,j)  C2(i,j)] = m_crossover( P1(i,j)  P2(i,j) )
9              end
10             case j = 3
                   // X(i,j) is link length
11                 [C1(i,j)  C2(i,j)] = l_crossover( P1(i,j)  P2(i,j) )
12             end
13         end
14     end
15 end
```

Figure 6.8: Pseudocode of the GeneAS-based crossover scheme

6.5.8 Mutation

With a probability of P_m, the crossover results may undergo the mutation operator. In the proposed TBCO, the mutation is performed as depicted in the pseudocode of Fig. 6.9, where R_{in} and $R_{mutated}$ represent a selected individual to undergo mutation and the mutated individual, respectively. According to the pseudocode, first, an element of individual R_{in} is randomly selected. Based on the type of the selected element, the appropriate type of mutation is employed.

If the selected element is a link length l_i, a continuous mutation operator [114] is performed. In order to mutate the link lengths such that the mutated value $l_{i,mutated}$ remains in the bound, $[\, l_{min}\ ,\ l_{max}\,]$, the mutation is carried out by the following steps.

1. A random number u in the range $[0, 1]$ is created.

2. $\triangle_{max}$ is computed by using the following equation:

$$\triangle_{max} = \begin{cases} |l_i - l_{min}| & \text{, if } u < 0.5 \\ |l_{max} - l_i| & \text{, if } u \geq 0.5 \end{cases} . \tag{6.25}$$

3. $\overline{\delta}$ is calculated as follows:

$$\overline{\delta} = \begin{cases} (2u)^{\frac{1}{\eta+1}} - 1 & \text{, if } u < 0.5 \\ 1 - [2\,(1-u)]^{\frac{1}{\eta+1}} & \text{, if } u \geq 0.5 \end{cases} . \tag{6.26}$$

4. The mutated link length is determined by computing:

$$l_{i,mutated} = l_i + \overline{\delta}\triangle_{max}\,. \tag{6.27}$$

The aforementioned process produces a mutated link length by using a polynomial probability distribution with the mean at the original link length and the variance a function of η. The maximum and minimum of the mutated value are dictated by Δ_{min} and Δ_{max}. An η between two and five produces a mutation which simulates the binary mutation [114].

If the selected variable is a joint relative orientation ϕ_i or a joint type m_i, a discrete version of the mutation operator used for link lengths is used.

6.5.9 Selection_For_Local_Search

In the stage, N_{pop}, the list of the individuals that are selected to undergo the local search, is formed. The number of individuals in N_{pop} depends on the desired frequency, which the local

Input: R_{in}: Individual to undergo mutation
Output: $R_{mutated}$: Mutated individual

```
1  i = Random_integer(1..n) ;
2  j = Random_integer(1..3) ;
3  switch j do
4      case j = 1
           // X(i,j) is joint relative orientation
5          R_mutated(i,j) = φ_mutation(R_in(i,j))
6      end
7      case j = 2
           // X(i,j) is joint type
8          R_mutated(i,j) = m_mutation(R_in(i,j))
9      end
10     case j = 3
           // X(i,j) is link length
11         R_mutated(i,j) = l_mutation(R_in(i,j))
12     end
13 end
```

Figure 6.9: Pseudocode of the mutation operator

search should be applied to the population. For the proposed TBCO algorithm, an adaptive local search frequency scheme, called the complete method [160], is adopted. The goal of the adaptive local search frequency methods is to reduce the number of local searches, and hence the computational cost. To do so, the chance of undergoing the local search for similar individuals, or the individuals which are clustered close to each other, is reduced. In the complete method, the i-th individual is added to N_{pop} with a probability, Λ_i. Λ_i is computed as follows:

$$\Lambda_i = \frac{\Lambda}{N_i} , \tag{6.28}$$

where N_i is the number of individuals in the population which represent identical solutions to the problem. In the proposed TBCO, N_i represents the number of individuals with the same Σ as the i-th individual, and therefore, $N_i = c_i$. Λ is a constant which represents the probability of the local search, when $N_i = 1$.

6.5.10 Local_Search

In this stage, the Δ search is applied to the members of N_{pop}. Since a gradual local improvement of the individuals decreases the computational cost of the local search in each generation, and simultaneously, increases the resilience of the algorithm to premature convergence, a method for limiting the number of iterations of the Δ search is employed.

As shown in the numerical analysis of Section 6.4.4, in the first few iterations of the Δ search, the reachability error decreases substantially. After the initial fall, the reachability error decreases gradually. To achieve a tangible decrease in the reachability error, if the reachability error is small, more iterations are required than if the reachability error is high. Therefore, the number of iterations ρ is determined, adaptively, for each individual according to its reachability error as follows

$$\rho_i = \begin{cases} \rho_L & \text{, If } F^i_{rch,\mathbb{T}} \leq \delta_{LS} \\ \rho_H & \text{, else} \end{cases} , \tag{6.29}$$

where the number of iterations of the local search for individual R_i is equal to constant ρ_L, if the reachability error is less than or equal to a threshold, δ_{LS}, otherwise the number of iterations is equal to ρ_H, where $\rho_L \leq \rho_H$. By using this scheme, the local search is applied to individuals with smaller reachability errors more aggressively.

Table 6.2: MA parameters in the numerical analyses of the results section

Parameter	Value
r_{min}	30
p	$\lfloor 0.5r \rfloor$
e	$\lfloor 0.1p \rfloor$
κ_r	$1/8$
g_{max}	20
ε_{cons}	$0.1t$
κ_c	$1/3$
δ_1	10^2
δ_2	10^3
ζ	0.2
η	2
Λ	1
ρ_L	20
ρ_H	50
δ_{LS}	$10^3\varepsilon_{cons}$

6.6 Results

In this section, the results of the numerical analysis of the proposed TBCO algorithm are discussed. In all the test cases of this section, the total reachability error is considered as the optimization constraint. The minimum required power to execute the task, as calculated by the trajectory optimization method of Chapter 5, is defined as the optimization criterion. Since the trajectory optimization algorithm requires all the solutions of the IK problem, the IK solver of Chapter 4 is used to determine the solutions.

Table 6.2 lists the value of the parameters of the MA for the numerical analysis of this section, where $\lfloor x \rfloor$ represents the nearest integer less than or equal to x.

6.6.1 Performance Test

Test Case 1: MA versus GA for solving the TBCO

In this test case, the TBCO problem in the $\mathbb{M}_3$ space for five tasks, each consisting of four task points, is solved by the proposed MA and a pure GA version of the algorithm. The GA version shares the genetic operators (**Crossover** and **Mutation**), and the elitism (**Elitism**) subroutines with the MA, but does not include **Restart_Population**, **Selection_For_Local_Search**, and **Local_Search** subroutines. Since the idea is to compare how fast each algorithm finds a manipulator capable of satisfying the constraints, no optimization criteria are considered, and only the third case of the selection operator, in which the constraints are compared, is implemented.

In Fig. 6.10, the minimum and average reachability errors of the population, with respect to the generation number for MA and GA are plotted. In both algorithms, the maximum permissible generation number is 40. The MA can find a manipulator capable of performing the task in all the test runs. In three of the runs, such a manipulator is found in the first generation of the MA. For the GA, none of the runs reaches a manipulator capable of satisfying the constraints.

Table 6.3 summarizes the operational measures of the test runs for the MA and GA. For the GA, the lowest reachability error belongs to a manipulator in the last generation of the third test run. Since the reachability error for the manipulator is 8.7758, the manipulator cannot perform the task with the desired precision. The MA finds a manipulator with a reachability error within the permissible tolerance, ε_{cons}, in all the test runs. However, the average generation run time of the MA is approximately three times the generation run time of the GA. The reason is that in each generation of the MA, a fraction of the population undergoes the local search algorithm. Although the generation run time of the GA is smaller than that of the MA, at the final generations, the total run time of the MA is less than the GA. The fact that, in the final generation, the MA has already found a good solution, whereas the GA is unable to determine a manipulator, capable of performing the task, shows the superior performance of the novel MA in finding manipulators capable of satisfying the constraints, i.e., performing the task.

Test Case 2 : The Effect of the Number of Task Points

To investigate the effect of increasing the number of task points on the performance of the proposed MA, the TBCO in the $\mathbb{M}_3$ and $\mathbb{M}_5$ space with varying number of tasks are solved.

Fig. 6.11 shows the minimum and average reachability errors, plotted with respect to the generation number for each search in $\mathbb{M}_3$, when the number of task points varies from 5 to 20. Since, in the ten task point case, a solution is reached in the first generation, the minimum and

Table 6.3: Performance measures of the MA and GA

Method	Test Run Number	Final Generation Number	Minimum Reachability Error	Average Generation Time (minutes)	Total Time (minutes)
GA	1	40	111.7028	4.93	197.41
	2	40	46.9296	4.38	175.40
	3	40	8.7758	2.99	119.62
	4	40	54.0351	4.48	179.40
	5	40	48.6138	3.98	159.22
MA	1	1	0.0643	14.66	14.66
	2	1	0.0254	13.52	13.52
	3	1	0.0204	13.57	13.57
	4	8	0.0334	11.43	91.45
	5	5	0.0369	12.73	63.67

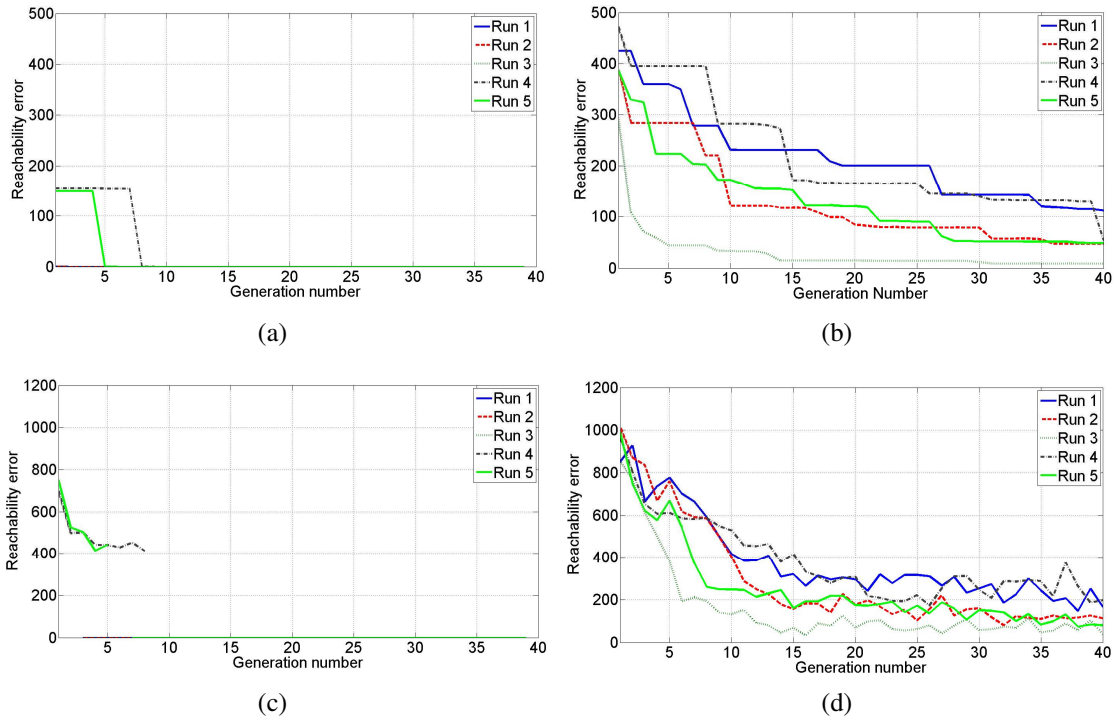
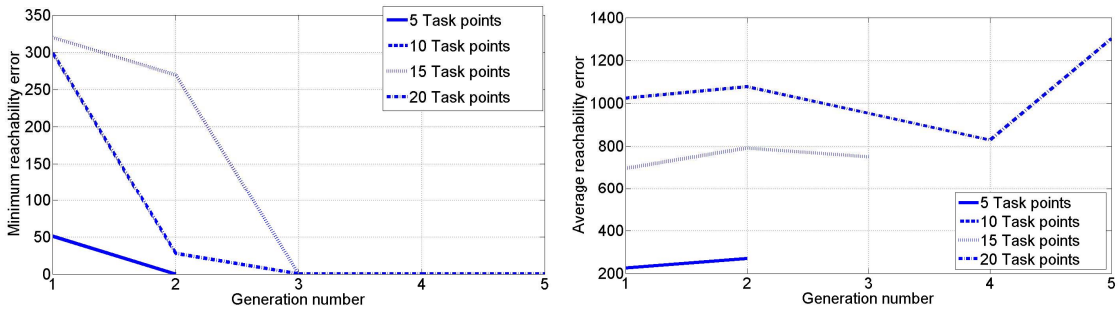


Figure 6.10: Comparison of the MA and GA: (a) MA minimum reachability error; (b) GA minimum reachability error; (c) MA average reachability error; and, (d) GA average reachability error

Table 6.4: MA generation number in which a KC, capable of performing the task, is reached

Search Space	5 Task Points	10 Task Points	15 Task Points	20 Task Points
$\mathbb{M}_3$	2	1	3	5
$\mathbb{M}_5$	9	6	4	16



(a) (b)

Figure 6.11: Effect of the number of task points in $\mathbb{M}_3$ TBCO: (a) minimum reachability error of the population; and, (b) average reachability error of the population

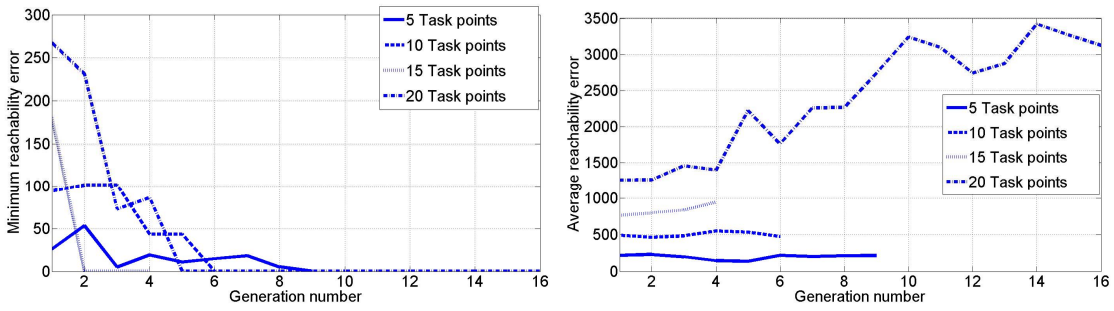
average reachability errors of the test are not visible on the plot. It is obvious that the algorithm finds a KC, capable of performing the task, In all the test cases.

In Fig. 6.12, the results of the same test for a search in $\mathbb{M}_5$ are plotted. Here, the algorithm has reached a solution in generation 9 for the 5 task point case, and a solution in generation 2 of the 15 task point case.

Table 6.4 summarizes the generation numbers in which a KC, capable of performing the task is reached. It seems that the generation, in which a manipulator capable of performing the task is found, is directly affected by the following factors:

- **The closeness of the individuals of the initial population to a KC capable of performing the task.** The measure of the closeness can be considered as the number of genetic operators needed to produce a KC with a kinematic structure which can be improved by the Δ search to perform the task.

- **The generation number in which the local search is applied to a potential solution.** In each generation for all the individuals, the probability for undergoing the Δ search is computed. Even though an individual close to a solution might exist in the population, it might not have the chance to undergo the Δ search until further generations.



(a) (b)

Figure 6.12: Effect of the number of task points in $\mathbb{M}_5$ TBCO: (a) minimum reachability error of the population; and, (b) average reachability error of the population

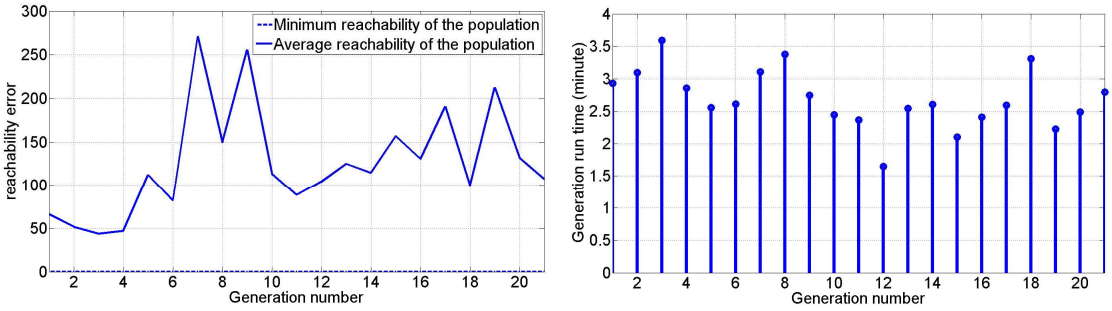
6.6.2 TBCO in $\mathbb{M}_5$

Test Case 1

In this test case, the proposed TBCO algorithm is applied for conducting the search in the $\mathbb{M}_5$ space for manipulators that are capable of performing a $\mathbb{T}_2$ task. To ensure that the all the task points are reachable with an $\mathbf{m}_5$ manipulator, the task points are selected randomly from the workspace of the following $\mathbf{m}_5^{ref}$ manipulator:

$$\mathbf{m}_5{}^{ref} = \begin{bmatrix} 0 & 0 & 60 \\ 0 & R & 30 \\ 0 & P & 30 \\ 0 & R & 10 \\ 0 & P & 10 \\ 0 & R & 10 \end{bmatrix}. \tag{6.30}$$

The minimum and average reachability errors, and the generation run time of the TBCO, applied to this problem are plotted with respect to the generation number in Fig. 6.13. According to Fig. 6.13(a), the first manipulator capable of performing the task is found in the first generation, but the run is not terminated in order to search for KCs with lower power requirements for executing the task. Fig. 6.13(b) illustrates the run time of the algorithm in each generation. The variation, which can be observed in the run time from one generation to another, is due to the stochastic nature of the genetic selection operator and the selection for numerical operators. The generations, in which the selection operator needed to compute F_{obj} more often, require more time. Another factor affecting the run time is the number of individuals in the generation which undergo the Δ search operator. In diverse populations, in which c_i is relatively small, Λ_i is higher, and the numerical search is applied to a higher fraction of the individuals, increasing



(a) (b)

Figure 6.13: Minimum and average reachability errors of the population and the generation run time for a $\mathbb{M}_5$ search using the proposed MA, test case 1: (a) minimum and average population reachability errors; and, (b) generation run time

the run time of the generation.

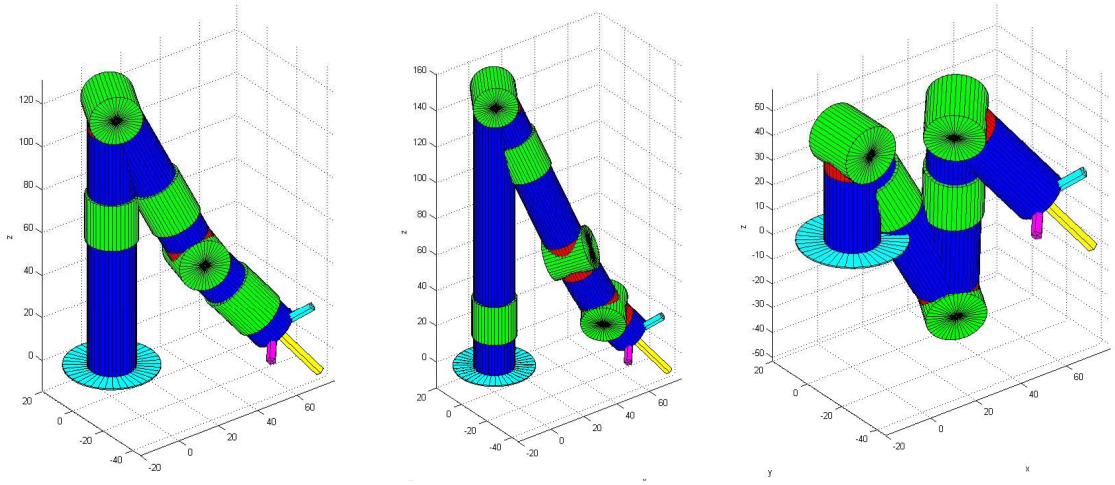
In the last generation of the MA, two KCs, $\mathbf{m}_5^{KC1}$ and $\mathbf{m}_5^{KC2}$, capable of performing the task, exist. The KC of $\mathbf{m}_5^{ref}$ with $\mathbf{m}_5^{KC1}$ and $\mathbf{m}_5^{KC2}$, when they reached the first task point, are signified in Fig. 6.14. The matrix representation of the solutions are as follows:

$$\mathbf{m}_5{}^{KC1} = \begin{bmatrix} 0 & 0 & 17.8 \\ 0 & R & 101.6 \\ 0 & P & 14.5 \\ 0 & R & 41.0 \\ 0 & P & 20.6 \\ \pi/2 & P & 10.0 \end{bmatrix}, \tag{6.31}$$

and

$$\mathbf{m}_5{}^{KC2} = \begin{bmatrix} 0 & 0 & 28.7 \\ 0 & P & 6.4 \\ 0 & R & 42.1 \\ 0 & P & 32.7 \\ 0 & R & 13.7 \\ 0 & P & 36.8 \end{bmatrix}. \tag{6.32}$$

Table 6.5 summarizes the operational measures of the two KCs. Although the position and orientation reachability errors of both manipulators indicate that they can reach the task points with a high precision, $\mathbf{m}_5^{KC1}$ is the final solution of the TBCO due to the lower power requirements in performing the task.

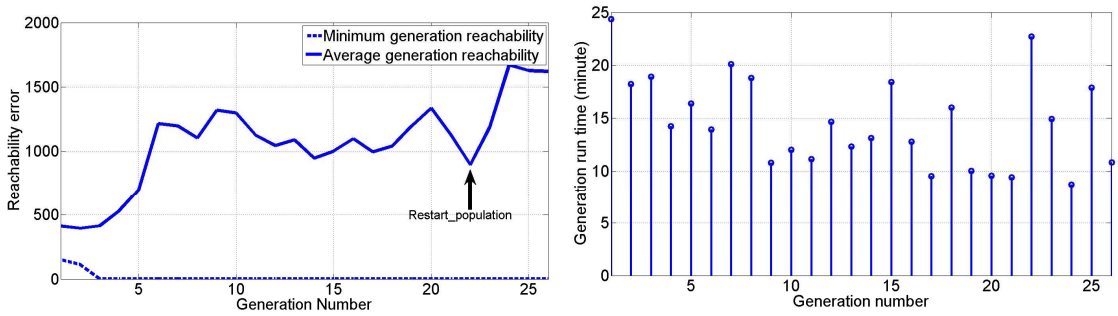


(a) (b) (c)

Figure 6.14: KCs of $\mathbf{m}_5^{ref}$ and the two outputs of the MA-based TBCO algorithm in test case 1: (a) $\mathbf{m}_5^{ref}$; (b) $\mathbf{m}_5^{KC1}$; and, (c) $\mathbf{m}_5^{KC2}$

Table 6.5: Results of the MA-based TBCO in $\mathbb{M}_5$, test case 1

Kinematic Configuration	Position Reachability Error (cm)	Orientation Reachability Error (Rad)	Required Power (KW)
$\mathbf{m}_5^{KC1}$	0.0006	0.0000	0.143
$\mathbf{m}_5^{KC2}$	0.0007	0.0000	0.301



(a) (b)

Figure 6.15: Minimum and average reachability errors of the population and the generation run time for a $\mathbb{M}_5$ search using the proposed MA, test case 2: (a) minimum and average population reachability errors; and, (b) generation run time

Test Case 2

In this test case, the proposed TBCO algorithm is used to determine a manipulator capable of performing a $\mathbb{T}_{10}$ task. The task points are randomly selected from the workspace of $\mathbf{m}_5^{ref}$.

In Fig. 6.15(a) the minimum and average reachability errors of the population are plotted with respect to the generation number. According to the minimum reachability error, the first manipulator, capable of performing the task, is found in generation six. To perform the search for the manipulators with a higher performance, in terms of the required power, the algorithm is not terminated in 20 more generations. As illustrated in the figure, in generation 22, the **restart_population** subroutine has substituted a part of the population with new individuals.

Fig.6.16(a) shows the KC of $\mathbf{m}_5^{ref}$ for the first task point. In the final population of the MA, two KCs $\mathbf{m}_5^{KC3}$ and $\mathbf{m}_5^{KC4}$, that are capable of performing the task are found. The matrix representation of $\mathbf{m}_5^{KC3}$ and $\mathbf{m}_5^{KC4}$ are

$$\mathbf{m}_5{}^{KC3} = \begin{bmatrix} 0 & 0 & 21.3 \\ 0 & R & 68.6 \\ \pi/2 & P & 17.7 \\ 0 & R & 22.2 \\ 0 & P & 3.8 \\ 0 & R & 16.1 \end{bmatrix}, \tag{6.33}$$

and

Table 6.6: Results of the MA-based TBCO in $\mathbb{M}_5$, test case 2

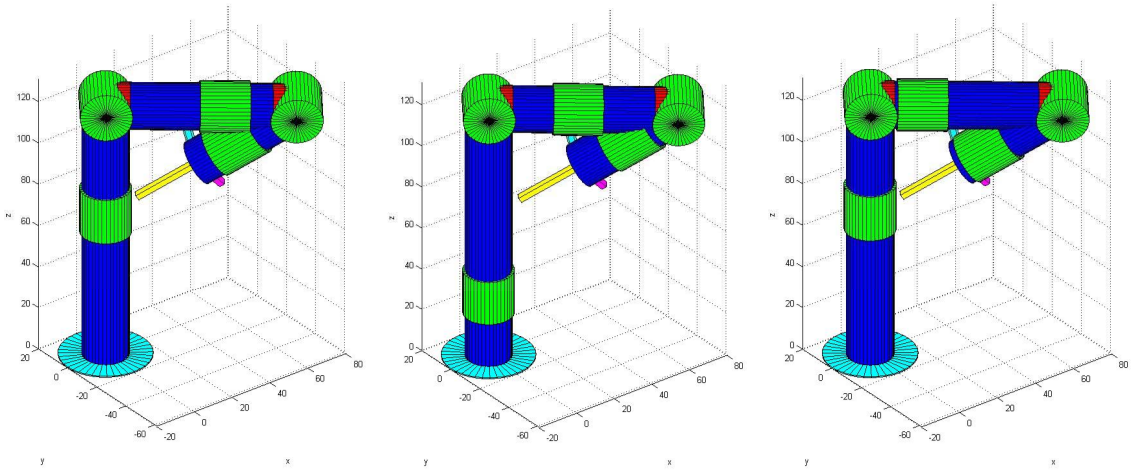
Kinematic Configuration	Position Reachability Error (cm)	Orientation Reachability Error (Rad)	Required Power (KW)
$\mathbf{m}_5^{KC3}$	0.0094	0.0004	2.2007
$\mathbf{m}_5^{KC4}$	0.0068	0.0006	3.0350

$$\mathbf{m}_5{}^{KC4} = \begin{bmatrix} 0 & 0 & 61.2 \\ 0 & R & 28.7 \\ 0 & P & 1.7 \\ 0 & R & 38.2 \\ \pi/2 & P & 16.6 \\ 0 & R & 3.3 \end{bmatrix}. \tag{6.34}$$

Table 6.6 lists the reachability errors and the required power to perform the task for $\mathbf{m}_5^{KC3}$ and $\mathbf{m}_5^{KC4}$. The position and orientation measure for both manipulators are within the permissible tolerance. Therefore both manipulators are capable of performing the tested $\mathbb{T}_{10}$. Fig. 6.16(b) and Fig. 6.16(c), show $\mathbf{m}_5^{KC3}$ and $\mathbf{m}_5^{KC4}$, respectively. Although the kinematic structure matrix, Σ, of $\mathbf{m}_5^{ref}$, $\mathbf{m}_5^{KC3}$ and $\mathbf{m}_5^{KC4}$ are distinct by changing the joint reference angle of the rotation joints 1 and 3, it can be shown that they all refer to a similar kinematic structure, i.e., the workspace of all the three manipulators has the same shape. Furthermore, although the Δ of the three manipulators are not equal, since the sum of the links connected to the input and output mechanical ports of rotational joints 1, 3, and 5 are equal, the volume (size) of their workspace is also equal. This could be an indication that only a KC with a workspace similar to $\mathbf{m}_6^{ref}$ can perform the desired $\mathbb{T}_{10}$ task. In such a KC, although the general kinematic structure should remain the same to perform all the tasks, the length of the links, connected to input and output mechanical ports of the rotational joints, can be changed such that the total length remains constant. The required power for performing the task are 2.2 KW and 3.0 KW for $\mathbf{m}_5{}^{KC3}$ and $\mathbf{m}_5{}^{KC4}$ respectively. Therefore, the final solution of the TBCO is $\mathbf{m}_5{}^{KC3}$ which can perform the task with the least required power.

6.6.3 TBCO in $\mathbb{M}_6$

In this test, the TBCO is applied for finding an $\mathbf{m}_6$ manipulator for performing a $\mathbb{T}_5$ task. The task points are generated randomly by the following manipulator



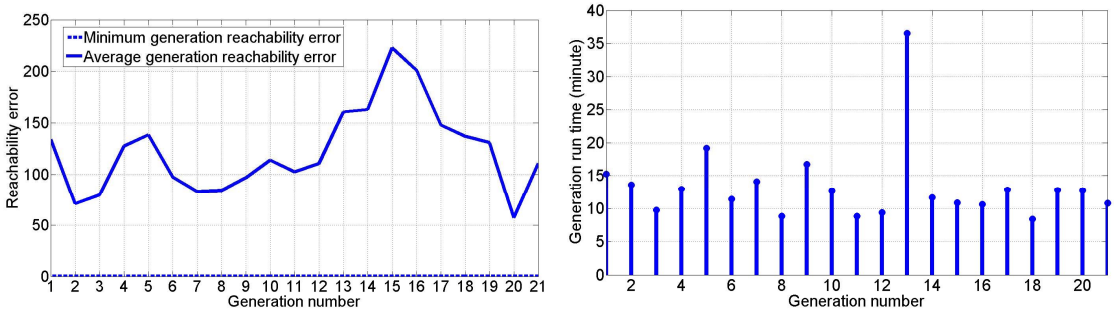
(a) (b) (c)

Figure 6.16: KCs of $\mathbf{m}_5^{ref}$ and the two outputs of the MA-based TBCO algorithm in test case 2: (a) $\mathbf{m}_5^{ref}$; (b) $\mathbf{m}_5^{KC3}$; and, (c) $\mathbf{m}_5^{KC4}$

$$\mathbf{m}_6{}^{ref} = \begin{bmatrix} 0 & 0 & 60 \\ 0 & R & 30 \\ 0 & P & 30 \\ 0 & P & 30 \\ 0 & R & 10 \\ 0 & P & 10 \\ 0 & R & 10 \end{bmatrix}. \tag{6.35}$$

Fig. 6.17(a) portrays the plot of the minimum and average reachability errors of the population with respect to the generation number. It is evident that in generation 1 a manipulator capable of performing the task is found. The MA iterates for 20 more generations in order to search $\mathbb{M}_6$ for manipulators that perform better in executing the task. Fig. 6.17(b) reflects the run time of each generation.

In the final generation of MA, eight KCs, $\mathbf{m}_6^{KCi}$ with $i = 1..8$, capable of performing the task, are found. In Fig. 6.18 the KCs of the aforementioned manipulators are illustrated. Table 6.7 summarizes the operational performance measures of the manipulators. The reachability errors of all the manipulators are within the permissible tolerance range. However, since $\mathbf{m}_6^{KC1}$ can perform the task with lower power requirements, $\mathbf{m}_6^{KC1}$ is selected as the solution to the TBCO problem.



(a) (b)

Figure 6.17: Minimum and average reachability errors of the population and the generation run time for a $\mathbb{M}_6$ search using the proposed MA: (a) minimum and average population reachability errors; and, (b) generation run time

Table 6.7: Results of the MA-based TBCO in $\mathbb{M}_6$

Kinematic Configuration	Position Reachability Error (cm)	Orientation Reachability Error (Rad)	Required Power (KW)
$\mathbf{m}_6^{KC1}$	0.0041	0.0007	1.247
$\mathbf{m}_6^{KC2}$	0.0032	0.0002	1.286
$\mathbf{m}_6^{KC3}$	0.0020	0.0001	10.558
$\mathbf{m}_6^{KC4}$	0.0055	0.0000	3.178
$\mathbf{m}_6^{KC5}$	0.0070	0.0004	4.985
$\mathbf{m}_6^{KC6}$	0.0061	0.0010	3.681
$\mathbf{m}_6^{KC7}$	0.0094	0.0001	7.862
$\mathbf{m}_6^{KC8}$	0.0013	0.0000	2.262

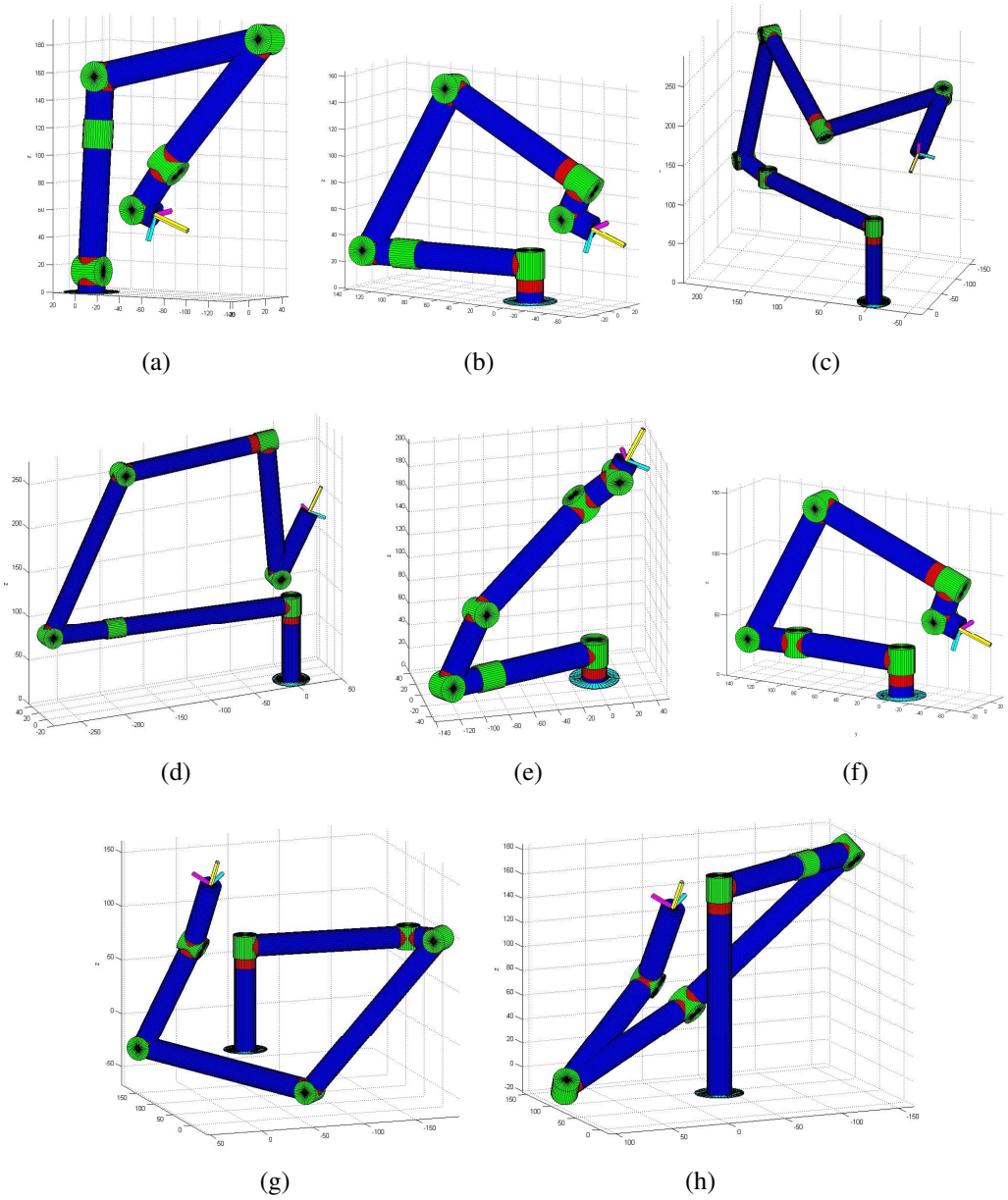


Figure 6.18: KCs of the outputs of the MA-based TBCO algorithm in $\mathbb{M}_6$: (a) $\mathbf{m}_6^{KC1}$; (b) $\mathbf{m}_6^{KC2}$; (c) $\mathbf{m}_6^{KC3}$; (d) $\mathbf{m}_6^{KC4}$; (e) $\mathbf{m}_6^{KC5}$; (f) $\mathbf{m}_6^{KC6}$; (g) $\mathbf{m}_6^{KC7}$; and, (h) $\mathbf{m}_6^{KC8}$

6.7 Summary

In this chapter, an approach for solving the TBCO problem by using an MA is introduced. The proposed algorithm is a hybridization of the GA and local search algorithms. The novel algorithm benefits from a kinematic structure-aware elitism scheme, kinematic structure-aware restart scheme, priority-based selection operator, and adaptive local search frequency. It is demonstrated that in an $\mathbb{M}_3$ TBCO , the proposed MA finds a manipulator capable of satisfying the task faster than the GA. Moreover, the search in $\mathbb{M}_5$ shows that when the number of the task points t in the desired task $\mathbb{T}_t$ increases, the set of manipulators, capable of performing $\mathbb{T}_t$, shrinks. For instance, in one of the test cases, it is shown that the algorithm only found one $\mathbf{m}_5$ kinematic structure, capable of performing the desired $\mathbb{T}_{10}$ task. The proposed MA-based TBCO algorithm is applied to find an optimized $\mathbf{m}_6$ manipulator for performing a $\mathbb{T}_5$ task.

Furthermore, a novel method is proposed for conducting the Δ search, for the link dimensions of a manipulator to enable it to reach a set of task points. The method is based on converting the Δ search to the IK problem of a redundant manipulator, and solving it for all the task points concurrently. The proposed method is verified for three distinct cases, each consisting of a task with 50 task points.

Although for the verification of the algorithm, only the total reachability error as constraint and the required power as the optimization criterion are considered, the algorithm can accommodate any number of constraints and optimization criteria. In a case with more criteria, a weighted sum of constraints and optimization criteria can be adopted as the high and low priority fitness values, respectively.

Chapter 7

Conclusions

7.1 Summary of Contributions

The primary contributions of this thesis are the development of novel efficient Task-based Configuration Optimization (TBCO) algorithms for Modular and Reconfigurable Robots (MRRs). Furthermore, new methods are proposed for incorporating the multiple solutions of the inverse kinematic (IK) problem in the TBCO to increase the optimality of the solutions.

First, an algorithm is proposed to obtain multiple solutions of the IK problem. The algorithm is an enhanced version of an adaptive niching Genetic Algorithm (GA) in conjunction with a clustering and numerical improvement stage, to detect and converge on multiple solutions of the IK problem with the desired precision. Although the method might be slower than some multi-solution IK solvers, it provides more flexibility in terms of incorporating any number of constraints and still yields valid solutions.

Another method is developed for obtaining multiple solutions of the IK problem. The method requires no closed-form solution or prior knowledge of the number of solutions. Also, the algorithm is applicable to all the possible KCs, which can be assembled from the standard modular joint set without any symbolic or analytical modifications. In addition, it is confirmed that the proposed algorithm is faster than the continuation method which is one of the most efficient algorithms for solving the IK problem.

Thirdly, an algorithm is proposed for incorporating multiple solutions of the IK in the trajectory optimization problem. The optimum sequence of performing the tasks, in conjunction with the IK solutions to reach each task, are obtained from the new algorithm. Also, the method can be applied to produce optimized trajectories, when any quantifiable performance measure is considered. The total cost of the optimized trajectory can then be adopted as a performance measure of TBCO.

Furthermore, a Memetic Algorithm (MA) for solving the TBCO problem is proposed. The algorithm benefits from an elitism scheme, a kinematic structure-aware restart process, and adaptive local search frequency. Although it is reported in the literature that MAs are superior to pure GAs for solving a wide range of optimization problems, MAs are not commonly used. The reason can be traced back to the fact that, unlike GAs, no general outline exists for designing competitive and efficient MAs. In specific terms, the architecture of MAs depends on the application.

Lastly, an application-specific local search method for determining the optimal link dimensions of a manipulator for a certain task is developed. The method converts the problem into the IK of a redundant robot, and solves it by a so-called Task Embedded Jacobian matrix of the new manipulator. The performance of the algorithm is verified by several numerical analyses.

7.2 Future Research

The contributions of this thesis can be extended by future research in three areas: (1) Task-Based Configuration Optimization algorithm, (2) Multi-solution IK solvers, and (3) Trajectory optimization algorithms.

7.2.1 Task-Based Configuration Optimization

- The TBCO algorithm can be further improved by the implementation of a multi-meme MA. In the proposed MA, only one meme (a Local search algorithm) is implemented. By developing a multi-meme MA, a set of local search algorithms with distinct characteristics can be incorporated in the MA to enhance the efficiency of the search. The set of local search algorithms can even include a method for performing a kinematic structure, Σ, search. A scheduler stage can then select the most appropriate meme, based on the characteristics of the individuals, the desired task, and the specifications of the local searchers such as the computational speed, solution accuracy, and considered optimization criteria. The use of multi-meme scheme can enable the MA to achieve advantages of a range of distinct local search approaches, instead of only one.

- In the proposed algorithm, the search is conducted in the space of $\mathbb{M}_n$ with a specific, constant n. If no manipulator is capable of performing the task, n should be increased and the search conducted again. The TBCO method can be generalized to accommodate the search in the kinematic space of manipulators with any DOF. In such an algorithm, each individual can represent manipulators from any DOF, and the result of the algorithm may have an optimized DOF. This development, requires new individual representations,

genetic operators, restarting processes, and new optimization criteria to incorporate the DOF as a fitness value.

- The Δ search algorithm (manipulator dimension search) developed in this thesis, converges to the solution which is closest to the search point with which the algorithm is initialized. The Δ search algorithm is based on converting the link dimension search to a redundant manipulator IK problem. In solving the IK of redundant manipulators, infinite solutions exist, and the redundant DOFs can be utilized to optimize a desired criteria, i.e., Among all the poses of the manipulator that produce a zero reachability, the one which optimizes a criteria is determined. Therefore, by using the same principles, a future direction of the current research should focus on extending the Δ search to identify the dimensions of the robot which, in addition to enabling the manipulator to perform the task, optimizes the performance measure.

7.2.2 Multi-Solution Inverse Kinematic Solvers

Adaptive Niching Genetic Algorithm-Based Inverse Kinematic Solver

- The population of the adaptive niching-based IK solver converges on potential solution regions in the joint space. Since the redundant manipulators have infinite IK solutions, the algorithm arranges the individuals as efficiently as it can in the proximity of any solution it determines. This arrangement can be dense around some solutions and leave some solutions with just a few individuals. A future direction for the niching GA based IK solver may be to enhance the method for attaining a set of IK solutions which are optimized according to the desired measure.

Joint Reflection Operator-Based Inverse Kinematic Solver

- A future development of the JRO-based IK solver should focus on the improvement of the speed of the algorithm by an enhanced intelligent method for selecting the joint couples with the potential of producing multiple positioning IK solutions. Such a method can analyze the entire kinematic structure of the manipulator, and produce a smaller list of joint couples to undergo the JRO operator, hence increasing the speed of the algorithm.

- The JRO-based algorithm can be used to find self-motion manifolds of the IK solutions in the joint space for redundant manipulators. By devising a recognition method for the solutions located in a common manifold, the JRO-based algorithm can be used to find a

representative IK solution from each of the manifolds. The set of the manifold representatives may then be used to find the most optimized IK solutions of each manifold. This facilitates the selection of the best overall solution for an application.

7.2.3 Trajectory Optimization Algorithms

- The proposed trajectory optimization algorithm is used for a multi-goal path planning of non-redundant manipulators. In redundant manipulators, the pose of the manipulator can change within the self-motion manifold without the end-effector departing from the task point. This characteristic, translated into the GTSP formulation, can be seen as a GTSP in which the location of each city, and the distance (or cost) of moving between cities can change within a predefined locus. A potential direction for further research is to solve this GTSP problem in redundant manipulators to reach the sequence of performing the tasks, optimizing the self-motion manifold for each task point, and a single IK solution within the self motion manifold to perform a task with the least task execution cost.

Appendices

Appendix A

Positioning Inverse Kinematic Solutions for $\mathbb{M}_2$

In this section, the positioning IK problem for $\mathbb{M}_2$ manipulators is solved. The results of this section are utilized in the proof of Theorems 1 and 2 in Section 4.3.

A.1 Forward Kinematics of $\mathbb{M}_2$ Class Manipulators

The Homogenous-Transformations of the link frames of a 2-DOF manipulator which are expressed with reference to the previous link frame are as follows:

$$\mathbf{T}_{01} = \begin{bmatrix} \cos(\theta_1) & -\cos(\alpha_1)\sin(\theta_1) & \sin(\alpha_1)\sin(\theta_1) & a_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\alpha_1)\cos(\theta_1) & -\sin(\alpha_1)\cos(\theta_1) & a_1\sin(\theta_1) \\ 0 & \sin(\alpha_1) & \cos(\alpha_1) & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

$$\mathbf{T}_{12} = \begin{bmatrix} \cos(\theta_2) & -\cos(\alpha_2)\sin(\theta_2) & \sin(\alpha_2)\sin(\theta_2) & a_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\alpha_2)\cos(\theta_2) & -\sin(\alpha_2)\cos(\theta_2) & a_2\sin(\theta_2) \\ 0 & \sin(\alpha_2) & \cos(\alpha_2) & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

where α_i, a_i, d_i, and θ_i represent the twist angle, link length, link offset, and joint angle according to the Denavit-Hartenberg(DH) convention. $\mathbb{M}_2$ manipulators exclusively consist of revolute

joints. Therefore, the variable link parameters are the joint angles θ_i, and the rest of the DH parameters only vary with the kinematic configuration of the manipulator.

The forward kinematic mapping of $\mathbb{M}_2$ manipulators can now be written as the following:

$$\mathbf{T}_{ee}\left(\theta_1, \theta_2\right) = \mathbf{T}_{01}\left(\theta_1\right).\mathbf{T}_{12}\left(\theta_2\right) \tag{A.3}$$

The homogenous transformation $\mathbf{T}_{ee}$ can be formatted as the following:

$$\mathbf{T}_{ee} = \left(\begin{array}{ccc|c} & \mathbf{R}_{ee} & & \mathbf{P}_{ee} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{A.4}$$

where $\mathbf{R}_{ee}$ is a 3×3 rotation matrix representing the orientation of the end-effector and $\mathbf{P}_{ee}$ is the 3×1 translation matrix representing the position of the end-effector in the Cartesian space.

Using (A.3), and (A.4), $\mathbf{P}_{ee}$ is written as the following:

$$\begin{aligned} \mathbf{P}_{ee} &= \begin{bmatrix} P_{x,ee} \\ P_{y,ee} \\ P_{z,ee} \end{bmatrix} \\ &= \begin{bmatrix} \cos\left(\theta_1\right) a_2 \cos\left(\theta_2\right) - \cos\left(\alpha_1\right) \sin\left(\theta_1\right) a_2 \sin\left(\theta_2\right) + \sin\left(\alpha_1\right) \sin\left(\theta_1\right) d_2 + a_1 \cos\left(\theta_1\right) \\ \sin\left(\theta_1\right) a_2 \cos\left(\theta_2\right) + \cos\left(\alpha_1\right) \cos\left(\theta_1\right) a_2 \sin\left(\theta_2\right) - \sin\left(\alpha_1\right) \cos\left(\theta_1\right) d_2 + a_1 \sin\left(\theta_1\right) \\ \sin\left(\alpha_1\right) a_2 \sin\left(\theta_2\right) + \cos\left(\alpha_1\right) d_2 + d_1 \end{bmatrix} \end{aligned} \tag{A.5}$$

A.2 Inverse Kinematics of $\mathbb{M}_2$ Class Manipulators

The IK in $\mathbb{M}_2$, is the problem of finding θ_1 and θ_2 such that $\mathbf{T}_{ee}$ is equal to a desired homogenous transformation $\mathbf{T}_{des}$. i.e. a solution for θ_i in the following equation is needed:

$$\mathbf{T}_{ee}\left(\theta_1, \theta_2\right) = \mathbf{T}_{des} \tag{A.6}$$

In positioning IK problem, which is the focus of the current appendix, a solution to the positioning part of (A.6) is required:

$$\mathbf{P}_{ee}\left(\theta_1, \theta_2\right) = \mathbf{P}_{des} \tag{A.7}$$

where $\mathbf{P}_{des}$ can be written as the following:

$$\mathbf{P}_{des} = \begin{bmatrix} P_{x,des} \\ P_{y,des} \\ P_{z,des} \end{bmatrix} \tag{A.8}$$

In the rest of this section, (A.7) is solved for θ_1 and θ_2, in all the considered classes of $\mathbb{M}_2$ manipulators.

A.2.1 Class 1: Intersecting Joint Axes

Table A.1 shows the DH parameters of manipulators of this class. By substituting the DH parameters into (A.7), the positioning IK problem for the manipulators of this class is as follows:

$$\begin{bmatrix} \cos\left(\theta_1\right) a_2 \cos\left(\theta_2\right) + \sin\left(\theta_1\right) d_2 \\ \sin\left(\theta_1\right) a_2 \cos\left(\theta_2\right) - \cos\left(\theta_1\right) d_2 \\ a_2 \sin\left(\theta_2\right) \end{bmatrix} = \begin{bmatrix} P_{x,des} \\ P_{y,des} \\ P_{z,des} \end{bmatrix} \tag{A.9}$$

The result of the addition of the square of the first two rows is as follows:

$$d_2^2 + a_2^2 . \cos^2\left(\theta_2\right) = P_{x,des}^2 + P_{y,des}^2 \tag{A.10}$$

Using (A.10) and the third row of (A.9), two values for θ_2 can be calculated:

$$\begin{aligned} \left[\theta_2\right]_1 &= \arctan\left(\frac{P_{z,des}}{a_2}, \frac{\sqrt{P_{x,des}^2 + P_{y,des}^2 - d_2^2}}{a_2}\right) \\ \left[\theta_2\right]_2 &= \pi - \arctan\left(\frac{P_{z,des}}{a_2}, \frac{\sqrt{P_{x,des}^2 + P_{y,des}^2 - d_2^2}}{a_2}\right) \end{aligned} \tag{A.11}$$

Substituting (A.11) into (A.9), θ_1 is determined:

Table A.1: Denavit-Hartenberg parameters of a 2-DOF manipulator with two intersecting joints

Joint	α_i	a_i	d_i	θ_i
1	α_1	0	0	θ_1
2	0	l_2	d_2	θ_2

$$
\begin{aligned}
[\theta_1]_1 &= \arctan\left(\frac{\sqrt{-{d_2}^2+{P_{x,des}}^2+{P_{y,des}}^2}P_{y,des}+P_{x,des}\,d_2}{{P_{x,des}}^2+{P_{y,des}}^2},\right. \\
&\qquad \left.\frac{\sqrt{-{d_2}^2+{P_{x,des}}^2+{P_{y,des}}^2}P_{x,des}-P_{y,des}\,d_2}{{P_{x,des}}^2+{P_{y,des}}^2}\right) \\
[\theta_1]_2 &= \arctan\left(-\frac{\sqrt{-{d_2}^2+{P_{x,des}}^2+{P_{y,des}}^2}P_{y,des}-d_2P_{x,,des}}{{P_{x,des}}^2+{P_{y,des}}^2},\right. \\
&\qquad \left.-\frac{P_{y,,des}\,d_2+\sqrt{-{d_2}^2+{P_{x,des}}^2+{P_{y,des}}^2}P_{x,des}}{{P_{x,des}}^2+{P_{y,des}}^2}\right)
\end{aligned}
\tag{A.12}
$$

A.2.2 Class 2: Parallel Joint Axes

The DH parameters of manipulators of this class are shown in Table A.2. Substituting the DH parameters into (A.7), the following positioning IK equations are obtained:

$$
\begin{bmatrix}
\cos(\theta_1)\,a_2\cos(\theta_2)-\sin(\theta_1)\,a_2\sin(\theta_2)+a_1\cos(\theta_1) \\
\sin(\theta_1)\,a_2\cos(\theta_2)+\cos(\theta_1)\,a_2\sin(\theta_2)+a_1\sin(\theta_1) \\
0
\end{bmatrix}
=
\begin{bmatrix}
P_{x,des} \\ P_{y,des} \\ P_{z,des}
\end{bmatrix}
\tag{A.13}
$$

The result of the addition of the square of the first two rows is as follows:

$$
{a_2}^2+2\,a_2\cos(\theta_2)\,a_1+{a_1}^2=P_{x,des}^2+P_{y,des}^2 \tag{A.14}
$$

from which θ_2 can be derived as:

$$\cos(\theta_2) = \frac{1}{2}\frac{-{a_2}^2 - {a_1}^2 + {P_{x,des}}^2 + {P_{y,des}}^2}{a_2 a_1} \tag{A.15}$$

Based on (A.15) two values for θ_2 can be calculated:

$$\begin{aligned} [\theta_2]_1 &= \arccos\left(\frac{1}{2}\frac{-{a_2}^2 - {a_1}^2 + {P_{x,des}}^2 + {P_{y,des}}^2}{a_2 a_1}\right) \\ [\theta_2]_2 &= -\arccos\left(\frac{1}{2}\frac{-{a_2}^2 - {a_1}^2 + {P_{x,des}}^2 + {P_{y,des}}^2}{a_2 a_1}\right) \end{aligned} \tag{A.16}$$

Substituting (A.16) into (A.13), two solutions for θ_1 are obtained as follows:

$$\begin{aligned} [\theta_1]_1 &= -\arctan\left(\frac{P_{x,des}\sqrt{P_m} + P_{y,des}\left(-{a_2}^2 + {a_1}^2 + {P_{xy}}^2\right)}{{P_{xy}}^2 a_1}, \right. \\ &\quad \left. -\frac{P_{x,des}\left({P_{xy}}^2 + {a_1}^2 - {a_2}^2\right).\sqrt{P_m} + P_{y,des}.P_m}{{a_1}^2.\sqrt{P_m}.{P_{xy}}^2}\right) \\ [\theta_1]_2 &= \arctan\left(\frac{P_{x,des}\sqrt{P_m} + P_{y,des}\left(-{a_2}^2 + {a_1}^2 + {P_{xy}}^2\right)}{{P_{xy}}^2 a_1}, \right. \\ &\quad \left. -\frac{P_{x,des}\left({P_{xy}}^2 + {a_1}^2 - {a_2}^2\right).\sqrt{P_m} + P_{y,des}.P_m}{{a_1}^2.\sqrt{P_m}.{P_{xy}}^2}\right) \end{aligned} \tag{A.17}$$

where P_m and P_{xy} are defined as follows:

$${P_{xy}}^2 = {P_{x,des}}^2 + {P_{y,des}}^2 \tag{A.18}$$

$$P_m = -\left({P_{xy}}^2 - (a_1 + a_2)^2\right)\left({P_{xy}}^2 - (a_1 - a_2)^2\right) \tag{A.19}$$

Table A.2: Denavit-Hartenberg parameters of a 2-DOF manipulator with two parallel joints

Joint	α_i	a_i	d_i	θ_i
1	0	l_1	0	θ_1
2	0	l_2	0	θ_2

Table A.3: Denavit-Hartenberg parameters of a 2-DOF manipulator with joint axes which are not intersecting nor parallel

Joint	α_i	a_i	d_i	θ_i
1	$\frac{\pi}{2}$	0	d_1	θ_1
2	0	l_2	0	θ_2

A.2.3 Class 3: Non-Intersecting and Non-Parallel Joint Axes

Table A.3 shows the DH parameters of the members of this class. Substituting the DH parameters into (A.7) results in the following positioning IK equation:

$$\begin{bmatrix} \cos(\theta_1)\, a_2 \cos(\theta_2) + a_1 \cos(\theta_1) \\ \sin(\theta_1)\, a_2 \cos(\theta_2) + a_1 \sin(\theta_1) \\ a_2 \sin(\theta_2) + d_1 \end{bmatrix} = \begin{bmatrix} P_{x,des} \\ P_{y,des} \\ P_{z,des} \end{bmatrix} \tag{A.20}$$

Dividing the second equation by the first results in:

$$\tan(\theta_1) = \frac{P_{y,des}}{P_{x,des}} \tag{A.21}$$

From (A.21), the values for θ_1 can be calculated as:

$$[\theta_1]_1 = \arctan\left(\frac{P_{y,des}}{P_{x,des}}\right) \tag{A.22}$$

$$[\theta_1]_2 = \pi + \arctan\left(\frac{P_{y,des}}{P_{x,des}}\right) \tag{A.23}$$

$\sin(\theta_2)$ and $\cos(\theta_2)$ can be calculated from the third equation and the addition of the square of the first and second equations of (A.20), respectively. Hence, the expression for θ_2 can be obtained as follows:

$$[\theta_1]_1 = \arctan\left(\frac{P_{z,des} - d_1}{a_2}, \frac{-a_1 + \sqrt{-{d_2}^2 + {P_{x,des}}^2 + {P_{y,des}}^2}}{a_2}\right) \quad \text{(A.24)}$$

$$[\theta_1]_2 = \arctan\left(\frac{P_{z,des} - d_1}{a_2}, \frac{-a_1 - \sqrt{-{d_2}^2 + {P_{x,des}}^2 + {P_{y,des}}^2}}{a_2}\right) \quad \text{(A.25)}$$

Appendix B

Waterloo Modular and Reconfigurable Robot (WMRR)

In this section, the general architecture of WMRR is explained. Moreover, the specifications of the joint modules control nodes are presented.

B.0.4 Architecture

The architecture of the WMRR is shown in Fig. B.1. Each joint module includes an actuator. The actuator's electrical drives are located on the base of the robot in order to decrease the total mass of the joints. Cables rated at 380V transfer the power to the actuators from the drives. In the future phases, using smaller and lighter drives will enable the robot to have the drive embedded inside joints.

Joint modules have Micro-Controller Unit boards (called MCU Node) embedded inside. Each node has three primary functions: controlling the joint actuator, communicating with other nodes and the host computer, and synchronization of operation. Each Node is connected to the drives through two main signals. It reads the Encoder Signal (A standard quadrature signal) from the drive and sends the desired Torque, velocity, or joint angle signal (10V Analogue signal) to the drive. The communication and synchronization between the nodes are achieved through a *Controller Area Network (CAN)* bus. A separate bus provides the 24/5V power to the controller boards. A host computer for uploading the trajectories to the nodes, debugging and performance analysis is connected to all the MCU nodes through the CAN bus.

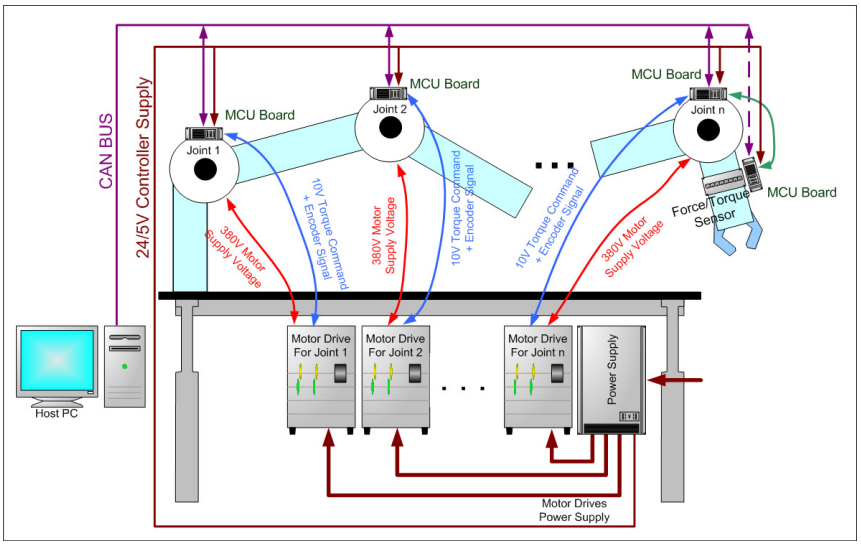


Figure B.1: Architecture of the Waterloo Modular and Reconfigurable Robot

B.0.5 MCU Node Design

Fig B.2 shows the electrical architecture of the MCU nodes. Modularity and the ability to manage multiple duties were the main objectives in the MCU node design. More specifically, the board is designed such that it can be used as a joint controller, a CAN logger, as well as a processor for other peripherals such as torque/force sensors.

The MCU board consists of the following components:

1. *Micro-Controller:* A PIC18F6680 Micro-Controller is used as the core processing unit of the nodes. Characteristics of the PIC Micro-Controller is shown in Table B.1.

2. *Power Regulation Unit:* This unit provides the necessary electrical energy for the board. To provide more flexibility in industrial environments this unit can be supplied from a 5V or 12V unregulated voltage source.

3. *ADC - Analogue to Digital Converter:* This converter is the main input port of the board. Most of the sensors, including force-torque and hall-effect sensors, produce analogue signals. Hence a fast noise free ADC is provided in the design to convert the inputs to digital signals usable by the Micro-controller.

4. *DAC - Digital to Analogue Converter:* This unit is the primary output port of the board. Digital signals are converted to analogue signals before being sent to the peripheral systems. Since the communication with the motor drive of each joint is conducted through a +10/-10V signal, the main application of this unit is to convert the desired torque, velocity, or joint angle to a +10/-10V analogue signal usable by the motor drive.

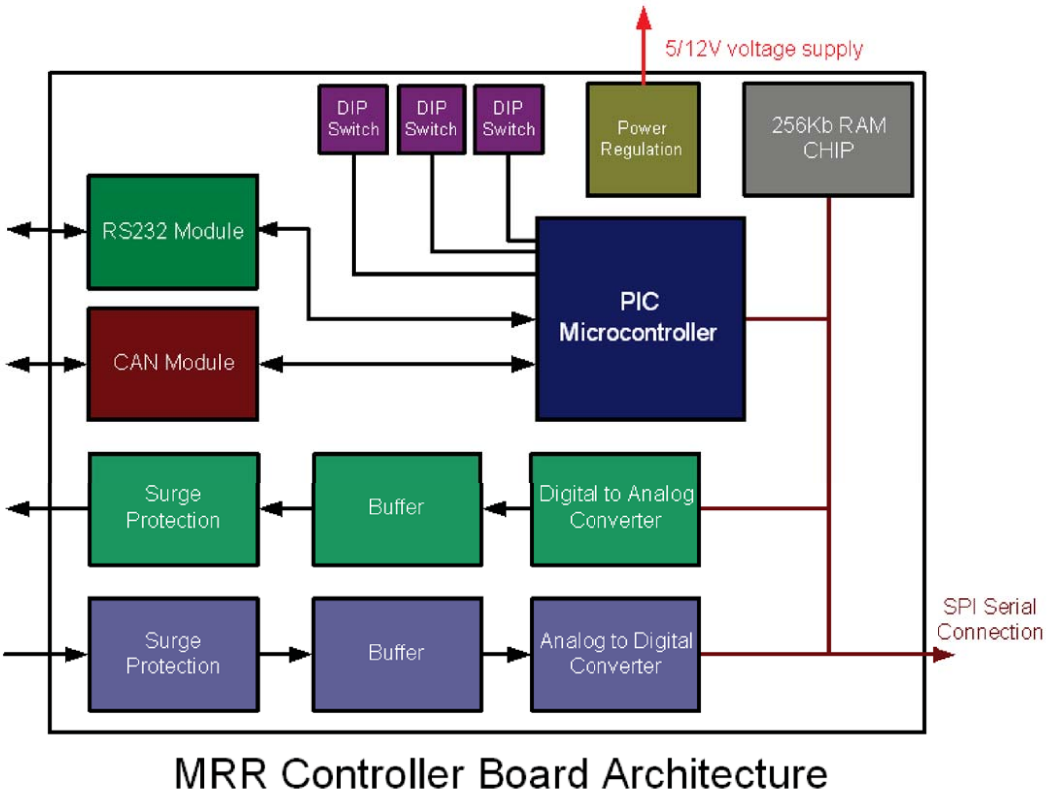


Figure B.2: Block diagram of the MCU board

Table B.1: Specifications of the Micro-Controller used in the MCU node design

Specification	Description
Micro-Controller	PIC18F6680
Programming Memory	64KB (32768 word instructions)
SRAM Memory	3328B
EEPROM Memory	1KB
Performance	Up to 10 MIPS, 40MHz

5. *256Kb RAM chip:* All the desired task points in the operational trajectory of the joints are stored in the corresponding MCU board. Therefore, a memory expansion to complement the internal memory of the micro-controller is necessary. The RAM chip provides an additional storage of 256Kb.

6. *DIP Switches:* Three rotary DIP switches are provided in each board. Each of these rotary switches can be set at 16 different positions/numbers. Therefore, 4096 unique ID numbers can be assigned to the boards. The ID numbers will be used on the CAN bus for communication between the nodes.

7. *RS232 Module:* This unit is a serial communication port. Programming and debugging the PIC Micro-Controllers are accomplished through this port. This module can also be used to connect each of the boards to an external computer for data acquisition.

8. *Controller Area Network(CAN) Module:* This module provides the means of forming networks with the other nodes through CAN bus. The network is used in the synchronous operation of all the joints. Another possible application for this bus is automatic configuration detection (providing Plug and Play capability for Modules). CAN bus is also used to provide communications between a host computer and the MCU nodes.

9. *Buffer and Surge Protection:* The main task of this unit is protection of the ADC and DAC units from unexpected and harmful electrical interferences.

10. *SPI Serial Connection:* The SPI Serial Connection provides the means of connecting two controller boards to exchange data directly (not through the CAN bus). Using the SPI connection, it becomes possible to perform more processing intensive tasks with parallel processing. For instance, the data processing of the force-torque sensors could be carried out on one MCU node and the results could then be transferred to another node for incorporation into the control algorithm.

In Fig.B.3 a photo of the MCU node populated with the electrical components is shown.

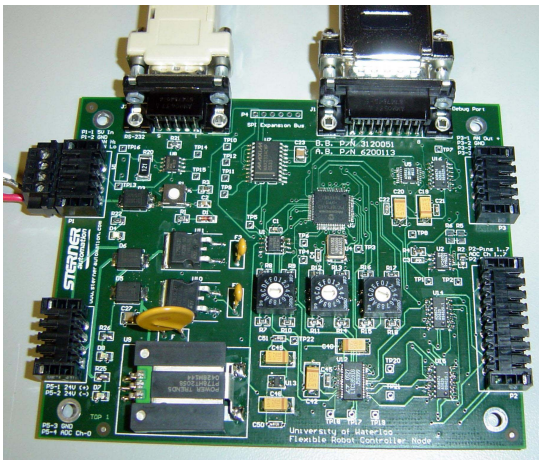


Figure B.3: The populated MCU node printed circuit board

References

[1] Board on Manufacturing and National Research Council Engineering Design. *Visionary Manufacturing Challenges for 2020*. NATIONAL ACADEMY PRESS, Washington, D.C., 1998. 1

[2] Z. M. Bi and W. J. Zhang. Modularity technology in manufacturing: Taxonomy and issues. *International Journal of Advanced Manufacturing Technology*, 18:381—390, 2001. 1, 3

[3] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang. Reconfigurable manufacturing systems: the state of the art. *International Journal of Production Research*, 46:4:967—992, 2007. 1, 3

[4] Z. M. Bi, S. Y. T. Lang, M. Verner, and P. Orban. Development of reconfigurable machines. *International Journal of Advanced Manufacturing Technology*, 39:1227—1251, 2008. 1

[5] S. Kirchhoff. Intelligent configuration selection and robust control for reconfigurable robots. Master's thesis, University of Waterloo,and Technical Universitat Hamburg-Harburg, 2005. 3

[6] S. Tabandeh, C. M. Clark, and W. W. Melek. Task-based configuration optimization of modular and reconfigurable robots using a multi-solution inverse kinematics solver. In *Proceedings of the 2nd Internation Conference on Changebale, Agile, Reconfigurable and Virtual Production (CARV 2007)*, Toronto, Canada, July 2007. 4

[7] T. Fukuda and S. Nakagawa. Dynamically reconfigurable robotics system. *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1581–1586, 1988. 10, 30

[8] D.E. Schmitz, P.K. Khosla, and T. Kanade. The *CMU* reconfigurable modular manipulator system. In Jarvis, editor, *Robots, coming of age*, volume 1, pages 473–488, 1988. 10

[9] C.J.J. Paredis, H.B. Brown, and P.K. Khosla. A rapidly deployable manipulator system. *Robotics and Autonomous Systems*, 21:289–304, 1997. 10

[10] C.J.J. Paredis. *An Agent Based Approach to the Design of Rapidly Deployable Fault Tolerant Manipulators*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890, August 1996. 10, 30

[11] I.-M. Chen. Realization of a rapidly reconfigurable robotic workcell. *Journal of Japan Society for Precision Engineering*, 66(7):1024–1030, 2000. 11

[12] I.-M. Chen. Rapid response manufacturing through reconfigurable robotic workcells. *Journal of Robotics and Computer Integrated Manufacturing*, 17(3):199–213, 2001. 11, 36

[13] I.-M. Chen. A rapidly reconfigurable robotics workcell and its applictions for tissue engineering. *Innovation in Manufacturing Science and Technology Program, Singapore-MIT Alliance Annual Symposium, Traders Hotel, Singapore*, 2003. 11

[14] R. Hui, N. Kircanski, A. Goldenberg, C. Zhou, P. Kuzan, J. Wiercienski, D. Gershon, and P. Sinha. Design of the *IRIS* facility - a modular, reconfigurable and expandable robot test bed. *Proc. IEEE Int. Conf. Robotics and Automation*, 3:155–160, 1993. 11

[15] T. Matsumaru. Design and control of the modular robot system: *TOMMS*. *Proc. IEEE Int. Conf. Robotics and Automation*, 2:2125–2131, 21-27 May 1995. 11

[16] Z. Li. Development and control of a modular and reconfigurable robot with harmonic drive transmission system. Master's thesis, University of Waterloo, 2007. 12

[17] A. Ellery. *An introduction to space robotics*. Springer, 2000. 19

[18] B. Siciliano and O. Khatib. *Handbook of Robotics*. Springer, 2008. 25, 34

[19] M. Uchimaya, K. Shimizu, and K. Hakomori. Performance evaluation of manipulators using the jaobian and its application to trajectory planning. *The Second International Symposium of Robotics Research*, 1:447–454, 1985. 26

[20] S.L. Chiu. Task compatibility of manipulator postures. *International Journal of Robotics Reseach*, 7(5):13–21, October 1988. 26

[21] C.A. Klein and T.A. Miklos. Spatial robotic isotrpy. *International Journal of Robotics Reseach*, 10(4):426–437, 1991. 26

[22] T. Yoshikawa. Manipulability of robotic mechanisms. *International Journal of Robotics Reseach*, 4(2):3–9, 1985. 26, 36, 127

[23] J.-O. Kim and P.K. Khosla. Dexterity measures for design and control of manipulators. *in Proceedings of IROS'91: International Workshop on Intelligent Robots and Systems, Osaka, Japan*, pages 758–763, Nov. 1991. 26

[24] J.-O. Kim. *Task based kinematics of robot manipulators*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890, August 1992. 26, 27, 30

[25] G. yang and I.-M. Chen. Reduced dof modular robot configurations. *5th Int. Conf. Control, Automation, Robotics and Vision, Singapore*, pages 1513–1517, 1998. 27, 28

[26] C.J.J. Paredis and P. Khosla. Kinematics design of serial link manipulators from task specifications. *Int. J. of robotics research*, 12(3):274–287, 1993. 27, 30

[27] I.-M. Chen and J.W. Burdick. Determining task optimal modular robot assembly configurations. *IEEE international conference on Robotics and Automation*, pages 132–137, 1995. 28

[28] I.-M. Chen. On optimal configuration of modular reconfigurable robots. *4th Int. Conf. on Control, Automation, Robotics and Vision, Singapore*, pages 1855–1859, 1996. 28

[29] I.-M. Chen. *Theory and Applications of Modular Reconfigurable Robotic Systems*. PhD thesis, Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA, USA, 1994. 28

[30] I.-M. Chen and G. Yang. Configuration independent kinematics for modular robots. *Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation*, pages 1440–1445, 1996. 28

[31] I.-M. Chen and G. Yang. Inverse kinematics for modular reconfigurable robots. *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, pages 1647–1652, 1998. 28, 36

[32] I.-M. Chen, G. Yang, and I.-G. Kang. Numerical inverse kinematics for modular reconfigurable robots. *Journal of Robotics systems*, 16(4):213–225, 1999. 28, 36

[33] I.-M. Chen and G. Yan. Closed-form inverse kinematics solver for reconfigurable robots. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 2395–2400, 2001. 28

[34] I.-M. Chen and G. Yang. An evolutionary algorithm for the reduction of dofs in modular reconfigurable robots. *In Proceeding of the ASME Dynamic System and Control Division, DSC-64*, pages 759–766, 1998. 28

[35] S. Ramachandran and I.-M. Chen. Distributed agent based design of modular reconfigurable robots. *Proc. of the 5th International Conference on Computer Integrated Manufacturing, Singapore*, pages 447–458, 2000. 28

[36] C. Leger and J. Bares. Automated synthesis and optimization of robot configurations. *Proceedings of the 1998 ASME Design Engineering Technical Conferences, Atlanta, GA*, 1998. 30

[37] C. Leger and J. Bares. Automated task-based synthesis and optimization of field robots. *Proceedings of the 1999 International Conference on Field and Service Robotics (FSR99), Pittsburgh, PA*, 1999. 30

[38] C. Leger. *Automated Synthesis and Optimization of Robot Configuations: An Evolutionary Approach*. PhD thesis, The Robotics Institute, Carnegie Mellon University, Pittsburg, Pennsylvania 15213, December 1999. 30

[39] C.J.J. Paredis and P.K. Khosla. An approach for mapping kinematic task specifications into a manipulator design. *Proc. of 5th International Conference on Advanced Robotics (ICAR '91)*, 1:556–561, 1991. 30

[40] J.-O. Kim and P. Khosla. A formulation for task based design of robot manipulators. *Proc. Of the 1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2310–2317, 1993. 30

[41] J. Kim and P. Khosla. Design of space shuttle tile servicing robot: An application of task based kinematic design. *IEEE International Conference on Robotics and Automation (ICRA '93)*, 3:867–874, May 1993. 30

[42] C.J.J. Paredis and P.K. Khosla. Agent-based design of fault tolerant manipulators for satellite docking. *Proc. Of the 1997 IEEE Int. Conf. on Robotics and Automation, Albuquerque, New Mexico*, April 1997. 30

[43] J. Han, W.K. Chung, Y. Youm, and S.H. Kim. Task based design of modular robot manipulator using efficient genetic algorithm. *Proceedings of the 1997 IEEE international conference on Robotics and Automation, Albuquerque, New Mexico*, pages 507–512, April 1997. 30

[44] Z.M. Bi and S.Y.T. Lang. General modular robot architecture and configuration design. *Proceedings of the IEEE International Conference on Mechatronics and Automation, Niagara Falls, Canada*, pages 268–273, July 2005. 30

[45] O. Chocron and P. Bidaud. Genetic design of 3d modular manipulators. *Proceedings of the 1997 IEEE Int. Conf. on Robot. And Auto., Albuquerque, New Mexico*, April 1997. 30

[46] O. Chocron. Evolutionary design of modular robotic arms. *Robotica*, 26:323—330, 2007. 30

[47] J.-Y. Park, P.-H. Chang, and J.-Y. Yang. Global optimal approach for robot kinematics design using the grid method. *International Journal of Control, Automation, and Systems*, 4(5):575—591, 2006. 30

[48] J.-Y. Park, P.-H. Chang, and J.-Y. Yang. Task-oriented design of robot kinematics using grid method. *Advanced Robotics*, 17(9):897—907, 2007. 30

[49] P.-H. Chang and H.-S. Park. Development of a robotic arm for handicapped people: A Task-Oriented design approach. *Autonomous Robots*, 15:81–92, 2003. 30

[50] P. S. Shiakolas and S. F. Haider. Optimum module selection and design based on kinematic and dynamic task requirements using DADS and genetic algorithms. In *Sensor Fusion and Decentralized Control in Robotic Systems II*, Boston, MA, September 1999. 30

[51] P. S. Shiakolas, D. Koladiya, and J. Kebrle. Optimum robot design based on task specifications using evolutionary techniques and kinematics, dynamic, and structuranl constraints. *Inverse Problems in Science and Engineering*, 10:4:359—375, 2002. 31

[52] M .Pettersson, J. Andersson, P. Krus, D. Wappling, and X. Feng. Industrial robot design optimization in the conceptual design phase. *Mechatronics and Robotics 2004, Aachen, Germany*, pages 13–15, September 2004. 31

[53] B. Paden and S. Sastr. Optimal kinematic design of 6r manipulators. *International Journal of Robotics Research*, 7(2):43–61, 1988. 31

[54] R. Vijaykumar, K.J. Waldron, and M.J. Tsai. Geometric optimization of serial chain manipulator structures for working volume and dexterity. *International Journal of Robotics Research*, 5(2):91–103, 1986. 31

[55] M. Kircanski. Kinematic isotropy and optimal kinematic design of planar manipulators and a 3-DOF spatial manipulator. *The International Journal of Robotics Research*, 15:61—77, 1996. 31

[56] B. K. Rout and R. K. Mittal. Optimal design of manipulator parameter using evolutionary optimization techniques. *Robotica*, 2009. 31

[57] R.P. Paul. *Robot Manipulator: Mathematics, Programming and Control*. MIT Press , Cambridge, Mass., 1981. 33, 34, 35

[58] L.Tsai and A. Morgan. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *Journal of Mechanisms, Transmissions and Automation in Design*, 107:189–200, 1985. 35

[59] S. Kucuck and Z. Bingul. The inverse kinematics solutions of fundamental robot manipulators with offset wrist. In *IEEE International Conference on Mechatronics*, pages 197–202, July 2005. 35

[60] Xie, S. Yan, and W. Qiang. A method for solving the inverse kinematics problem of 6-dof space manipulator. In *1st International Symposium on Systems and Control in Aerospace and Astronautics*, pages 379–382, January 2006. 35

[61] D. Pieper. *The kinematics of manipulators under computer control*. Unpublished, Stanford University, 1968. 35, 43, 100

[62] D. Pieper and B. Roth. The kinematics of manipulators under computer control. In *Proceedings of the second International Congress on Theory of Machines and Mechcanics*, volume 2, pages 159–169, 1969. 35, 43, 100

[63] G.P.A. Alards and Th.E. Schouten. The intersection method: a new approach to the inverse kinematics problem. In *excerpts from the eighth international conference on CAD/CAM, robotics and factories of the future*, pages 17–19, Metz, France, August 1992. 35

[64] I.S. Fischer. A geometric method for determining joint rotations in the inverse kinematics of robotic manipulators. *Journal of Robotics Systems*, 17(2):107–117, 2000. 35

[65] M. Raghavan and B. Roth. Inverse kinematics of the general 6R manipulator and related linkages. *ASME Journal of Mechanical Design*, 115:502–508, 1993. 35

[66] D. Manocha and J. F. Canny. Efficient inverse kinematics for general 6R manipulators. *IEEE Trans. Rob. Auto.*, 10(5):648–657, 1994. 35

[67] H.-Y. Lee and C.-G. Liang. Displacement analysis of the general spatial 7-link 7R mechanism. *Mechanism and Machine Theory*, 23(3):219–226, 1988. 35

[68] J. M. Porta. Multi-loop position analysis via iterative linear programming. In *in Proc. of Robotics, Science, and Systems*, pages 169–176. MIT Press, 2006. 35, 37

[69] L.W. Tsai and A.P. Morgan. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *ASME J. Mechan. Transmission Autom. Design*, 107:189–195, 1985. 35

[70] B. Roth and F. Freudenstein. Synthesis of path-generating mechanisms by numerical methods. *ASME Journal of Engineering for Industry*, 85:298–307, 1963. 36

[71] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME Journal of Mechanical Design*, 112:59–68, 1990. 36

[72] A. J. Sommese, J. Verchelde, and C. W. Wampler. Advances in polynomial continuation for solving problems in kinematics. *Journal of mechanical design*, 126(2):262–268, 2004. 36

[73] A.A. Goldenberg and D.L. Lawrence. A generalized solution to the inverse kinematics of robotics manipulators. *ASME Journal of Dynamic Systems, Measurements, Control*, 107:103–106, March 1985. 36, 97

[74] A.A. Goldenberg, B. Benhabib, and G. Fenton. A complete generalized solution to the inverse kinematics of robots. *IEEE Journal of Robotics and Automation*, RA-1(1), March 1985. 36, 97

[75] J. Angeles. On the numerical solution of the inverse kinematic problem. *The International Journal of Robotics Research*, 4(2):21–37, 1985. 36

[76] L.-C.T. Wang and C.C. Chen. A combined optimization method for solving the inverse kinematicsproblems of mechanical manipulators. In *IEEE Trans. on Robotics and Automation*, volume 7, pages 489–499, 1991. 36

[77] Y. Choi. Singularity-robust inverse kinematics using lagrange multiplier for redundant manipulators. *Journal of Dynamic Systems, Measurements, and Control*, 130(5):51009–51016, 2008. 36

[78] D.E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10:47–53, 1969. 36, 127

[79] A. A. Maciejewski and C. A. Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *International Journal of Robotic Research*, 4:109–117, 1985. 36

[80] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Trans. ASME Journal of Dynamic Systtems, Measurements, and Control*, 108:163–171, 1986. 36, 128

[81] C.W. Wampler II. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions of Systems, Man, and Cybernetics*, 16:93–101, 1986. 36, 128

[82] W. A. Wolovich and H. Elliot. A computational technique for inverse kinematics. In *Proceedings of the 23rd IEEE Conference on Decision and Control*, pages 1359–1363, 1984. 36

[83] J. Baillieul. Kinematic programming alternatives for redundant manipulators. In *IEEE International Conference of Robotics and Automation*, pages 722–728, St. Louis, 1985. 36

[84] P.H. Chang. A closed-form solution for inverse kinematics of robot manipulators with redundancy. *IEEE Journal of Robotics and Automation*, 3:393–403, 1987. 36

[85] L. Sciavicco and B. Siciliano. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE Journal of Robotics and Automation*, 4:403–410, 1988. 37

[86] O. Egeland. Task-space tracking with redundant manipulators. *IEEE Journal of Robotics and Automation*, 3:471–475, 1987. 37

[87] H. Seraji. Configuration control of redundant manipulators: theory and implementation. *IEEE Journal of Robotics and Automation*, (5):472–490, 1989. 37

[88] R.S. Rao, A. Asaithambi, and S.K. Agrawal. Inverse kinematic solution of robot manipulators using interval analysis. *ASME Journal of Mechanical Design*, 120(1):147–150, 1998. 37

[89] A. Castellet and F. Thomas. Towards an efficient interval method for solving inverse kinematics problems. In *In proceedings of the 1997 IEEE international conference on Robotics and Automation*, pages 3615–3620, Albuquerque, New Mexico, 1997. 37

[90] A. Castellet. *Solving Inverse Kinematics Problems Using an Interval Method*. PhD thesis, Universitat Politècnica de Catalunya, 1998. 37

[91] G.Z. Grudic and P.D. Lawrence. Iterative inverse kinematics with manipulator configuration control. *IEEE Trans. on Robotics and Automation*, 9(4):476–483, August 1993. 37

[92] V. D. Tourassis and M. H. Ang. A modular architecture for inverse robot kinematics. *IEEE Trans. on Robotics and Automation*, 5(5):555–568, 1989. 37

[93] T. Balkan, M.K. Ozgoren, M.A.S. Arokan, and H.M. Baykurt. A method of inverse kinematics solution including singular and multiple congurations for a class of robotic manipulators. *Mechanism and Machine Theory*, 35:1221–1237, 2000. 37

[94] J.K. Parker, A.R. Khoogar, and D.E. Goldberg. Inverse kinematics of redundant robots using genetic algorithm. In *IEEE Int. Conf. on Robotics and Cybernetics*, pages 271–276, 1989. 37

[95] K.A. Buckley, S.H. Hopkins, and B.C.H. Turton. Solution of inverse kinematics problems of a highly kinematically redundant manipulator using genetic algorithms. In *2nd Int. Conf. on Genetic Algorithms in Engineering - Innovations and Applications (Conf. Publ. No.449)*, pages 264–269, Sep. 1997. 37

[96] L. Sheng, L. Wan-long, D. Yan-chun, and F. Liang. Forward kinematics of the stewart platform using hybrid immune genetic algorithm. In *IEEE Conf. on Mechatronics and Automation*, pages 2330–2335, June 2006. 37

[97] Y. Zhang, Z. Sun, and T. Yang. Optimal motion generation of a flexible macro-micro manipulator system using genetic algorithm and neural network. In *IEEE Conf. on Robotics, Automation and Mechatronics*, pages 1–6, Dec 2006. 37

[98] F. Chapelle, O. Chocron, and Ph. Bidaud. Genetic programming for inverse kinematics problem approximation. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3364 – 3369, 2001. 37

[99] O. Chocron and Ph. Bidaud. Evolutionnary algorithms in kinematic design of robotic systems. In *International Conference on Intelligent Robots and Systems*, pages 1111–1117, September 1997. 37

[100] P. Karla, P.B. Mahapatra, and D.K. Aggarwal. On the solution of multimodal robot inverse kinematic function using real-coded genetic algorithms. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 2, pages 1840–1845, 2003. 37

[101] P. Karla, P.B. Mahapatra, and D.K. Aggarwal. On the comparison of niching strategies for finding the solution of multimodal robot inverse kinematics. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 6, pages 5356–5361, 2004. 37

[102] J.B. Kuiper. *Quaternions and rotation Sequences: a Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 2002. 41

[103] S.L. Altmann. *Rotations, Quaternions, and Double Groups*. Dover Publications, 1986. 41

[104] J.J. Craig. *Introduction to Robotics: mechanic and control*. Addison-Wesley, 1989. 42

[105] P.I. Corke. A robotics toolbox for matlab. *IEEE Robotics and Automation Magazine*, 3(1):24–32, 1996. 44

[106] D.E. Goldberg. *Genetic Algorithms in search, optimization, and machine laearning*. Addison-Wesley, 1989. 48, 49

[107] B. Sareni and L. Krähenbühn. Fitness sharing and niching methods revisited. *IEEE Trans. on Evolutionary Computation*, 2(3), SEP 1998. 54

[108] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. *Proc. 2nd. Int. Conf. Genetic Algorithms*, pages 41–49, 1987. 54

[109] D.E. Goldberg and L. Wang. Adaptive niching via coevolutionary sharing. *Illigal Report No.97007, Urbana, IL: University of Illinois at Urbana-Champaign*, 1997. 54

[110] M. Neef, D. Thierens, and H. Arciszewski. A case study of a multiobjective recombinative genetic algortihm with coevolutionary sharing. *Proc. of 1999 Congress on Evolutionary Computation*, 1, 1999. 55

[111] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, Feb. 1999. 57, 135

[112] G. Rudolph. Evolutionary search under partially ordered fitness sets. In *In Proceedings of the International Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems (ISI 2001*, pages 818–822. ICSC Academic Press, 1999. 57, 135

[113] K. Deb and R.B. Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995. 58, 60

[114] K. Deb and M. Goyal. A combined genetic adaptive search *(GeneAS)* for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1995. 60, 139, 140

[115] S. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3), SEP. 1994. 63

[116] J.-Y. Park, P.-H. Chang, and J.-Y. Yang. Real-time Perception-Guided motion planning for a personal robot. *Advanced Robotics*, 17(9):879—907, 2003. 76

[117] J. E. Dennis and V. J. Torczon. Derivative-free pattern search methods for multidisciplinary design problems. In *Proceedings of the 5th AIAA/ USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City, FL, 1994. 85

[118] V. J. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Rice University, 1989. 85

[119] L.W. Tsai. *Robot analysis: the mechanics of serial and parallel manipulators*. Wiley-IEEE, 1999. 97

[120] C.G. Broyden. The convergence of a class of double-rank minimization algorithms. *Journal Inst. Math. Applic.*, 6:76–90, 1970. 97

[121] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970. 97

[122] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computing*, 24:23–26, 1970. 97

[123] D.F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computing*, 24:647–656, 1970. 97

[124] A. J. Sommese and C. W. Wampler. Homlab 10. http://www.nd.edu/ cwample1/HomLab/main.html, 2005. 99

[125] A. J. Sommese and C. W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific, 2005. 99

[126] F. Pfeiffer and R. Johanni. A concept for manipulator trajectory planning. *IEEE J. of Robotics and Automation*, RA-3(2):115–123, 1986. 107

[127] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotic Research*, 4:3–17, 1985. 107

[128] K. G. Shin and N. D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans. Automat. Contr.*, AC-30:531–541, 1985. 107

[129] S. F. P. Saramago and V. Steffen. Optimal trajectory planning of robot manipulators in the presence of mocing obstacles. *Mechanism and Machine Theory*, 35(8):1079–1094, 2000. 107

[130] S. F. P. Saramago and V. Steffen. Trajectory modeling of robot manipulators in the presence of obstacles. *J. of Optimization Theory and Applications*, 110(1):17–34, 2001. 107

[131] S. Miossec, K. Yokoi, and A. kheddar. A method for trajectory optimization of robots having contacts or motion constraints. In *IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–6, 2006. 107

[132] A.R. Hirakawa and A. Kawamura. Trajectory generation for redundant manipulators under optimizationof consumed electrical energy. In *Conference Record of the IEEE Industry Applications Conference,*, volume 3, pages 1626–1632, 1996. 108

[133] R. C. Carignan. Trajectory optimization for kinematically redundant arms. *J. of Robotics Systems*, 8(2):221–248, 1991. 108

[134] L. Tian and C. Collins. An effective robot trajectory planning method using a genetic algorithm. *Elsevier J. of mechatronics*, 14(5):455–470, 2003. 108

[135] M. Saha, G. Sanchez-Ante, J.C. Latombe, and T. Roughgarden. Planning multi-goal tours for robot arms. *Int. J. Robotics Research*, 25(3):207–223, March 2006. 108

[136] C. Wurll and D. Henrich. Point-to-point and multi-goal path planning for industrial robots. *J. of Robotic Systems*, 18(8):445–461, 2001. 108

[137] G. Laporte, Y. Nobert, and S. Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Sci.*, 22(3):161–172, 1988. 114

[138] C. E. Noon and J. C. Bean. A lagrangian based approach for asymmetric generalized traveling salesman problem. *Operations Research*, 39(4):623–632, 1991. 114

[139] M. Fischetti, J. S. Gonzlez, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997. 114

[140] D.L. Applegate, R.M. Bixby, and V. Chavatal. *The Travelling Salesman Problem*. Cook, 2006. 114

[141] J.J. Grefenstette, R. Gopal, B.J. Rosmaita, and D.V. Gucht. Genetic algorithms for the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 160–168, Hillsdale, NJ, USA, 1985. 114

[142] C.S. Jeong and M.H. Kim. Fast parallel simulated annealing for traveling salesman problem. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 947–953 vol.3, Jun 1990. 114

[143] Alfonsas Misevic(ius, Jonas Smolinskas, and Aru nas Tomkevic(ius. Using iterated tabu search for the traveling salesman problem. *Information Technology and Control*, 3:2004, 2004. 114

[144] Y. Lien, E. Ma, and B. W. S. Wah. Transformation of the generalized traveling salesman problem into the standard traveling salesman problem. *Information Sci.*, 74(1–2):177–189, 1993 1993. 114

[145] V. Dimitrijevic and Z. Saric. Efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs. *Information Sci.*, 102(1-4):105–110, 1997. 114

[146] A. Behzad and M. Modarres. A new efficient transformation of the generalized traveling salesman problem into traveling salesman problem. In *Proc. of 15th International Conference of Systems Engineering (ICSE)*, pages 6–8, Las Vegas, Nevada, August 2002. 114

[147] Glpk (gnu linear programming kit). http://www.gnu.org/software/glpk/. 116

[148] R. Gomory. Essentials of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 1958. 116

[149] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep. Caltech Concurrent Computation Program 826, California Institute of Technology, USA, 1989. 120

[150] R. Dawkins. *The selfish gene*. Oxford University Press, Oxford, 1976. 120

[151] G.C. Onwubolu and B.V. Babu. *New optimization techniques in engineering*, volume 141 of *Studies in Fuzziness and Soft Computing*. Springer, 2004. 120, 135

[152] D.Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions in Evolutionary Computation*, page 6782, 1997. 120

[153] J. Culberson. On the futility of blind search: An algorithmic view of no free lunch. *Evolutionary Computation*, 6(2):109–128, 1998. 120

[154] D. Goldberg and S. Voessner. Optimizing global-local search hybrids. In Morgan Kaufmann, editor, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, 1999. 120

[155] N. Krasnogor and J. Smith. Competent memetic algorithms: Model, taxonomy and dessing issues. *IEEE Transactions on Evolutionary Compuation*, 9:474—488, 2005. 120

[156] N. Krasnogor. Memetic algorithms. A tutorial given in the seventh International Conference on Parallel Problem Solving from Nature (PPSN VII), September 2002. 120

[157] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Compuation*, 9(5):474—488, 2005. 120, 135

[158] N.J. Radcliffe and P.D. Surry. Formal memetic algorithms. In *Selected Papers from AISB Workshop on Evolutionary Computing*, pages 1–16, London, UK, 1994. Springer-Verlag. 122

[159] C.R. Rao and S.K. Mitra. *Generalized Inverse of Matrices and its Applications*. Wiley, New York, 1971. 127

[160] W.E. Hart. *Adaptive global optimization with local search*. PhD thesis, University of California, San Diego, 1994. 135, 142

[161] Brad L. Miller and David E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212, 1995. 136