Universidad Carlos III de Madrid
www.uc3m.es

TESIS DOCTORAL

A Memetic Approach to the Inverse Kinematics Problem for Robotic Applications

Autor:
Carla Elena González Uzcátegui

Director:
María Dolores Blanco Rojas

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Leganés, 2014

TESIS DOCTORAL (DOCTORAL THESIS)

A Memetic Approach to the Inverse Kinematics Problem for Robotic Applications

Autor (Candidate): Carla Elena González Uzcátegui

Director (Adviser): María Dolores Blanco Rojas

Tribunal (Review Committee)

	Firma (Signature)
Presidente (Chair):	____________
Vocal (Member):	____________
Secretario (Secretary):	____________

Título (Degree): Doctorado en Ingeniería Eléctrica, Electrónica y Automática
Calificación (Grade): ________________

Leganés, de de

"Oh Yoshimi, they don't believe me
But you won't let those robots defeat me "

— The Flaming Lips

Acknowledgements

First and foremost, I would like to thank Prof. M. Dolores Blanco for her guidance, support, patience, and encouragement at times of doubt and uncertainty. For that, I will be forever grateful.

I also would like to extend my gratitude to the systems engineering and automation department at the University Carlos III of Madrid. I have been fortunate for having shared these years with a group of very talented people, especially with those who also offered me their friendship: María, Concha, Rakel, Julio, Arnaud, Álvaro, Javi, Fernando, and Ana, just to name a few.

A special acknowledgement to the researchers and staff of the Institute for Systems and Robotics in Lisbon for a rewarding professional and personal experience.

To my friends Teresa and Pablo. If friends are the family one chooses in life, then they are my dearest family members. Life has gifted me with their friendship and companion. Teresa's immeasurable solidarity and kindness and Pablo's laughter and joy have made the hard times bearable, and the good times better.

Finally, I would like to dedicate this thesis to my loving family and especially to my mother. Everyday her unconditional support and love vanish the ocean between us to make me feel the warmth of home.

Abstract

The inverse kinematics problem of an articulated robot system refers to computing the joint configuration that places the end-effector at a given position and orientation. To overcome the numerical instability of the Jacobian-based algorithms around singular joint configurations, the inverse kinematics is formulated as a constrained minimization problem in the configuration space of the robot. In previous works this problem has been solved for redundant and non-redundant robots using evolutionary-based algorithms. However, despite the flexibility and accuracy of the direct search approach of evolutionary algorithms, these algorithms are not suitable for most robot applications given their low convergence speed rate and the high computational cost of their population-based approach. In this thesis, we propose a memetic variant of the Differential Evolution (DE) algorithm to increase its convergence speed on the kinematics inversion problem of articulated robot systems. With the aim to yield an efficient trade-off between exploration and exploitation of the search space, the memetic approach combines the global search scheme of the standard DE with an independent local search mechanisms, called *discarding*. The proposed scheme is tested on a simulation environment for different benchmark serial robot manipulators and anthropomorphic robot hands. Results show that the memetic differential evolution is able to find solutions with high accuracy in less generations than the original DE.

Resumen

La cinemática inversa de los robots manipuladores se refiere al problema de calcular las coordenadas articulares del robot a partir de coordenadas conocidas de posición y orientación de su extremo libre. Para evitar la inestabilidad numérica de los métodos basados en la inversa de la matriz Jacobiana en la vecindad de configuraciones singulares, el problema de cinemática inversa es definido en el espacio de configuraciones del robot manipulador como un problema de optimización con restricciones. Este problema de optimización ha sido previamente resuelto con métodos evolutivos para robots manipuladores, redundantes y no redundantes, obteniéndose buenos resultados; sin embargo, estos métodos exhiben una baja velocidad de convergencia no adecuada para aplicaciones robóticas. Para incrementar la velocidad de convergencia de estos algoritmos, se propone un método memético de evolución differencial. El enfoque de búsqueda directa propuesto combina el esquema estándar de evolución diferencial con un mecanismo independiente de refinamiento local, llamado *discarding* o descarte. El desempeño del método propuesto es evaluado en un entorno de simulación para diferentes robot manipuladores y manos robóticas antropomórficas. Los resultados obtenidos muestran una importante mejora en precisión y velocidad de convergencia en comparación del método DE original.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Kinematics is one of the most fundamental aspects of any robot system. From design and calibration to simulation and control, a well defined kinematics plays a key role in the efficient planning and accurate completion of the tasks commanded to the robot.

Robot tasks can be manifold, but generally they are described as a set of target trajectories in the Cartesian space. Accordingly, to perform the task the robot must transform these target trajectories into suitable actuators' actions. From a purely kinematics approach, this implies to map the representation of the desired trajectories into the motion variables of the robot. For instance, for a wheeled mobile robot it means computing the velocities of the motors of each wheel. Whereas, for an articulated robot system it means finding the configuration of all the joints in the kinematic chain. This is known as the *inverse kinematics problem*. Conversely, computing the robot motion in the Cartesian space from the robot variables using the geometric relations between its kinematics components is called the *direct* or *forward kinematics problem*.

Solving these two kinematics problems is at the core of the servo control system of any robot. Their importance has been acknowledged over decades of research devoted to find better and more efficient solutions. Therefore, their analysis represents one of the basic foundations of robotics.

Nonetheless, whilst the forward kinematics problem can be analytically solved using a smart collection of geometric and algebraic tools, the same does not hold true for the inverse kinematics. Although the analytical closed-form solution to this problem can be obtained for some special kinematic geometries, a general and systematic analytical method is not available. The main reason is that for most robots, the inverse kinematics involves solving a system of coupled non-linear equations with any, multiple, or infinite possible solutions, and whose complexity grows with the number of degrees of freedom.

In this dissertation we revisit the inverse kinematics problem to formulate a solution framework based on a direct search algorithm. Such that, instead of directly solving the system of non-linear equations, the inverse kinematics problem is reformulated as a constrained minimization problem defined in the configuration space of the robot. Although the proposed formulation might accommodate the kinematics inversion problem of any type of robot; this dissertation will be limited to the study of the kinematics of serial robot manipulators or a multibody arrangement of them.

A serial robot manipulator is an articulated robot system with an open kinematic chain topology. Kinematically it can be described as an assembly of links connected by means of joints that provide relative motion between adjoining links. One end of the kinematic chain is firmly attached to the base whereas the other end interacts with the environment. Therefore, the inverse kinematics consists in finding the joint coordinates that locate the free end at a given position and orientation coordinates in the Cartesian space.

A large body of work has been devoted to address the kinematics inversion of serial robot manipulators using numerical methods. A conventional, but not exclusive, classification groups these approaches into symbolic elimination, continuation, and iterative methods. Symbolic elimination methods reduce the inverse kinematics problem to a univariate polynomial by analytical manipulation of the system of equations (Raghaven & Roth, 1990; Manocha & Canny, 1994). Continuation methods compute the solution by tracking paths in the complex space from a start system with known solution (Wampler & Morgan, 1991). These methods are efficient and provide all the possible solution to the inverse kinematics problem; however, they are mostly limited to certain kinematic structures of serial robot manipulators. Moreover, they are not easily adaptable when the kinematic layout of the robot changes, e.g., reconfigurable robot manipulators.

In contrast to symbolic elimination and continuation methods, most iterative methods can only converge to one of the multiple solutions of the inverse kinematics problem. However, their formulation admits the solution of a wide range of redundant and non-redundant serial robot manipulators. These methods include Jacobian- and optimization-based methods, among others.

Jacobian methods linearize the inverse kinematics problem based on the differential kinematics equation that relates the end-effector and joint velocities through the inverse Jacobian matrix (Goldenberg et al., 1985). These methods are local and therefore their convergence strongly depends on a good initial estimation of the solution and a suitable integration step-size value (Whitney, 1969a). Moreover, Jacobian-based methods fail convergence at kinematic singular joint configurations, because at these configurations the Jacobian matrix becomes rank-deficient, and thus, not invertible. Damped-least-squares methods tackle the numerical instability around singularities in detriment of motion accuracy by penalizing the norm of the velocity vector with a damping factor, as the residual end-effector error is minimized (Wampler, 1986). In this case, the damping factor value play a important role in the fare trade-off between precision and speed of motion (Nakamura & Hanafusa, 1986; Maciejewski & Klein, 1988).

Optimization-based methods circumvent altogether the singularities issues by solving the inverse kinematics directly on the configuration space of the robot, and consequently the Jacobian is no longer needed for mapping. Instead, the inverse kinematics is formulated as a constrained optimization problem, such that, the solution is obtained as the joint configuration that extremizes an objective function subjected to the restrictions imposed by the mechanical structure of the robot or by the task. This objective function could be defined as the end-effector pose error, energy consumption rate, overall joint displacement, distance from an obstacle, or as a combination of these criteria (Lenarcic, 1985; Wang & Chen, 1991).

For the inverse kinematics problem, optimization methods are robust, reliable, and numerical stable around singularities since they do not depend on the Jacobian matrix. However, most traditional optimization methods explore the solution search space using a point-to-point approach, i.e., in each iteration the algorithm uses local information about the search space landscape to decide which direction is best to explore next. In multimodal optimization, such as the inverse kinematics problem, this strategy may lead to premature convergence towards suboptimal solutions.

Furthermore, in complex tasks in which the robot must meet multiple goals in addition to follow a desired end-effector trajectory, the optimization scheme must be flexible enough to cope with discrete, non-linear, and multi-modal search spaces. In this scenario, population-based optimization algorithms are a good choice to find a global optimal solution for the inverse kinematics problem. In this regard, evolutionary algorithms (EAs) offer a number of advantageous features. First, its flexible search scheme does not impose differentiability conditions on the objective function. Second, constraints are handled directly by the algorithm thus guaranteeing feasibility of the solution. And third, in contrast to the local search approach of traditional optimization methods, evolutionary algorithms are intrinsically global search mechanisms thanks to their population-based search scheme.

In (Parker et al., 1989), a binary genetic algorithm was proposed to solve the inverse kinematics problem of a redundant robot manipulator by minimizing the end-effector's position error and the joint maximum displacement. This formulation was extended to include obstacle avoidance in (Buckley et al., 1997; Nearchou, 1998). In (Yang et al., 2007), a real-coded genetic algorithm was proposed to avoid the inaccuracies of the binary representation of genetic algorithms. In (González et al., 2009) the inverse kinematics associated to each node of an optimal path generation problem was solved based on the differential evolution algorithm. Experimental results showed high accuracy in both end-effector position and orientation. More recently, real-coded evolutionary algorithms combined with *niching* and *clustering* methods have also proved to be an effective approach to obtain the multiple solutions of the inverse kinematics problem of modular robot manipulators (Kalra et al., 2004; Tabandeh et al., 2010).

Despite the overall good performance, the computational complexity of the aforementioned EA-based kinematics inversion methods grows with the number of degrees of freedom and the population size. This, coupled with the inherent low convergence speed of most population-based algorithms makes EAs an unsuitable choice for most real-world kinematic control applications.

However, the global search approach and flexibility of the evolutionary optimization strategy have motivated us to further explore new mechanisms to improve the convergence speed for the inverse kinematics problem of robot manipulators.

We have based our approach on the Differential Evolution (DE) algorithm, a simple direct search scheme that has proved to be an effective optimization tool for several benchmark and real-world application problems. DE is a population-based direct search mechanisms driven by a powerful arithmetic mutation and crossover strategies and a greedy selection mechanism (Storn & Price, 1997). In DE scheme, mutation is a mainly exploratory operator controlled by a constant factor. At early stages mutation keeps the population diverse to efficiently explore the search space, and then gradually clusters the population around the most promising solutions. However, a major emphasis on exploration and the lack of effective exploitation mechanisms leads the process to a low convergence speed rate.

With the goal to improve the convergence speed rate of the standard DE, in this thesis we propose a memetic differential evolution for the inverse kinematics problem of robots, hereafter denoted by δDE (González & Blanco, 2013).

Memetic algorithms (MAs) are defined as any evolutionary meta-heuristic direct search method that embeds an individual learning or local refinement procedure. They owe their name to Richard Dawkins' memes theory. In this theory, Dawkins draws an analogy between the genetic evolution of animals and the cultural evolution of the human society, by defining a unit of cultural information called *meme*. A meme can be an idiom, idea, belief or any other cultural expression that can be transmitted by imitation across generations. In this context, Imitation also conveys the enrichment of the meme by the own culture of each individual during its transmission to other peers. Collectively this gives rise to a form of evolution.

In artificial systems, memetic algorithms aims to balance the collective evolution of a set of artificial entities (global search) with the individual learning of each one (local search) for an effective trade-off between exploration and exploitation of the search space. Similarly, our memetic differential evolution algorithm synthesizes the global exploratory features of the standard DE scheme with a local search mechanism, called *discarding*, to improve the efficiency of the search.

The *discarding* mechanism comprise *migration* and *local search* into a single operator. On the one hand, migration injects new information into the population by replacing poor performing solutions in each iteration (Goldberg, 1989; Moed et al., 1990); and on the other hand, local search provides with a replacement strategy that enhances exploitation in the neighbourhood of the fittest solutions of the population. Originally it was devised as a secondary operator within a differential evolution-based localization filter for mobile robots (Martín et al., 2011; Martín, 2012). Its role was to mitigate the deceleration effect introduced by the noise handling selection operator (*thresholding*).

In these previous works, the intuition behind the *discarding* operator was superficially presented. A first attempt to provide a mathematical formulation was made in (Monar et al., 2010), but the analysis of its control parameters was neglected. The first objective of this thesis is to afford a sound mathematical description of the elements that intervene in the *discarding* mechanism, identify its control parameters and define their role within a memetic search scheme.

Second, analyse the dynamic of the convergence behaviour of δDE in relation of the setting of its control parameters. Given the difficulty of the analytical study of the interaction of the control parameters with the search mechanisms and among themselves, we will rely the analysis on the empirical data obtained from simulation experiments on the inverse kinematics problem of robot manipulators. Therefore, the results will be only valid for this particular optimization problem, and the generalization to other kinematics geometries will be limited.

Third, evaluate the performance of δDE as kinematics inversion method for different arrangements of serial robot manipulators; and compare these results in terms of accuracy, convergence speed, success rate, and computational time with the performance assessment of the standard DE.

Fourth, investigate alternative applications for δDE in the kinematics analysis of articulated robot systems with a large number of joint variables.

1.1 Thesis outline

This dissertation is organized in chapters. Next we provide a brief synopsis of these chapters:

- **Chapter 2**. In this chapter we briefly present the basic notions of robot manipulators and their kinematics components. Further, the forward and inverse kinematics problems are formally defined for these articulated mechanisms together with the concepts of kinematic redundancy and singularity. The last part of the chapter is devoted to summarize the most relevant numerical methods for kinematics inversion found in the literature.

- **Chapter 3**. This chapter provides an introductory overview on the evolutionary optimization paradigm. In the first part of the chapter, the search principles of this biological-inspired optimization approach are described using three different perspectives: Genetic Algorithms (GAs), Evolution Strategies (ESs), and Evolutionary Programming (EP). Once a general overview on the common mechanisms of the evolutionary search scheme has been provided, we proceed to describe in detailed the operators and control parameters of the Differential Evolution (DE) algorithm.

- **Chapter 4**. The proposed memetic differential evolution scheme (δDE) is described in this chapter. First, a survey on the state-of-the-art of the acceleration mechanisms of the differential evolution scheme is presented. Within this context, a novel acceleration mechanism, called *discarding*, is introduced. The search behaviour of the proposed memetic algorithm is illustrated with practical examples on three benchmark objective functions.

- **Chapter 5**. In this chapter we provide the mathematical formulation of the kinematics inversion of serial robot manipulators as a constrained optimization problem in the configuration space of the robot. The kinematics inversion solution is presented as the joint configuration that minimizes the position and orientation error of the end-effector, relative to an arbitrary target pose coordinates defined in the workspace of the robot. In addition, the same formulation is extended to a set of consecutive target pose coordinates, known as the *path generation problem*.

- **Chapter 6**. This chapter presents a comparative evaluation between the proposed memetic scheme (δDE) and the standard DE as kinematics inversion methods. The simulation-based study compares the performance of both algorithms in terms of accuracy, success rate, and convergence speed on three benchmark serial robot manipulators.

- **Chapter 7**. In this chapter we address the role of the control parameters setting in the convergence behaviour of the memetic approach as kinematics inversion method. δDE has six different control parameters, three are diversification parameters since they control the exploratory variation mechanisms (standard DE); and the other three are intensification parameters that regulate the local search mechanism (discarding). Based on empirical simulation data we explore the influence of the intensification parameters under different diversification scenarios.

- **Chapter 8**. In this chapter we examine the performance of the δDE on kinematics applications with a higher-dimensional search space. Here we consider the inverse kinematics problem involved in two different tasks for anthropomorphic robot hands. The first task consists in the path generation problem of simultaneously bending all the fingers to make a fist. The second tasks reproduces a set of routines used for the clinical evaluation of the human hand mobility, called the Kapandji method.
- **Chapter 9**. In this chapter we draw some conclusions on the performance of the memetic differential evolution scheme δDE, and mark the route to future research and developments.

Chapter 2

Notions of Robot Kinematics

2.1 Robot Manipulators

A robot manipulator is a kinematic chain set as an assembly of rigid bodies (links) connected by means of kinematic pairs (joints) that provide relative motion between adjoining links. In a typical configuration, one end is rigidly constrained to a fixed point and the other is attached to a customized tool (end-effector) that interacts with the environment. The end-effector could be any device that can be installed at the robot wrist to perform an specific task. In industrial applications some of the most common end-effector types include: grippers, cutting and drilling tools, welding torches, force/torque and collision sensors, etc.

Joints are essentially of two types: revolute and prismatic. Both are kinematic pairs with one-degree of freedom, such that, their motion is constrained to single-axis rotation or linear displacement, respectively. From a kinematic point of view any joint with more than one degree of freedom can be easily simplified as an arrangement of revolute and/or prismatic joints connected by zero length links (e.g., screw and ball joints).

Robot manipulators can be divided according to their kinematic topology into open and closed kinematic chains. An open kinematic chain is an articulated structure with a serial arrangement of links and joints, as illustrated in Figure 2.1. Note that the degrees of freedom (DOF) of a serial robot manipulator are determined by the total number of joints in the open kinematic chain. In contrast, closed kinematic

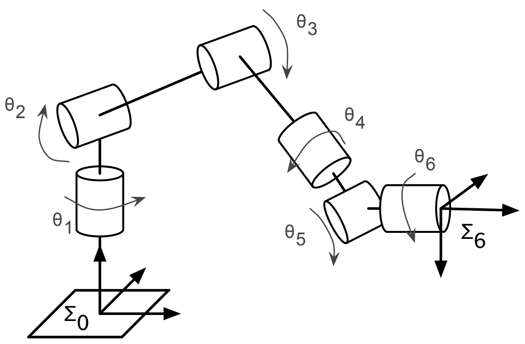


Figure 2.1: Open kinematic chain configuration of a robot manipulators.

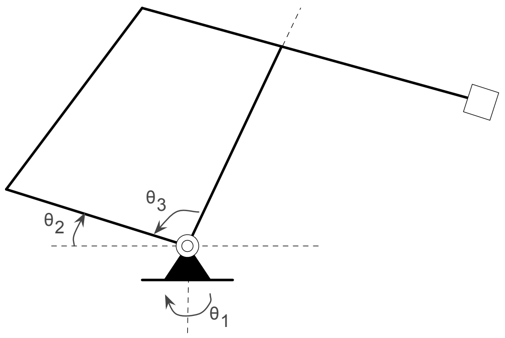


Figure 2.2: Closed kinematic chain configurations of robot manipulators (Somolinos et al., 2002).

chains show loops in their mechanical structure, as depicted in Figure 2.2.

Serial robotic arms are widely extended in industrial applications since they provide a relatively large workspace with a compact mechanical structure. In particular, the large workspace and high dexterity of anthropomorphic robot manipulators (Figure 2.1) have shown to be advantageous in complex manipulation tasks. Nonetheless, a heavyweight structure, low stiffness and low effective load are also inherent features to their open kinematic configuration. Hence, alternative kinematic structures with higher complexity have been introduced in an attempt to increase stiffness and to reduce the total weight of the mechanical structure. In Figure 2.2 such mechanism is illustrated. In this example, the closed kinematic configuration allows to locate all actuators at the base of the structure which yields lightweight robot manipulators suited for high speed operation tasks. For simplicity, in this document only serial robot manipulators with revolute joints will be considered.

2.2 Kinematics of Robot Manipulators

Given a serial robot manipulator with N degrees of freedom, each joint in the kinematic chain is represented by a variable q_j in the robot manipulator configuration space $\mathcal{Q} \subseteq \mathbb{R}^N$. Such that, at any time instant the overall robot configuration

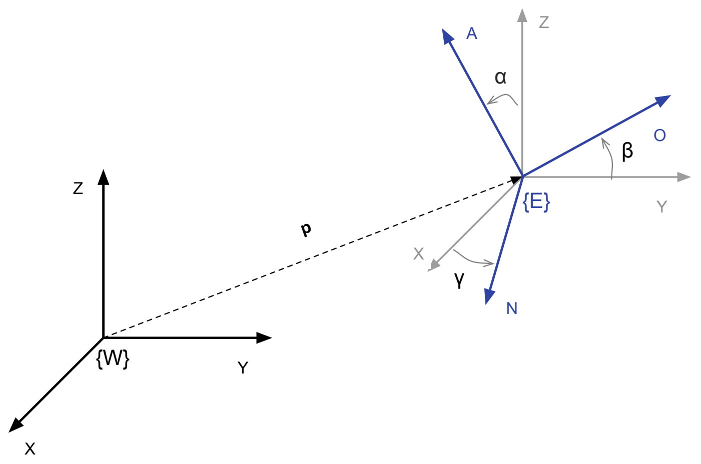


Figure 2.3: Definition of the position and orientation of a rigid body.

can be defined by a vector

$$\mathbf{q} = (q_1 \dots q_j \dots q_N)^\top ,$$

where $\mathbf{q} \in \mathcal{Q}$. In practice, joint variables are bounded values within the angular or linear displacement range of the joint mechanical actuators.

During the performance of a task the end-effector is the interacting tool of the robot manipulator with the environment. It is therefore convenient to define robot tasks in the Cartesian space in terms of target *pose* coordinates, i.e., position and orientation of the end-effector. According to rigid body transformations, the translation and rotation of a given reference frame E attached to a body (end-effector) can be described relative to a fixed inertial reference frame W by the pose coordinates

$$\xi = (\mathbf{p}, \mathbf{R})$$

where $\mathbf{p} \in \mathbb{R}^n$ is the position vector of the origin of frame E from the origin of frame W, and the rotation matrix $\mathbf{R} \in SO(n)$ represents the orientation of frame E relative to frame W (see Figure 2.3). It follows that the robot task space is defined as $\mathcal{C} \subseteq SE(n) = \{(\mathbf{p}, \mathbf{R}) \mid \mathbf{p} \in \mathbb{R}^n, \mathbf{R} \in SO(n)\}$, where n is dimension of the task space, and $m = n(n+1)/2$ is the minimal number of task variables necessary to completely define any arbitrary pose in $\mathcal{C}$. For robot manipulators only the two- and three-dimensional cases apply, i.e., $n = 2, 3$.

In the most general case, the robot manipulator moves in a three-dimensional task space ($n = 3$), such that, the end-effector's position is defined by the vector

$$\mathbf{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}, \tag{2.1}$$

and the orthonormal rotation matrix

$$\mathbf{R} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix} \tag{2.2}$$

represents the end-effector orientation. Vectors $\mathbf{n}$, $\mathbf{o}$, $\mathbf{a}$ are mutually orthogonal unit vectors, and therefore, only three independent variables are needed to represent the nine parameters of the rotation matrix: Euler angles, Roll-Pitch-Yaw angles, etc. Therefore, in a three-dimensional task space the end-effector's pose can be completely defined with six variables, three in translation and three in rotation.

Conventionally, these pose coordinates are compactly represented as homogeneous transformation matrices of the form

$$T = \left(\begin{array}{ccc|c} & & & p_x \\ & \mathbf{R} & & p_y \\ & & & p_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right). \tag{2.3}$$

Another aspect to be considered in the kinematic analysis of robot manipulators is the kinematic redundancy. This occurs when the number of joints variables is greater than the number of task space variables ($N > m$), meaning that a given end-effector pose would correspond to infinite joint configurations in $\mathcal{Q}$. Functional redundancy can be regarded as a concept mainly inherent to the task and not to the robot. Given that, the same robot manipulator can be redundant for one task and non-redundant for another, e.g., a 6DOF robot manipulator is non-redundant for any task in the three-dimensional Cartesian space requiring a certain end-effector position and orientation ($m = 6$). However, it becomes redundant for any task in the two-dimensional Cartesian plane ($m = 3$). On the other hand, intrinsic redundancy occur when the robot manipulator has more than the six degrees of freedom required to position and orient the end-effector in the three-dimensional space ($n > 6$). Further, in some applications kinematic redundancy is a desirable feature since it increases dexterity and enhances the tractability of secondary goals, such as, collision avoidance or joint saturation, without modifying the main task, e.g., following a prescribed end-effector path.

From a kinematic point of view the motion of an articulated body can be described by the functional mapping between the configuration and Cartesian space variables, which in turn arise two different kinematics subproblems, namely: *forward* and *inverse* kinematics.

2.3 Forward Kinematics Problem (FK)

The forward kinematics problem refers to computing the end-effector pose ξ given a known joint configuration vector $\mathbf{q}$, this is:

$$\xi = fk\left(\mathbf{q}\right), \tag{2.4}$$

where $fk : \mathcal{Q} \rightarrow \mathcal{C}$ is the forward kinematics function.

For most serial robot manipulators the forward kinematics problem can be analytically solved with the Denavit-Hartenberg (D-H) convention and matrix transformation algebra (Denavit & Hartenberg, 1955). The D-H convention establishes a set of rules to assign a suitable reference frame to each link in the kinematic chain, such that, each joint-link pair j is characterized by four parameters, namely: link length a_j, link offset d_j, joint twist α_j, and joint angle θ_j. These four parameters are used to build homogeneous transformation matrices expressing the relative position and orientation between two consecutive links. Once all transformation matrices are defined, the forward kinematics problem can be solved as

$$T_N^0 = T_1^0 \cdots T_j^{j-1} \cdots T_N^{N-1},$$

which relates the position and orientation of the base and end-effector frames. Matrices T_j^{j-1} are of the form

$$T_j^{j-1} = \begin{pmatrix} c\theta_j & -c\alpha_j s\theta_j & s\alpha_j s\theta_j & a_j c\theta_j \\ s\theta_j & c\alpha_j c\theta_j & -s\alpha_j c\theta_j & a_j s\theta_j \\ 0 & s\alpha_j & c\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

if joint j is revolute, and as

$$T_j^{j-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_j & -s\alpha_j & 0 \\ 0 & s\alpha_j & c\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

if joint j is prismatic[1]. A comprehensive overview on the D-H convention and transformation matrices algebra can be found in (Sciavico & Siciliano, 1996).

2.4 Inverse Kinematics Problem (IK)

Robots tasks are naturally defined as a set of pose, velocity, and acceleration trajectories in the Cartesian space. To attain these trajectories in a given time interval, the servo control system has to rapidly generate the corresponding joint actuators commands. Therefore, the inverse kinematics can be stated as the problem of computing the set of joint variables $\mathbf{q}$ corresponding to a given end-effector pose ξ, expressed as

$$\mathbf{q} = fk^{-1}(\xi)\,. \tag{2.5}$$

Obtaining analytical closed-form solutions for the inverse kinematics problem is not a trivial task, since it involves solving a system of coupled non-linear equations with any, multiple, or infinite possible solutions. Only when the robot manipulator has intersecting and/or parallel consecutive joint axes, it is possible to express the kinematics inversion problem as a simplified decoupled system of equations (Paul

[1] $c\phi$ and $s\phi$ are the shorthand notation for $\cos(\phi)$ and $\sin(\phi)$, respectively.

& Shimano, 1978; Elgazzar, 1985; Kucuk & Bingul, 2004). Further, quaternion theory has been successfully used to obtain the analytical inverse kinematics of robot manipulators, however, its use has been limited to simple non-redundant robots (Aydin & Kucuk, 2006). More recently, an analytical general solution has been proposed for solving the inverse kinematics problem of 6DOF robot manipulator with rotational joints (Vasilyev & Lyashin, 2010).

The lack of analytical procedures to solve the inverse kinematics of general kinematics structures has motivated the use of numerical methods. These are conventionally classified into: symbolic elimination, continuation, and iterative methods.

Symbolic elimination methods transforms the inverse kinematics problem into univariate polynomials by analytical manipulation of the system of non-linear equations (Raghaven & Roth, 1990; Manocha & Canny, 1994). The main advantage of this method is that by solving the resulting polynomial equation one is able to find all possible solutions to the inverse kinematics problem. Continuation methods cast the solution to the inverse kinematics as tracking paths in the complex space from a start system with a known solution (Wampler & Morgan, 1991). Although these methods are efficient in finding all the possible solutions to the inverse kinematics problem they are limited to certain kinematics structures of serial robot manipulators. Moreover, the heavy mathematical manipulation of these methods does not allow to easily adapt them when the kinematics layout of the robot manipulator changes, e.g., reconfigurable robot manipulators.

Iterative methods comprise many different approaches and their main advantage is that they admit the solution of the inverse kinematics problem of complex kinematics structures. Among the many iterative methods for kinematics inversion here we will focus on the two most common approaches: Jacobian and optimization-based methods.

2.4.1 Jacobian-based Methods

Most of numerical methods for kinematics inversion are based on the differential kinematics equation

$$\dot{\xi} = J\left(\mathbf{q}\right)\dot{\mathbf{q}}, \tag{2.6}$$

that establishes a linear mapping between joint velocities $\dot{\mathbf{q}}$ and end-effector velocities $\dot{\xi}$, given by the robot Jacobian matrix $J \in \mathbb{R}^{m\times N}$.

Thus, if J^{-1} is defined, the joint velocity vector is obtained from (2.6) as:

$$\dot{\mathbf{q}} = J\left(\mathbf{q}\right)^{-1} \dot{\xi}, \tag{2.7}$$

which is commonly known as the resolved-motion rate-control method (Whitney, 1969b).

One can obtain the joint configuration vector by formulating (2.7) as a numerical integration problem and using the Newton-Raphson method to find a solution (Goldenberg et al., 1985). However, its convergence strongly depends on a good initial estimation of the solution and a suitable integration step-size value (Whitney, 1969a). Moreover, the expression (2.7) only holds when the number of joint variables and the number of independent end-effector task variables are equal ($N = m$), i.e., the Jacobian matrix is a full-ranked square matrix. Instead, if $N < m$, no solution exist for the inverse kinematics problem since the number of joints is not sufficient to generate an arbitrary pose at the end-effector; whereas, if $N > m$, the robot manipulator is said to be kinematically redundant and the inverse kinematics problem has infinite solutions.

In case of redundancy, the kinematics inversion is solved as an optimization problem by defining a secondary vector in the null space of J and then computing the pseudo-inverse Jacobian matrix $J^{\dagger}$[2] (Goldenberg et al., 1985). A modified approach consist in setting these secondary constraints as additional rows in the Jacobian matrix in order to build an invertible square matrix, known as the extended Jacobian (Baillieul, 1985).

Singularities

Jacobian-based methods are particularly sensitive to joint configurations at which the Jacobian matrix becomes rank-deficient, and therefore, not invertible. These configurations are called kinematic singularities and occur when one or more joints no longer represent independent variables, which consequently leads to the reduction of the structure mobility. Furthermore, in the neighbourhood of such configurations the Jacobian matrix is ill-conditioned and all Jacobian-based inverse kinematics methods become unstable, that is, the solution demands large joint velocities

[2]The pseudo-inverse or Moore-Penrose generalized inverse matrix $J^{\dagger}$ of matrix $J(\mathbf{q})$, is defined as: $J^{\dagger} = J^{\top}(JJ^{\top})$, which minimizes the Euclidean norm of $\mathbf{q}$.

even for small end-effector displacements.

Based on the virtual work or generalized forces principle[3], singularities issues can be tackled by replacing the inverse Jacobian matrix with the transpose Jacobian $J^\top$ (Wolovich & Elliott, 1984). The obtained solution is interpreted as an approximation of the robot manipulator tendency of motion instead of the actual inverse kinematics solution. Additional drawbacks are the slow convergence rate, scaling issues, and instability in the vicinity of singularities.

An alternative and more effective approach to address singularities is to formulate the inverse kinematics as a damped least-squares problem. Such that, when close to a singular configuration the norm of the velocity vector is penalized by a damping factor as the residual end-effector error is minimized (Wampler, 1986). In this case, the damping factor value has to be carefully chosen in order to achieve a fare trade-off between precision and speed of motion. In (Nakamura & Hanafusa, 1986; Bensalah et al., 2014), this value is adjusted based on the manipulability measure, which gives a measure of how close a joint configuration is from a singularity (Yoshikawa, 1991). Other selection criteria include, minimum singular value of J (Maciejewski & Klein, 1988) and tracking deviation minimization (Deo & Walker, 1995; Phuoc et al., 2008). A comprehensive overview on damped least-squares-based methods for kinematic inversion can be found in (Chiaverini et al., 1994; Deo & Walker, 1995).

2.4.2 Optimization-based Methods

Although damped least-squares methods circumvent the numerical instability in the vicinity of singularities, Jacobian-based algorithms still fail to converge at singular joint configurations. An alternative approach is to formulate the inverse kinematics as a constrained optimization problem, i.e., to find a joint configuration vector that minimizes a cost function subjected to the restrictions imposed by the mechanical structure or by the task.

Several optimization schemes have been used to solve this non-linear programming problem, such as: gradient-based methods (Lenarcic, 1985), heuristic direct-search methods (Wang & Chen, 1991), artificial neural networks (Guez & Ahmad,

[3] $F^\top.\triangle\xi = \tau^\top\triangle\mathbf{q}$, where F and τ are the end-effector and joints generalized force vector, respectively.

1988; Kuroe et al., 1993; Morris & Mansor, 1997; Karlik & Aydin, 2000; Bingul et al., 2005; Hasan et al., 2006; Morris & Mansor, 1998; Driscoll, 2000; Zhang et al., 2005; Chiddarwar & Babu, 2010), and evolutionary algorithms (Parker et al., 1989; Kim & Kim, 1996; Tabandeh et al., 2006), among others. The main advantages of these approaches over the Jacobian-based algorithms, are their robustness and numerical stability around singularities.

The conjugate gradient method was used to solve the inverse kinematics optimization problem by iteratively minimizing the position and orientation error until some maximum error threshold (Lenarcic, 1985). Since this method is driven by the calculation of the gradient value, only continuous and differentiable objective functions can be considered. Moreover, an approximation of the inverse Jacobian matrix is used as a weighting factor and neither redundancy nor kinematic singularities are solved.

On the other hand, cyclic coordinate descent method (CCD) implements a different approach to solve the inverse kinematics of an NDOF robot manipulators (Wang & Chen, 1991). This is, instead of solving an N DOF optimization problem altogether, the inverse kinematics problem is formulated as N optimization problems of 1 DOF, i.e., one per each joint in the kinematic chain. The algorithm is divided into cycles, and each cycle is further divided into N steps. In each step the inverse kinematics equation for one joint (1DOF) is solved analytically by minimizing its individual contribution to the end-effector's total pose error. Moreover, since it does not require the Jacobian matrix calculation, the method is not ill-conditioned by singularities. In order to improve accuracy and convergence speed, CCD has been used in combination with the Broyden-Fletcher-Shanno method (BFS) (Wang & Chen, 1991).

Besides conventional optimization methods, the inverse kinematics problem has also been addressed by artificial intelligence techniques. In this regard, artificial neural networks (ANNs) have been widely used to represent the inverse kinematics equation of robot manipulators due to their accurate representation of highly non-linear functional relationships.

An artificial neural network is first trained using forward or inverse kinematics information as training data. This data comprises input-output information obtained either from a model or experimentally. Further, a training mechanism adjusts the neurons connections weights that minimize the prediction error. Thus, once trained, the network is able to predict the robot joint configuration when a given end-effector pose is presented as input. Since the training phase involves an optimization process, the computational complexity and accuracy of the network lie on the minimization method, commonly based on the Levenberg-Marquardt algorithm. Moreover, good accuracy requires large data sets which might lead to slow convergence speed. Among the several ANN architectures for kinematic inversion found in the literature, multi-layer backpropagation neural networks (Guez & Ahmad, 1988; Kuroe et al., 1993; Morris & Mansor, 1997; Karlik & Aydin, 2000; Bingul et al., 2005; Hasan et al., 2006) and radial basis function networks (RBF) (Morris & Mansor, 1998; Driscoll, 2000; Zhang et al., 2005; Chiddarwar & Babu, 2010) are the most commonly used. Reported experimental results have shown a better performance when different networks are trained for each joint variable, e.g., N networks for NDOF, which in turn carries out a heavy computational cost for training and validation.

Most traditional optimization methods explore the search space using a point-to-point approach, i.e., in each iteration the algorithm uses local information about the search space landscape to decide which direction is best to explore next. In particular, in multimodal problems, such as the inverse kinematics of robot manipulators, a point-to-point approach might lead to premature convergence towards suboptimal solutions. In this scenario, population-based optimization algorithms are more likely to find a global optimal solution due to their parallel search scheme.

Based on the above analysis, J. Parker proposed a population-based binary genetic algorithm to solve the inverse kinematics as the minimization of the end-effector position error and the joint maximum displacement (Parker et al., 1989). Similarly, in (Buckley et al., 1997; Nearchou, 1998) binary genetic algorithms were proposed to solve the inverse kinematics of highly redundant robot manipulators with obstacle avoidance. Under this approach, singularities are avoided because the search is performed within the robot configuration space and the Jacobian matrix is no longer used for mapping. However, binary coding of real parameters introduces accuracy errors. Real-coded evolutionary algorithms are more suitable to solve the inverse kinematics problem as shown in (Yang et al., 2007). In addition, the global search scheme of real-coded evolutionary algorithms combined with *niching* and *clustering* methods have proved to be an effective mechanism to obtain the multiple solutions of the inverse kinematics problem of standard and modular robot manipulators (Kalra et al., 2004; Tabandeh et al., 2010).

Evolutionary algorithms global search approach and flexibility have motivated us to explore new approaches to the inverse kinematics using population-based algorithms. In the following chapters, Differential Evolution (DE) (Price et al., 2005) is used to solve the inverse kinematics of robot manipulators under a multi-objective optimization model, to minimize the end-effector position and orientation given a desired pose. DE is a population-based search algorithm driven by an arithmetic mutation operator combined with crossover and selection mechanisms.

2.5 Summary

The foregoing overview did not aim to be an exhaustive record on the kinematic inversion mechanisms found in the literature, but a brief summary of the most important methods used to tackle the inverse kinematics problem as the cornerstone of the kinematics control of robot manipulators. Since obtaining a closed-form solution is not possible for most kinematic structures, several numerical approaches have been proposed for redundant and non-redundant robots. Therefore, two key aspects should be considered when evaluating the algorithm performance: (i) convergence speed, and (ii) computational cost.

Jacobian inversion methods linearise the inverse kinematics problem and solve it recursively using the robot's inverse Jacobian matrix. Nonetheless, these methods are local and thus its convergence greatly depends on a good initial condition estimation. More importantly, at singular configurations convergence is not attained since the Jacobian matrix becomes rank-deficient, whereas that at the vicinity of such configurations the solution demands unfeasible joint velocities.

Artificial Neural Networks (ANNs) are also a popular approach for kinematic inversion. Their structure allow representing complex non-linear functional relationships from partial input-output data. This data is used to train the network by modifying the neurons weights such that the output prediction error is minimized.

However, the optimization mechanism used for training requires large data sets and usually leads to estimation errors, especially around singular joint configurations.

To avoid singularities issues, the inverse kinematics is formulated as a constrained optimization problem in the configuration space, such that, the Jacobian matrix is no longer needed for mapping. Given a desired end-effector pose, the solution is obtained as the joint configuration that extremizes an objective function defined according to the task. Traditional methods such as gradient descent algorithms provide an efficient and stable solution to the inverse kinematics, however, gradient-based methods tend to converge prematurely and impose constraints on the objective function. Hence, these methods are only suitable for simple tasks.

Nevertheless, for complex tasks in which the robot must meet multiple goals in addition to follow a desired end-effector path, the optimization scheme must be flexible enough to cope with discrete, non-linear, and multimodal search spaces. In this regard, population-based optimization approaches, in particular Evolutionary Algorithms (EAs), outperform those based on local point-to-point search mechanisms. Hitherto, an important number of biological-inspired optimization methods as well as major improvements on well-known standard EAs have been proposed to efficiently solve the inverse kinematics problem of robot manipulators.

Chapter 3

Overview on Evolutionary Optimization

3.1 Optimization Problem

Optimization can be regarded as the process of seeking for the best solution to a problem among a set of available alternatives. Such broad definition encompasses any mechanism found in nature, science, or in our daily life in which some features of a system are minimized (energy, time, costs, effort,...), maximized (profits, efficiency, wellness, happiness,...) or both, by setting internal and external factors that interplay with its performance. For instance, computing a minimal time trajectory is a common path planning problem for a mobile robot moving safely amid obstacles, but also to anyone driving somewhere while avoiding traffic and not breaking the speed limit.

From the above definition one can intuitively identify the three main components of an optimization problem. First, the *objective* that is usually expressed as a quantitative measure of the system's feature to be optimized. Second, a set of *variables* or unknown parameters that have to be adjusted to get the optimal value. And third, a set of *constraints* that defines the domain of admissible values that the unknowns variables can take. These three elements set the optimization model.

Let us formalize these concepts by considering the following constrained optimization problem

$$\min_{\mathrm{s}\in\mathcal{S}} f(\mathrm{s}) \quad \text{subject to:} \quad \begin{aligned} & c_i(\mathrm{s}) = 0, \quad i \in \mathcal{E}, \\ & c_j(\mathrm{s}) \geq 0, \quad j \in \mathcal{I}, \\ & \mathcal{S} = \{\mathrm{s} \mid c_i(\mathrm{s}) = 0 \wedge c_j(\mathrm{s}) \geq 0\}. \end{aligned} \tag{3.1}$$

In the above formulation, the design vector s symbolizes the unknown parameters or variables, f is usually a function that represents the optimization objective, and $\mathcal{S}$ denotes the feasible solution space defined by the constraints on the design vector. Here $\mathcal{E}$ and $\mathcal{I}$ are sets of indices, such that, if $\mathcal{E} = \mathcal{I} = \emptyset$ the optimization problem is said to be unconstrained.

Therefore, a vector s^* is said to be an optimal solution of (3.1) , if $\mathrm{s}^* \in \mathcal{S}$ and $f(\mathrm{s}^*)$ attains the objective minimum value. Such optimal solution can be either local or global depending on the domain region over which the solution is defined. This is, a solution vector s^* is a local minimizer of f, if $f(\mathrm{s}^*) \leq f(\mathrm{s})$ for all $\mathrm{s} \in \mathcal{S}$, such that, $\|\mathrm{s} - \mathrm{s}^*\| < \varepsilon$ for some $\varepsilon > 0$. Instead, s^* is said to be a global minimizer if $f(\mathrm{s}^*) \leq f(\mathrm{s})$ for all $\mathrm{s} \in \mathcal{S}$. Furthermore, real-world applications with limited hardware resources also admit near-optimal solutions (feasible solutions with a superior objective function value) since they provide a good compromise between accuracy and computational cost.

An analogous formulation can be drawn for maximization problems since any maximization problem can be formulated as a minimization problem by simply inverting the sign of the objective function. Therefore, hereafter we will assume minimization as optimization goal unless otherwise is explicitly said.

3.2 Optimization Methods

The underlying goal of most optimization problems is to find global optimal, or near-optimal, solutions. The approach used to obtain these solutions basically would depend on the objective function attributes (linearity, differentiability, modality, noise,..). Other important criteria to consider are the constraints conditions (equalities, inequalities or unconstrained), the search space (continuous, discrete, mixed), and, for real-world applications, the computational cost.

Optimization methods comprise a broad-spectrum of techniques, each of them has been designed to tackle a particular class of optimization problem. However, given the constrained and non-linear nature of the optimization problems considered in this dissertation, we will limit the further discussion to numerical optimization.

3.2.1 Numerical Optimization

A numerical optimization algorithm is an iterative method that starts with an initial guess of the solution, and then, by subsequently modifying the current solution, it generates improved trial points (iterates) until the optimal solution is obtained. The criteria followed to generate these iterates are manifold, but they mainly depend on the objective function attributes.

Conventionally, the landscape information provided by the gradient of the objective function is used as search criteria. This means that, if $f(\mathbf{s})$ is differentiable and the gradient $\nabla f(\mathbf{s})$ can be computed or estimated, then the optimal solution can be found by taking subsequent steps along the descent direction pointed by the gradient in each iteration. This strategy is known as the line search method and includes approaches based on the steepest-descent, Newton, quasi-Newton, and non-linear conjugate methods (Nocedal & Wright, 1999).

The main advantage of the line search approach is its local (almost) quadratic convergence speed in solving unconstrained optimization problems for which accurate information about the gradient of the objective function is available. However, the objective function must be unimodal and at least two times differentiable. Nonetheless, if the objective function cannot be expressed in algebraic or analytical form, it is non-smooth, or the gradient calculation is affected by noise, then line search methods might lead to erroneous solutions or do not work whatsoever. In these cases, the alternative is to use zeroth-order methods in which the gradient or higher derivatives of the objective function are no longer required to search for the optimal solutions, e.g., direct search, evolutionary algorithms, particle swarm optimization, among others.

Direct search methods are heuristic-based algorithms that rely on a generate-and-test strategy. This is, each new trial point is compared with the best solution so far, if any improvement is observed, i.e., a lower objective function value, then the trial point is admitted as the current best solution; otherwise, the iterate is rejected and the process is repeated until some improvement is obtained.

The sampling strategy used to generate new iterates can be either deterministic or stochastic. In the stochastic search, new solutions are obtained by adding a random deviation in each coordinate of the current best solution, e.g., random walk; whereas that in the deterministic approach, the search is performed along predefined coordinate axes or towards the direction of the current best solution, e.g., pattern search and simplex method. Nonetheless, in comparison to the derivative-based approaches, direct search methods show a slow convergence speed, and their performance deteriorates as the number of variables increases. Yet, the lack of gradient calculations and its easy implementation give some advantages to direct search over other methods in solving complex optimization problems.

Notwithstanding many advantages of line search and direct search methods, their true strength lies in local optimization. This is, both mechanisms perform the search around the basin of attraction located at the vicinity of the starting point. For multimodal objective functions, i.e., functions that exhibit more than one local minimum, poor initialization combined with a greedy search strategy could lead convergence towards sub-optimal solutions. Consequently, regardless of the starting point, an effective sampling of the search space becomes essential.

In this regard, population-based approaches provide a solution to the initialization problem by sampling the objective function landscape with a population of starting points that act as independent local optimizers. This might rise the question of how many initial points are necessary to successfully detect the global minimum. In this scenario, evolutionary-based optimization algorithms go a step further by not only setting a population of candidate solutions as parallel local optimizers; but instead, gathering the information contained within the entire population to globally explore the search space.

The first half of this chapter will be devoted to give a brief introduction on the evolutionary optimization approach by using the three main algorithms as guiding threads to explain its basic components. These general concepts are later used in the second half of the chapter as basis to present an overview on Differential Evolution, a population-based direct search mechanism for real-valued and multimodal optimization problems.

3.3 Evolutionary Computation

Evolution in nature can be thought of as the selection process of those organisms in a population that are best adapted to their surrounding environment. Therefore, a powerful optimization mechanism. As the neo-Darwinian theory argues, evolution is the result of the interacting physical mechanisms of reproduction, mutation, competition, and selection over the organisms of a species. The theory states that in a hostile environment with scarce resources, individuals are forced to compete for survival; and only those whose genetic traits are better suited for the environmental conditions, survive (selection) to breed offspring (reproduction). Evolution also benefits from the sudden increase of diversity in the genetic traits pool brought by sporadic random perturbations during reproduction (mutation).

Evidently, evolution in dynamic environments is a much more complex mechanism than the one previously described. Yet, this simplified model has inspired a powerful optimization paradigm known as Evolutionary Computation (EC) that encompasses a set of optimization techniques called Evolutionary Algorithms (EAs).

EAs are meta-heuristic algorithms based on populations of candidate solutions that evolve by means of reproduction, mutation, and selection of the fittest individuals in the population. The main advantage of this search approach over traditional optimization schemes is that, instead of relying on the local information provided by one point in each iteration, these methods use a parallel search mechanism to both explore different promising regions in the search space, and exploit the information encoded within its current elements. These algorithms have been widely used in different research and industrial applications, especially in those wherein standard optimization techniques fail, e.g., optimal control, cognitive modelling, signal processing, robotics, pattern recognition, and recently in drugs design.

To understand the EAs' operating procedure outlined in Algorithm 3.1, consider the constrained minimization problem stated in (3.1). First, the feasible search space is randomly sampled to build an initial population of μ design vectors or candidate solutions. Once initialized, the population undergoes an iterative process of variation and selection until a suitable convergence criterion is met, e.g., cost value threshold, maximum number of functions evaluations, among others. Variation includes all mechanism that modify the information encoded within the current

Algorithm 3.1 Evolutionary Algorithm

```
1: INITIALIZE population.
2: EVALUATE each candidate.
3: while TERMINATION CONDITION $\neq$ true do
4:     SELECT parents.
5:     RECOMBINE parents.
6:     MUTATE obtained offspring.
7:     EVALUATE new candidates.
8:     SELECT individuals for next generation.
9: end while
```

population to generate a new trial population, e.g., recombination and mutation. And selection determines which solutions in the trial population replace those of the current population into the next generation. Throughout the evolutionary process each candidate solution in the current and trial population is assigned with a fitness (or cost) value that represents how close the solution is from the minimum. In each step this value helps the search process to determine which candidates are more likely to reproduce and survive for more generations.

Ultimately, convergence would depend on a critical balance between exploration and exploitation of certain regions of the search space. In this scheme, variation operators are responsible of exploring new regions of the search space, whereas the selection mechanism is responsible of exploiting the genetic traits of the fittest individuals to ensure that the information contained therein is transmitted to future generations.

Conventionally, EAs have been grouped into three main branches, namely: Genetic Algorithms (GAs), Evolution Strategies (ESs), and Evolutionary Programming (EP). These three basic evolutionary approaches emerged from different research fields during the 1960s and early 1970s, and can be summarized as:

- **Genetic Algorithms (GAs)**, were first proposed by J. Holland as an artificial system based on the behaviour of natural systems. Holland seminal work laid the basis for GAs as a set of discrete logical operations acting upon populations of candidate solutions, in a process that emulates the adaptation process of natural systems over successive generations during evolution (Holland, 1962). In this scheme, solutions are encoded as binary strings of **0**'s and **1**'s with fixed length. Nowadays, GAs are consider a powerful optimization tool.

- **Evolution Strategies (ESs)**, emerged from the research on parametric optimization of aero- and hydro-dynamic components using parameter discrete random variation. Later, this idea was extended into a stochastic mutation scheme framed as an evolutionary algorithm mainly used in complex real-valued parameter optimization problems (Rechenberg, 1973; Schwefel, 1975).

- **Evolutionary Programming (EP)**, was originally pioneered by L.J. Fogel as an attempt to simulate intelligent behaviour with finite-state-machines (Fogel, 1962, 1964). However, it was not until the 1990's that D.B Fogel proposed an optimization mechanism based on the original evolutionary programming algorithm but with a closer approach to the one of evolution strategies (Fogel, 1994).

In the following sections, the key elements of EAs are briefly summarized and the differences among GAs, ESs, and EP are further highlighted. For a more comprehensive survey on EAs the reader is referred to (Bäck & Schwefel, 1993; Fogel, 1994; Bäck, 1996).

3.3.1 Representation

In nature the genetic code or genotype in interaction with the environment determines the physical and behavioural traits of an organism. Similarly, in the context of evolutionary algorithms one can define two different search spaces in which any design vector can be represented. This is, the physical parameters that determine the behaviour of the system are represented in the solution space $\mathcal{S}$ (phenotype), whereas their coding as abstract mathematical objects belongs to the representation space $\mathcal{X}$ (genotype). Since genetic operators manipulate candidate solutions in the representation space and the evaluation is performed in the solution space, it is then necessary to define encoding and decoding functions to map vectors from solution to representation space, and vice-versa (Bäck & Schwefel, 1993; Rothlauf, 2006).

Conventionally, design parameters are represented as l-dimensional arrays, where each parameter is either binary or real-valued encoded, that is, $\mathrm{x} \in \{0,1\}^l$ or $\mathrm{x} \in \mathbb{R}^l$, respectively.

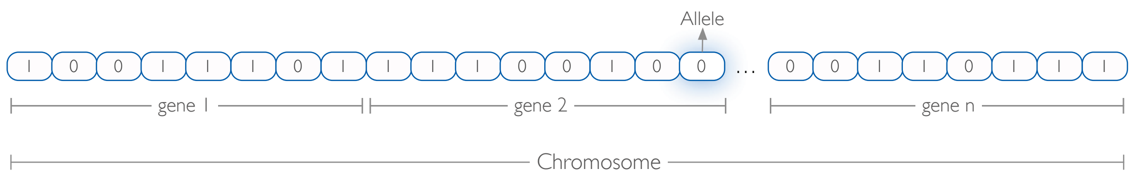


Figure 3.1: Binary representation in genetics algorithms. A chromosome is an array of genes, where each gene is the encoded as a binary string of alleles.

Real-valued representations are more often used by mutation-based EAs, such as evolution strategies and evolutionary programming. In both algorithms the design and control parameters are encoded in a single array to let the algorithm structure evolve with the solution. In ESs, each candidate solution is represented by a vector

$$\mathbf{x} = (\mathbf{s}, \sigma, \alpha),$$

here $\mathbf{s} \in \mathbb{R}^n$ is the design vector, and σ and α represent endogenous strategy parameters. In the most general ESs scheme, $\sigma \in \mathbb{R}_+^{n_\sigma}$ ($n_\sigma \in \{1, \dots, n\}$) is a vector of standard deviations that modifies the variation amplitude during mutation of $\mathbf{s}$, and α is a set of n_α ($n_\alpha \in \{0, \dots, (2n - n_\sigma)(n_\sigma - 1)/2\}$) rotation angles that modify axes orientation to adapt mutation to the search space topology (Bäck, 1996). Analogously, in evolutionary programming each individual is represented as a vector that comprises the parameter design vector and a vector of variances ν, this is:

$$\mathbf{x} = (\mathbf{s}, \nu) = (x_1, \dots, x_n, \nu_1, \dots, \nu_n)$$

where $\mathbf{s} \in \mathbb{R}^n$ is a vector of real parameters and $\nu \in \mathbb{R}_+^n$ is a vector of real positive variances.

On the other hand, binary representation is mostly used by genetics algorithms or crossover-based EAs. According to the building block hypothesis of the schema theory, the sampling search strategy of genetic algorithms is almost optimal due to their binary representation of solutions (Holland, 1975). In this representation each candidate solution is encoded as an array of genes, or chromosome. Each gene represents a parameter in the design vector encoded as a binary string of fixed length of alleles. Figure 3.1 depicts a chromosome with n genes each encoded as eight-bit words.

3.3.2 Initialization

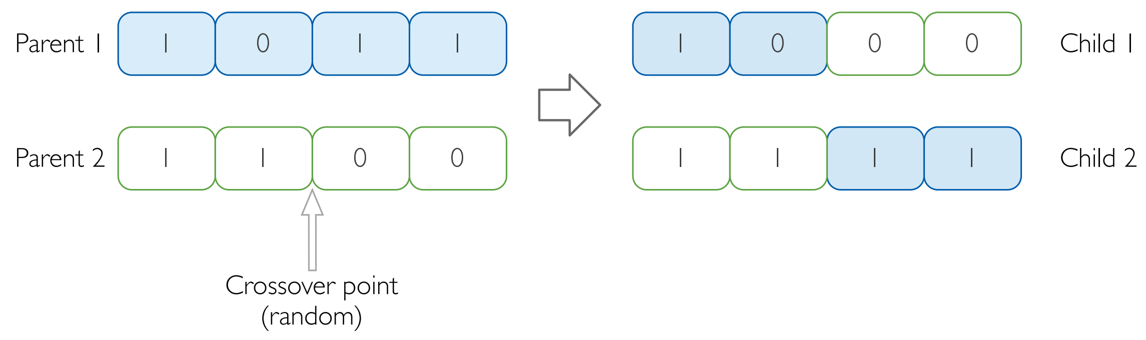
Initialization consist in building a population of candidate solutions by sampling the feasible search space (representation space). In most cases, a convenient procedure is to randomly sample the entire search space and thereby guarantee high diversity in the initial genetic pool. Instead, if prior knowledge about the optimum location is available, then this information could be used as seed in the initialization process.

In GAs, the initialization process is performed by randomly sampling $l \cdot \mu$ times the binary alphabet $\{0, 1\}$. As for ESs, the initial population is built by means of mutation upon a single starting point randomly selected or defined by the user, where small initial values for the standard deviation are suggested, e.g., $\sigma_i = 3.0$ (Bäck & Schwefel, 1993). Finally, EP follows a uniform random distribution for the initialization of object vectors and variances.

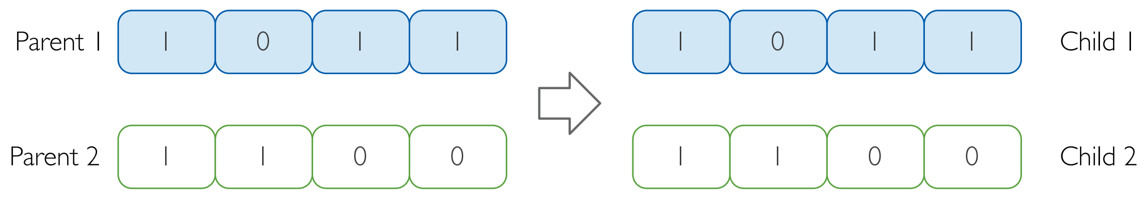
3.3.3 Recombination

Genetic variation operators can be asexual, sexual, or panmictic depending on whether the mechanism involves one, two, or more parents to generate new offspring. Recombination, denoted by $R : X^\rho \rightarrow X^q$, is a sexual or panmictic operator ($2 \leq \rho \leq \mu$). This mechanism emulates the biological process of reproduction by exchanging genetic information encoded in the individuals of the population (parents) to create new candidate solutions (offspring).

In genetic algorithms, recombination is generally sexual between randomly selected chromosomes ($\rho = 2$), and it is triggered by a crossover probability value $p_c \in (0, 1)$. In the simplest case, a uniform random number $\chi \in (0, 1)$ is generated and measured against p_c. If $\chi \leq p_c$, then an integer random crossover point is selected along the bit-string. Once the crossover point is selected, all data beyond that point is swapped between both parents to create two children chromosomes, one of which is discarded at random and the other is kept as a trial candidate solution ($q = 1$). In case that $\chi \geq p_c$, the parent chromosomes are duplicated unaltered (see Figure 3.2). Other discrete recombination mechanisms include multipoint and uniform crossover.



(a) $\chi \leq p_c$



(b) $\chi > p_c$

Figure 3.2: One-point crossover in genetic algorithms. Two chromosomes are randomly selected from the current population as parents. First, a random number χ is generated, (a) if $\chi \leq p_c$ a random crossover point position is selected, then two new chromosomes are created swapping the information encoded in the parent binary strings. (b) if $\chi > p_c$ the offspring is an unaltered duplicate of the parent chromosomes.

Recombination in evolution strategies could be sexual or panmictic among randomly selected parents to generate new offspring ($q = 1$). Accordingly, sexual recombination acts on a pair of vectors ($\rho = 2$) selected anew for each new offspring. Whereas in panmictic recombination one parent is held fixed throughout the process while the second is randomly selected anew from the parent population ($\rho = \mu$) for each component of the offspring. Regardless the number of parents, recombination can be discrete or intermediate. In discrete recombination, each component of the offspring vector is randomly sampled between the parents vectors; whereas in intermediate recombination, offspring's components are calculated as the arithmetic mean value of the corresponding parent's components.

Contrary to the GAs and ESs, in evolutionary programming recombination is omitted in favour to a major emphasis on mutation as the main variation operator driving the evolutionary process. In (Fogel, 1992), Fogel argues that crossover, as

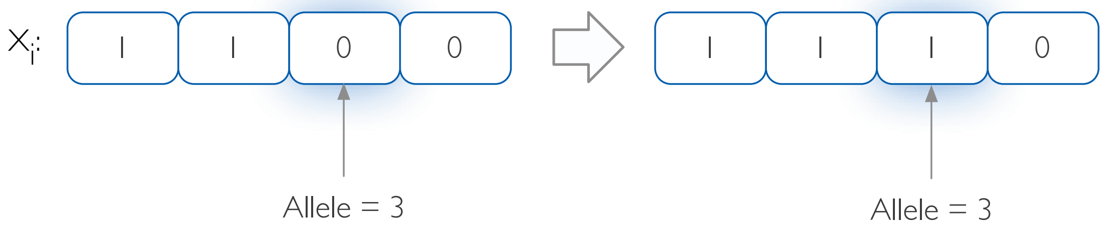


Figure 3.3: Mutation in genetic algorithms. A random value χ is sampled anew for each allele in the binary string, if $\chi \leq p_m$, the allele value is inverted, otherwise, the bit value does not change.

the combination of pieces of genetic code, is negligible during evolution if one considers each individual to represent a phenotype trait. Thus, any tangible variation on the individual is not obtained as a result of specific modification of a gene in the genetic code.

3.3.4 Mutation

Conventionally, mutation is an asexual operator, $M : X \to X$, that introduces random small variations into the genetic code of an individual. The mutation operator aims to maintain the population diversity by adding small perturbations on the individuals and thereby driving the search towards unexplored regions in the search space. Mutation also prevents stagnation, which occurs when the population becomes homogeneous.

GAs mutation is a random mechanism that inverts the value of arbitrary bits (alleles) of a chromosome. Similarly to one-point crossover, this mechanism is triggered by a mutation probability value $p_m \in (0, 1)$, such that, a uniform random number $\chi \in (0, 1)$ is sampled anew for each bit in the binary string and measured against p_m, if $\chi \leq p_m$ the bit value is inverted, otherwise is left unaltered (see Figure 3.3).

In evolution strategies, mutation is a stochastic perturbation mechanism that acts separately upon the design vector s and the strategy parameters σ and α. Mutation on σ uses either an adaptive deterministic criterion, as the $1/5$-success rule (Rechenberg, 1994), or incorporates a stochastic log-normal distributed perturbation criterion; as for α, mutation adds a normally distributed random value. Formally, mutation of the strategy parameters can be expressed as:

$$\sigma'_j = \sigma_j \cdot e^{(\tau \cdot N(0,1) + \tau_1 \cdot N_j(0,1))}$$
$$\alpha'_k = \alpha_k + \beta \cdot N_k(0,1)$$

$\forall j \in \{1, \ldots, n\}$ and $\forall k \in \{1, \ldots, n \cdot (n-1)/2\}$. Here τ, τ_1, and β are constant parameters, and $N(0,1)$ denotes a standard normally distributed random variable. Finally, mutation of s is obtained as:

$$\mathbf{s}' = \mathbf{s} + N(0, \mathbf{C}_v(\sigma', \alpha'))$$

where $\mathbf{C}_v$ is a covariance matrix that controls the mutation step size (Schwefel, 1981). Including strategy parameters into the object vector representation, and by further letting them to undergo variation, endows the algorithm with a self-adaptive component.

On the other hand, EP mutation scheme consists in perturbing the object vector s with a standard normal distribution having zero mean and $\sqrt{\nu}$ standard deviation, ν being the variance value, this is:

$$s'_j = s_j + \sqrt{\nu_j} \cdot N_j(0,1)$$
$$\nu'_j = \nu_j + \sqrt{\zeta \nu_j} \cdot N_j(0,1)$$

$\forall j \in \{1, \ldots, n\}$, where $N_j(0,1)$ is a standard normal distributed random variable, and ζ is an algorithm control parameter.

3.3.5 Selection

Whereas recombination and mutation supply with mechanisms to explore the search space, selection is responsible of exploiting the fittest solutions by promoting them into next generations. Since selection exploits the most promising regions in the search space, the performance of each individual in the population must be adequately measured. To do so, one must define a fitness or cost function that

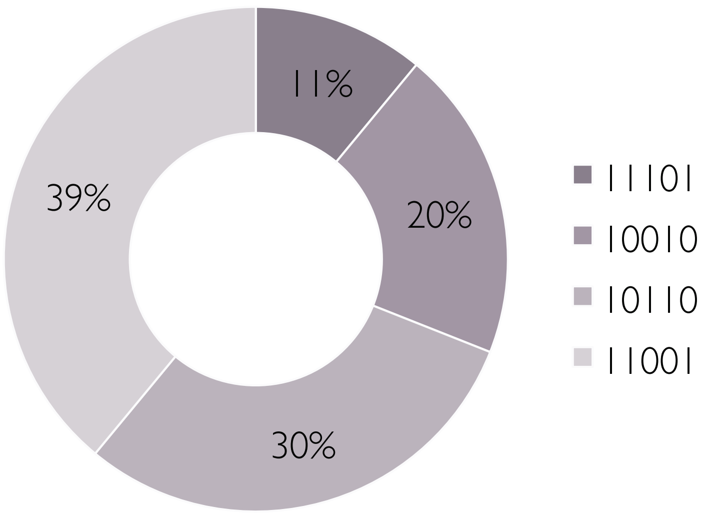


Figure 3.4: Roulette wheel selection method. A population of four chromosomes ($\mu = 4$) and fitness function $f(\mathbf{x}) = \mathbf{x}^2$. In descending order, the selection probability of each chromosome is 0.11, 0.2, 0.3, and 0.39 respectively.

quantifies how close the solution is from the optimal value. For consistency, let us assume that such fitness or cost function is defined by the objective function f.

In genetic algorithms, for each chromosome $\mathbf{x}_i$, $i = 1, \ldots, \mu$, in the population, a selection probability value is calculated as:

$$p_i = \frac{f(\mathbf{x}_i)}{\sum_{k=1}^{\mu} f(\mathbf{x}_k)}$$

where μ is the population size and f is the fitness function. Schematically, selection can be pictured as a roulette wheel partitioned into μ slots whose sizes are proportional to the selection probability of each individual. A new population is built by spinning the wheel μ times, and each time randomly selecting a chromosome from the current population as member of the next generation (see Figure 3.4). Hence, the higher the selection probability the greater the expected numbers of copies of an individual in the new population.

Contrary to the above mechanism, selection in evolution strategies is completely deterministic and based on a fitness ranking of the population members. Basically, two distinctive selection mechanisms can be identified, namely: comma-selection (μ, λ) and plus-selection $(\mu + \lambda)$, where μ denotes the parent population size and λ the size of the offspring population. The difference between both selection mechanisms lies in the number of individuals from the parent and offspring populations involved therein.

Strategy (μ, λ), with $\lambda > \mu$ and $\mu > 1$, builds a new population using the μ best individuals in the offspring population, which means that the parent generation is completely replaced by its offspring in each generation. Obviously, by completely replacing the parent population in each generation one might be discarding good solutions in favour of worse fitted ones. However, in multimodal optimization problems this could be an advantageous strategy to avoid premature convergence to local minima. On the other hand, the selection strategy $(\mu + \lambda)$, $\lambda \geq \mu$, selects the best μ individuals from the union of the parent and offspring populations, which means that well-fitted solutions can spread over more than one generation.

In contrast to ESs, selection in evolutionary programming is stochastic and is always performed over the union of the parents and offspring populations $(\mu + \mu)$. In EP selection, each vector $\mathbf{x}_i$ ($\forall i \in [1, 2\mu]$) competes against $r \geq 1$ vectors randomly chosen and ranked based on the score obtained. In this context, competition is held by comparing the fitness value of $\mathbf{x}_i$ against its competitor's, thereby, the score would be a proportional measure of the number of competitors with worse fitness value. Once all vectors in the population have undergone this process, the μ vectors with the highest score are selected as the new population. This mechanism is called selection by tournament where r is the tournament size.

3.4 Differential Evolution (DE)

Differential Evolution (DE) is a population-based direct search mechanism for real-valued and multimodal optimization problems developed by R. Storn and K. Price (Storn & Price, 1997). DE search scheme accommodates the evolutionary iterative process of mutation, crossover, and selection to steer the population towards convergence.

Similarly to evolution strategies and evolutionary programming, mutation plays a primarily role in the DE search strategy. However, differs on the non-probabilistic

Table 3.1: DE vs. EAs paramater notation.

Parameter		**EAs**	**DE**
Problem Dimension		n	D
Population Size	Parent	μ	N_P
	Offspring	λ	
Crossover rate		p_c	C_R
Mutation scale factor		σ,ν	F

approach used to explore the search space. By contrast, DE mutation is an algebraic operation in which weighted vector differences are added to a base vector to obtain new solutions, thus yielding in an adaptive mutation step-size scheme controlled by a constant factor. In this case, crossover acts as a complementary variation mechanism used to increase the population diversity. Further, DE selection implements a deterministic binary tournament known as one-to-one selection. In the following sections, mutation, crossover, selection, and other components of the DE algorithm will be further explained in detail.

3.4.1 Notation

Before going any further, let us give a brief note on DE notation. In their original work, Storn and Price followed a particular notation to denote DE parameters that differs from the one that has been used to describe EAs in the previous sections. With the aim to be consistent with the DE research community, throughout this document the DE standard notation is adopted. However, to avoid any confusion the equivalence between both notations is listed in Table 3.1.

3.4.2 Representation

Differential Evolution was originally designed to solve real-valued optimization problems. Thus, candidate solutions are encoded as arrays of floating-points variables, such that, given a population of N_P candidate solutions

$$P^g = \{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_{N_P}\}, \tag{3.2}$$

each design vector in the problem search space is represented by a D-dimensional array of the form:

$$\mathbf{x}_i = (x_1, \ldots, x_j, \ldots, x_D)^\top . \tag{3.3}$$

where $x_j \in \mathbb{R}$ represents a variable in the design vector. Indices $i = 1, \ldots, N_P$ and $g = 0, 1, 2, \ldots$ denote the candidate solution index within the population and the running generation number, respectively. Note that contrary to ESs and EP, DE control parameters are not encoded within the candidate solution representation, therefore, their value remain constant during the optimization process.

3.4.3 Initial Population

DE grants mutation with the task of exploring different regions of the search space. This is achieved by perturbing existing solutions with differentials of randomly chosen vectors from the current population, i.e., mutation uses information of already explored regions to create new trial points. Therefore, preserving population diversity is key to avoid stagnation and premature convergence in multimodal problems, especially in early generations of the optimization process.

Therefore, initialization must promote diversity in the initial population by conveniently spreading the candidate solutions over the search space. Unless a priori knowledge about the optimum location is available, a uniform probability distribution is used to randomly initialize $P^{g=0}$, such that, each initial candidate solution is generated as follows:

$$\mathbf{x}_i = \mathbf{x}^{(lo)} + \text{rand}_{1\times D}(0,1)\left(\mathbf{x}^{(hi)} - \mathbf{x}^{(lo)}\right), \tag{3.4}$$

for $i = 1, \ldots, N_P$. Here, $\text{rand}_{1\times D}(0,1)$ is an array of uniformly distributed random numbers $[0,1]$, and $\mathbf{x}^{(hi)}$ and $\mathbf{x}^{(lo)}$ are the parameters' upper and lower bounds, respectively.

The population size N_P influences both, the convergence speed and the computational cost of the algorithm. This is directly linked with the number of functions evaluations per iteration and also with the exploration of the search space. Empirical studies have revealed the effect of the population size in avoiding stagnation and premature convergence (Mallipeddi & Suganthan, 2008). Small values of N_P favour convergence speed but might lead to premature convergence or stagnation.

On the other hand, higher values of N_P might help to avoid premature convergence but increases the computational cost of the algorithm. In general, the size of the population depends on the search space complexity and the dimension D. Based on the value of D, the population size can vary between $2D$ and $100D$, although, a value of $N_P = 20D$ is suitable for most optimization problems (Price, 1999). Recently, Rönkkönen et al. (Ronkkonen et al., 2005) also suggested that typical values of N_P range from $2D$ to $40D$.

3.4.4 Mutation

According to the theory of natural evolution, only the best adapted individuals in a population are prone to reproduce and propagate their genetic traits into future generations. In the EAs context this usually translates into variation mechanisms that favour fitter solutions in the population over poor performing ones. Contrary to this notion, DE variation mechanisms are not biased by the fitness value of individuals in the mating pool; but instead, all candidate solutions are equally likely to undergo mutation and crossover regardless to their fitness value.

DE mutation can thus be defined as an arithmetic panmictic variation operator, $M : X^\rho \to X$, with $\rho \geq 3$. In its most basic implementation, a donor vector is obtained by adding the weighted difference of two vectors to a third ($\rho = 3$). Thus, for each candidate solution $\mathbf{x}_i$, $i = 1, 2, \ldots, N_P$, a mutant vector is generated according to:

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right), \tag{3.5}$$

where $F \in (0, 1+)$ is a real constant scale factor which controls the amplification of the differential variation. The *difference vectors*, $\mathbf{x}_{r_2}$ and $\mathbf{x}_{r_3}$, and the *base vector*, $\mathbf{x}_{r_1}$, are randomly chosen from the current population P^g, such that, $i, r_1, r_2, r_3 \in \{1, \ldots, N_P\}$ are mutually exclusive indices, i.e., $i \neq r_1 \neq r_2 \neq r_3$. DE mutation is illustrated in Figure 3.5 for a three dimensional optimization problem.

DE simple mutation scheme yields a powerful self-adaptive search mechanism, however, a wrong setting of the scale factor value rapidly degrades its performance. Moreover, tuning F is neither intuitive nor a straightforward task, but instead, a highly problem dependant issue as it was reported in (Gämperle et al., 2002; Mallipeddi & Suganthan, 2008). Several studies have thus been carried out to draw some guidelines on how to set this value efficiently. Storn and Price (Storn & Price,

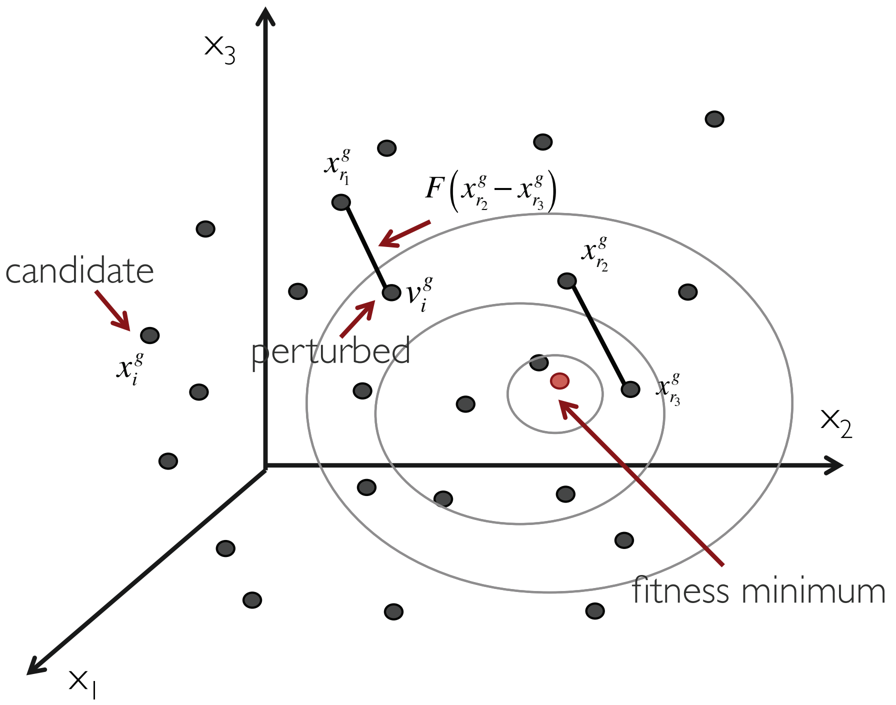


Figure 3.5: Mutation scheme DE/rand/1. The weighted difference vector of two arbitrarily chosen vectors is added to a third vector to obtain the vector $\mathbf{v}$.

1997) suggested that reasonable values of F range between 0.4 and 1, with 0.5 being a good initial choice. Moreover, further research has shown that values less than 0.3 lead to premature convergence whereas $F = 1$ degenerates mutation (Zaharie, 2002; Price et al., 2005). More recently, Rönkkönen et al. set typical values of F within the interval $0.4 < F < 0.95$, with $F = 0.9$ yielding a good compromise between velocity and probability of convergence (Ronkkonen et al., 2005).

One of the most interesting feature of DE mutation is its flexible arithmetic structure. In their seminal work (Storn & Price, 1997), Storn and Price already exploited this attribute by introducing a number of different mutation variants in which vector differentials are used as building blocks. Accordingly, in addition to the basic mutation scheme defined in(3.5), six additional alternative mutation strategies were introduced as listed in Table 3.2. These mutation strategies have the same the arithmetic structure, but can be differentiated by two key aspects: (1) base vector and (2) number of vector differentials. Further, in order to make a clear distinction among them the shorthand notation rule DE/X/Y was introduced;

Table 3.2: Mutation Strategies of Differential Evolution.

Mutation Strategy	**Formula**[1]
`DE/rand/1`	$\mathbf{v}_i = \mathbf{x}_{r_1} + F\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right)$
`DE/best/1`[2]	$\mathbf{v}_i = \mathbf{x}_{best} + F\left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}\right)$
`DE/rand-to-best/1`	$\mathbf{v}_i = \mathbf{x}_i + F\left(\mathbf{x}_{best} - \mathbf{x}_i\right) + F\left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}\right)$
`DE/rand/2`	$\mathbf{v}_i = \mathbf{x}_{r_1} + F\left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}\right) + F\left(\mathbf{x}_{r_3} - \mathbf{x}_{r_4}\right)$
`DE/rand-to-best/2`	$\mathbf{v}_i = \mathbf{x}_{r_1} + F\left(\mathbf{x}_{best} - \mathbf{x}_i\right) + F\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F\left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right)$
`DE/current-to-rand/1`	$\mathbf{v}_i = \mathbf{x}_i + K\left(\mathbf{x}_{r_1} - \mathbf{x}_i\right) + F^{'}\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right)$
`DE/rand/1/either-or`	$\mathbf{v}_i = \begin{cases} \mathbf{x}_{r_1} + F\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right), & \text{if rand(0,1)} < p_f \\ \mathbf{x}_{r_1} + K\left(\mathbf{x}_{r_2} + \mathbf{x}_{r_3} - 2\mathbf{x}_{r_1}\right), & \text{otherwise} \end{cases}$

[1] $\mathbf{x}_{r_1}$, $\mathbf{x}_{r_2}$,$\mathbf{x}_{r_3}$,$\mathbf{x}_{r_4}$, and $\mathbf{x}_{r_5}$ are randomly chosen vectors with $i \neq r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5$.
[2] $\mathbf{x}_{best}$ is the best fitted vector in the current population.

where `X` is replaced by the base vector selection scheme and `Y` by the number of differentials. For example, the standard DE mutation strategy defined in (3.5) is simply denoted as `DE/rand/1`, since the base vector is randomly chosen from the mating pool (***rand***) and one differential is added to the base vector (***1***).

Each mutation scheme in Table 3.2 enhances a different feature of the search strategy. For instance, incorporating the current best vector $\mathbf{x}_{best}$ into the search helps to speed-up convergence by enhancing the greediness of the algorithm, however, it often leads to premature convergence. Furthermore, using more than one differential to perturb the base vector also increases convergence speed but deteriorates correlation. Alternatively, mutation schemes `DE/current-to-rand/1` and `DE/rand/1/either-or` provide a rather generalized variation mechanism since both implement mutation and recombination into a single operator. In the case of `DE/current-to-rand/1`, mutation and arithmetic recombination are merged together to generate rotationally invariant trial vectors as the linear combination of a target vector and a randomly chosen donor. In this scheme, parameter K represents the combination coefficient which has been shown to be effective when its value is randomly chosen from $[0,1]$, whereas $F^{'} = K \cdot F$ is a new constant parameter. On the other hand, scheme `DE/rand/1/either-or` alternates between mutation-only and arithmetic recombination under a stochastic framework controlled by a mutation probability $p_f \in [0,1]$, where K is set equal to $K = 0.5\,(F+1)$ for a given value of F. Evidently, the relative importance of each variation scheme during search (i.e., exploration vs. exploitation) comes determined by the p_f value (Price, 1999).

3.4.5 Recombination

In DE, recombination acts as a complementary variation mechanism to mutation that increases population diversity by incorporating information from the current generation into the intermediary donor population. In standard DE, recombination is a binary discrete recombination mechanism with two variants, namely: binomial and exponential crossover. To differentiate between the two crossover mechanisms a third parameter is introduced into the general notation `DE/X/Y/Z`. Here `Z` is replaced by "`bin`" or "`exp`" if the crossover mechanism is binomial or exponential, respectively.

Both crossover mechanisms are sexual recombination operators, $R : X^2 \rightarrow X$, controlled by a constant crossover rate value $C_R \in [0,1]$. This is, a trial candidate solution is built as the parameter-wise combination of two parent vectors, such that, each parameter is stochastically inherited from either one of the parents.

Lets consider two parent candidate solutions, one from the current population P^g and the other from the intermediary population P_v^g, known as target vector ($\mathbf{x}_i$) and donor vector ($\mathbf{v}_i$), respectively. By binomial crossover both parents are combined into one candidate solution, $\mathbf{u}_i$, as follows:

$$\mathbf{u}_i = \{u_j\} = \begin{cases} v_j, & \text{if } (\text{rand}_j\,(0,1) \leq C_R) \\ x_j, & \text{otherwise} \end{cases}, i = 1, \ldots, N_P. \tag{3.6}$$

Each parameter u_j ($j = 1, \ldots, D$) in $\mathbf{u}_i$ is a copy of the corresponding parameter of either $\mathbf{v}_i$ or $\mathbf{x}_i$. To decide from which parent the child inherits each parameter, a random number $\chi \in (0,1)$ is generated anew and measured against the crossover rate value C_R, if $\chi \leq C_R$ then the parameter is inherited from the mutant vector $\mathbf{v}_i$,

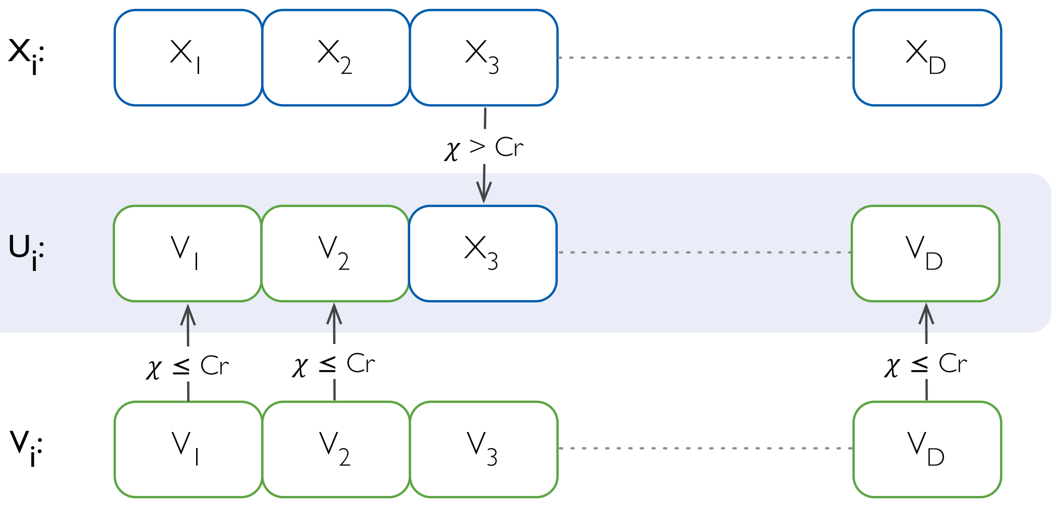


Figure 3.6: Binomial crossover. A trial vector is created as the combination of two candidate solutions. A uniform random number $\chi \in (0,1)$ is generated for each parameter in the trial vector. If $\chi \leq C_R$ then the parameter is copy from the vector $\mathbf{v}_i$, otherwise, is inherited from the target vector $\mathbf{x}_i$.

otherwise the parameter is copied from the target vector $\mathbf{x}_i$ (see Figure 3.6).

The second discrete recombination mechanisms used in DE is exponential crossover. Likewise binomial crossover, this mechanism combines two parent vectors into one child. This is, starting from a random position on the parameter vector, exponential crossover copies consecutive elements from the vector $\mathbf{v}_i$ to the trial vector $\mathbf{u}_i$ while the crossover probability condition is met ($rand(0,1) \leq C_R$). Once this condition fails the remaining of the parameters are inherited from the target vector $\mathbf{x}_i$. Hence, the probability of replacing an element in the sequence decreases exponentially with the increasing index j. Exponential crossover is illustrated in Figure 3.7.

The role of mutation in the DE algorithm is to explore the search space using the information encoded in the current population; whereas the role of the binomial or exponential crossover is to exploit the regions of the search space already explored. Hence, the crossover probability $C_R \in [0,1]$ determines the population diversity in successive generations by controlling the number of parameters inherited from the intermediary mutated population P_v^g. This means that small values of C_R will proportionally reduce the number of parameters inherited by the trial population P_u^g

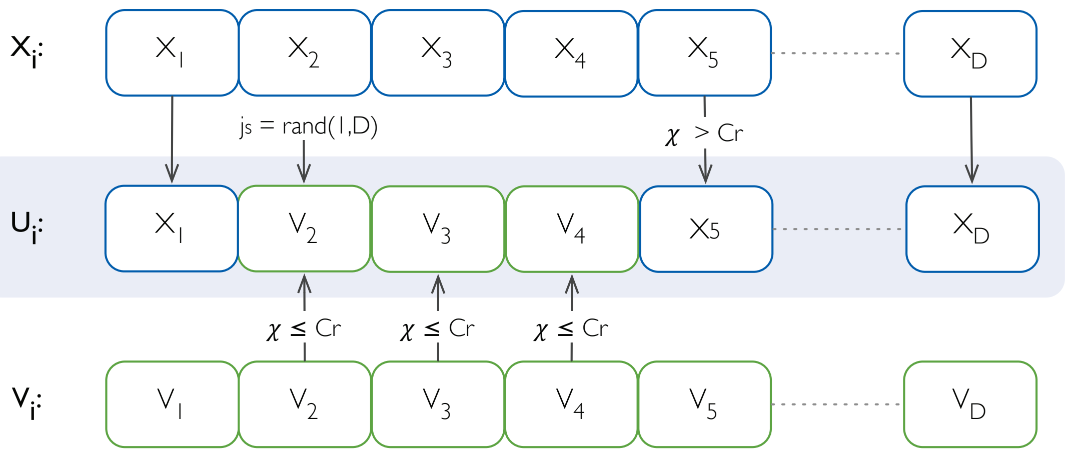


Figure 3.7: Exponential crossover. A trial vector $\mathbf{u}_i$ is built by combining consecutive elements of two candidate solutions. First, a starting crossover position $j_s = 2$ is randomly chosen from the interval $[1, D]$. Therefore, starting from j_s consecutive elements of $\mathbf{u}_i$ are copied from $\mathbf{v}_i$ while condition $\chi \leq C_R$ is met. Here, $\chi \in (0, 1)$ is a random number generated anew for each consecutive element $(j = j_s, j_s + 1, \ldots)$.

and values close to one will transform the algorithm into a mutation-only scheme. Moreover, a good tuning of C_R depends on the objective function to be optimized, e.g., separable objective functions admit values within $0.0 \leq C_R \leq 0.2$, whereas for non separable and multimodal objective functions values around $0.9 \leq C_R \leq 1$ are more suitable.

3.4.6 Selection

DE selection is a deterministic mechanism based on the direct comparison between the fitness values of candidate solutions, where fitness (or cost) is a quantitative measure of how close the solution is from the objective's optimum. It differs from other deterministic approaches, such as ES (μ, λ)- and $(\mu + \lambda)$-selection schemes, in that, it neither completely replaces the current population with the trial population nor ranks their elements to select the μ best candidate solutions. Instead, DE implements a binary tournament $(r = 2)$, known as one-to-one selection, since each vector in the current and trial population competes only once for survival.

In DE, current (parent) and trial (offspring) populations are always of the same

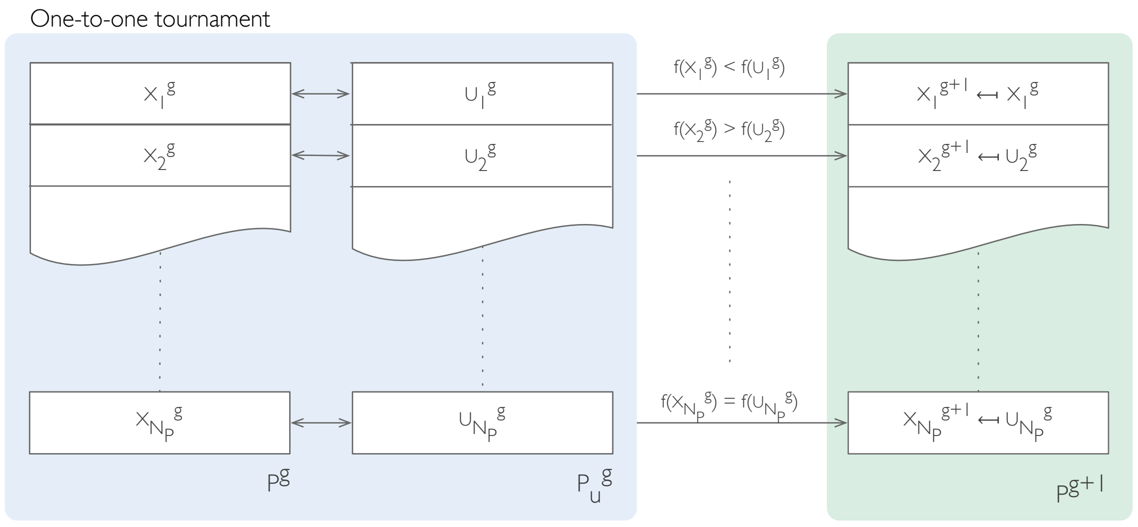


Figure 3.8: DE selection mechanism.

size, i.e., $\mu = \lambda = N_P$. Furthermore, being both indexed populations, each candidate solution $\mathbf{x}_i$ in P^g and the corresponding vector $\mathbf{u}_i$ in the trial population P_u^g are related by crossover, thus, act as natural opponents. The fact that opponents are not randomly chosen from the trial population is what mainly differentiates DE selection scheme from EP tournament.

Thus, selection is held by pairing one candidate solution from P^g with one from P_u^g and comparing their fitness values. Finally, the fittest one survives while the other is discarded. More formally, selection can be expressed as:

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g, & \text{if } f(\mathbf{u}_i^g) \leq f(\mathbf{x}_i^g) \\ \mathbf{x}_i^g, & \text{otherwise} \end{cases}, \forall i = 1, \ldots, N_P, \tag{3.7}$$

where f is the fitness function, usually represented by the objective function or a normalized version of the same (see Figure 3.8).

As stated in(3.7), DE selection is an *elitist* and *greedy* mechanism since it preserves the best-so-far solution in the current and trial populations and rejects any trial vector that does not improve the fitness value of its opponent in the current population. These two features increase the selective pressure on the trial population, which consequently decreases the overall convergence speed. Once the population P^{g+1} is generated, the iterative mutation-crossover-selection process continues until a suitable stop criterion is met.

3.4.7 Constraint Handling

Constraints are restrictions on the domain of values a parameter can take. In real-world systems these restrictions account for the internal and external factors that affect the feasibility of a system performance. Therefore, the solution not only has to be optimal but also must satisfy all constraints imposed on the design vector. In the optimization model, these restrictions can be expressed by equalities, inequalities, and boundary constraints.

For instance, in the inverse kinematics problem of a robot manipulator, internal constraints are imposed by the mechanical and proprioceptive systems of the robot (e.g., joint limits, maximum payload, etc.), whereas external constraints are due to those elements in the environment that modify the task execution (e.g., static or dynamic obstacles). Therefore, in a cluttered workspace an optimal solution to the inverse kinematics problem would be a joint configuration that provides the minimum end-effector pose error without exceeding the joint physical limits and not colliding with an obstacle. Assuming a free-of-obstacles environment, the search space of the inverse kinematics is only bounded by the lower and upper joint limits values. Joint limits define the robot manipulator workspace but also represent configurations of reduced dexterity and hence they should be avoided in the execution of the task. Consequently, we restrict our study to boundary constraints handling mechanisms.

In DE scheme, bounds constraints are only explicitly handled during initialization in which parameters are set with values randomly distributed within the feasible search space. DE mutation and crossover are unbounded mechanisms, however, mutation is the only mechanism in which out-of-bounds solutions are likely to appear. Therefore, additional mechanisms have to be considered to guarantee feasibility on the trial candidate solutions.

A popular approach to boundary constraints handling is to penalize the fitness value of offending candidate solutions and thus avoiding to promote them into the following generations. However, penalty methods may disrupt convergence when the optimum point lies near a boundary. An alternative to penalty methods is to reset with feasible values any parameter that exceeds a bound, by either setting the out-of-bound parameter with the value of the violated boundary (lower or upper) or by generating new values within the search space. Despite its simplicity the reset to bounds approach has shown to be detrimental for population diversity, eventually leading to stagnation and premature convergence. Therefore, in (Price et al., 2005) two more convenient resetting mechanisms are suggested, namely: bounce back and random reset.

In general terms, bounce back can be regarded as a complementary mechanism to mutation since it replaces offending parameters with random values that lie between the violated boundary and the mutation base vector ($\mathbf{x}_{r_1}$ in (3.5)). On the other hand, random reset is a much simpler mechanism since it re-samples the parameter over the entire feasible search space, as in (3.4). Intuitively, one can argue that randomly reinitializing out-of-bound parameters might actually decrease convergence speed, however, a preliminary study carried out by the authors of this work revealed that, for the inverse kinematics problem, random reset outperforms bounce back mechanisms in both robustness and convergence speed. DE algorithm is outlined in the pseudo-code listed in Algorithm 3.2.

3.5 Summary

The notion of optimality arises in a wide range of problems in economy, engineering, chemistry, design, and other scientific disciplines. Mathematically, this means finding a maximum or minimum value of a function of several variables under a set of constraints. The approach used to obtain the optimal solution would depend on the optimization model attributes and the application requirements, e.g., number of variables, search space domain, optimization criteria, number of optimal points, among others.

Linear search methods are largely used in unconstrained optimization problems thanks to their high convergence speed and sound theoretical background. These methods are based on the information about the topology of the search space retrieved from the first and second derivatives of the objective function. However,

Algorithm 3.2 Differential Evolution Algorithm

```
Input: $D$, $N_P$, $F$, $C_R$, $\mathbf{x}^{(lo)}$ and $\mathbf{x}^{(hi)}$.
 1: $g \leftarrow 0$
 2: INITIALIZE a population with $N_P$ random parameter vector $\mathbf{x}^g \in \left[\mathbf{x}^{(lo)}, \mathbf{x}^{(hi)}\right]$.
 3: EVALUATE fitness of each individual in population.
 4: while Termination condition $\neq$ true do
 5:     for $i = 1$ to $N_P$ do
 6:         SELECT random indices $r_1 \neq r_2 \neq r_3 \neq i \in \{1, \ldots, N_P\}$
 7:         for $j = 1$ to $D$ do                              ▷ MUTATION and CROSSOVER.
 8:             if rand$(0,1) < C_R$ then
 9:                 $u_{i,j} \leftarrow x_{r_1,j} + F\left(x_{r_2,j} - x_{r_3,j}\right)$
10:             else
11:                 $u_{i,j} \leftarrow x_{i,j}$
12:             end if
13:             if $u_{i,j}$ violates bound constraints then      ▷ CONSTRAINTS HANDLING
14:                 $u_{i,j} \leftarrow \mathbf{x}^{(lo)} +$ rand$\,(0,1)\left(\mathbf{x}^{(hi)} - \mathbf{x}^{(lo)}\right)$
15:             end if
16:         end for
17:         EVALUATE offspring fitness.
18:         if $\mathbf{u}_i$ is better than $\mathbf{x}_i^g$ then          ▷ ONE-TO-ONE TOURNAMENT
19:             $\mathbf{x}_i^{g+1} \leftarrow \mathbf{u}_i$
20:         else
21:             $\mathbf{x}_i^{g+1} \leftarrow \mathbf{x}_i^g$
22:         end if
23:     end for
24:     $g \leftarrow g + 1$
25: end while
Output: Best candidate solution in final population
```

since their search strategy rely on the gradient of the objective function they usually fail when the objective is non-smooth, noisy, or when it cannot be expressed in analytical form. In contrast, direct search methods do not require the gradient of the objective function to find the optimum. Instead, they are meta-heuristic algorithms based on a generate-and-test strategy to successive detect promising regions of the search space. Nonetheless, the consequence of a derivative-free approach is a noticeable decrease in the convergence speed rate.

A common feature of most of gradient-based and direct search methods is their tendency to converge towards local minima when the initial guess of the solution is not correctly chosen. To help to overcome this initialization problem, population-based or multi-start search methods have been proposed to explore the search space with a set of parallel local optimizers. A further improvement on the population-based approach is represented by the evolutionary computation paradigm, in which, the population of candidate solutions is not treated as a set of independent local optimizers but as a shared pool of information used to obtain better solutions in each iteration.

Evolutionary algorithms adopt the population-based approach and combines it with a biologically inspired search strategy. This strategy mimics the evolution of natural species to steer the population towards convergence relying on three basic mechanisms, namely recombination, mutation, and selection. Recombination and mutation reinforce exploration of the search space by generating new solutions from the information contained therein the population or by allowing random perturbation of its individuals; whereas selection promotes exploitation by building new populations out of the best solution found during the evolutionary process.

Differential evolution accommodates the evolutionary computation paradigm in a simple yet powerful meta-heuristic direct search algorithm. In this scheme, mutation explores the search space by adding weighted vector differences to a base vector. This non-probabilistic mutation mechanism admits different variants within a flexible arithmetic structure, such that, it can be modified to enhance exploration or exploitation aspects of the search strategy. Further, discrete recombination (crossover) increases population diversity by combining the current and new information contained therein the population members. Finally, mutation and crossover are synthesized with a greedy deterministic selection mechanism based on the direct comparison between the fitness values of the candidate solutions to steer the population towards the best regions of the search space.

A further study on DE search scheme shows that control parameters play an important role in the overall algorithm performance. Population size (N_P), mutation scale factor (F), and crossover rate (C_R) are directly related to the computational cost, convergence speed rate, and robustness of DE search strategy. Moreover, contrary to other real-valued evolutionary algorithms these parameters do not evolve with the population to adapt their values to the fitness landscape; but instead, N_P,

F, and C_R remain constants throughout the optimization process. Nonetheless, efficiently setting DE control parameters depends on the objective function and should aim at achieving the right balance between exploration and exploitation while reducing computational cost. A detailed study by Rönkkönen et al. (Ronkkonen et al., 2005) suggested that typical values for the population size range from $2D$ to $40D$, a good compromise between accuracy and speed is obtained with values of $F \in (0.4, 0.95)$, and C_R admits values within $(0, 0.2)$ for separable functions and around $C_R = 0.9$ when the objective function is multimodal and non-separable.

Chapter 4

Memetic Differential Evolution: Improving Convergence Speed

4.1 Background

The term *memetic algorithms* (MAs), also known as cultural algorithms, encompasses any evolutionary or population-based meta-heuristic search approach that embeds an individual learning or local refinement procedure. The name was inspired by Richard Dawkins' meme theory, which establishes an analogy between the genetic evolution of animals and the cultural evolution of humankind (Dawkins, 1976). The term *meme* was coined by Dawkins in his book *"The Selfish Gene"* to denote a unit of cultural information (trend, idea, or belief) that evolves from generation to generation by individual imitation and behavioural replication. The idea itself implies a local search performed by each individual to modify and improve a meme to be later transmitted to other individuals.

In a direct search algorithm this idea is equivalent to synthesize global search with local refinement of good solutions to speed up convergence and improve solution accuracy. Since an efficient search requires an effective trade-off between exploration and exploitation of the search space, many authors have echoed the idea of hybridizing the global search scheme of evolutionary algorithms (EAs) with independent local search techniques (LS) to help overcome the exploitation deficiencies observed in most EAs. Moreover, if one considers that most real-world optimization problems demand high convergence rates to satisfy real-time requirements and limited computing resources, then memetic algorithms could represent a suitable approach to reduce the number of functions evaluations required to converge to a global optimum.

In this chapter, we propose a memetic differential evolution scheme that combines the exploratory abilities of the standard DE with a simple LS heuristic for local refinement. First, we present a brief overview on the state-of-the-art of acceleration mechanisms for differential evolution.

4.2 Acceleration Mechanisms

Differential evolution (DE) simple search scheme has proved to be an effective optimization tool for several benchmark and real-world optimization problems. Likewise most evolutionary-based algorithms, DE's weakest feature is its low convergence speed. This has prompted the DE research community to investigate effective methods to reduce the overall number of functions evaluations without disrupting the global search performance.

DE acceleration mechanisms agree in promoting a well-balanced trade-off between exploration and exploitation to further assertive detection of promising regions of the search space, either by adaptively varying the control parameter values (Liu & Lampinen, 2005; Das et al., 2005; Brest et al., 2006; Zhang & Sanderson, 2007; Qin et al., 2009; Epitropakis et al., 2009; Neri & Tirronen, 2009), introducing new variation operators (Fan & Lampinen, 2003; Thangaraj et al., 2010; Epitropakis et al., 2011), or by hybrid memetic algorithms (MAs) (Rahnamayan et al., 2007, 2008; Noman & Iba, 2008; Neri & Mininno, 2010; Neri et al., 2011; Jia et al., 2011; Iacca et al., 2011).

4.2.1 Parameter Control

Among the most attractive features of DE as optimization method are its simple structure and few control parameters, namely: population size N_P, mutation scale factor F, and crossover rate C_R. Empirical results have shown that control parameter setting has a noticeable impact on the convergence performance of DE (Gämperle et al., 2002; Ronkkonen et al., 2005). For instance, large values of the mutation scale factor (F) and crossover rate (C_R), as suggested for multimodal problems, put a major emphasis on the exploration of different regions in the search space rather than the exploitation of the good detected ones. For such parameter setting, DE rapidly clusters the population around the basin of attraction of the global optimum, but from then on, the convergence speed rate to the exact optimal solution is low. Furthermore, given DE endogenous variation approach, setting large control parameters values increases the probability of stagnation (Lampinen & Zelinka, 2000).

Originally, these parameters were conceived as constant values set by the user, thus, several studies have been conducted to draw general guidelines on how to effectively tune N_P, F, and C_R. Yet, tuning DE parameters is not a straightforward task since it highly depends on the fitness function attributes (see Chapter 3 for further discussion). For this reason, recent efforts have been mainly devoted to find automatic mechanisms for parameter control, i.e., heuristic rules to adapt the parameters' values during the evolutionary process (Liu & Lampinen, 2005; Das et al., 2005; Zhang & Sanderson, 2007; Brest et al., 2006; Qin et al., 2009; Epitropakis et al., 2009; Neri & Tirronen, 2009).

One of the earliest adaptive strategy for parameter control was the Fuzzy Adaptive Differential Evolution (FADE) (Liu & Lampinen, 2005). In FADE, fuzzy logic controllers are used to adapt the values of the mutation scale factor (F) and the crossover rate (C_R) based on human knowledge and the feedback information extracted from the current and previous populations. Alternatively, Zaharie proposed a parameter control scheme guided by the evolution of the population diversity (Zaharie, 2003). In this approach, F and C_R were adjusted proportionally to the diversity change rate (measured as the population variance) to control the exploration/exploitation behaviour of the algorithm. In a different approach, F was linearly reduced with time to promote exploration of promising regions during early generations and thus progressively enhancing exploitation for fine tuning at later stages (Das et al., 2005).

A further step in parameter control is the so called "evolution of the evolution", which means that, the algorithm itself undergoes the evolutionary process through self-adaptation of its parameters (Brest et al., 2006; Zhang & Sanderson, 2007; Epitropakis et al., 2009; Neri & Tirronen, 2009). This approach is based on the idea that better parameter values lead to better solutions, which in turn, are more likely to propagate their genotype to future generations through selection. A simple self-adaptive parameter control scheme was proposed in (Brest et al., 2006), later denoted jDE. Here, self-adaptation was implemented by encoding the mutation scale factor and crossover rate value into each candidate solution, and thereby, allowing them to evolve with the solution by adjusting their values at random in each iteration. A similar self-adaptive approach, called JADE, was proposed to improve the robustness of the novel greedy DE search scheme, `DE/current-to-p-best` (Zhang & Sanderson, 2007). In this work, authors discussed that by introducing a self-adaptive parameter control, premature convergence of greedy search mechanisms can be overcome while still achieving high convergence rates.

More recently, Qin et al. (Qin et al., 2009) proposed a self-adaptation scheme for a set of mutation strategies and their parameter settings, called SaDE. The rationale behind this approach advocated that at different stages of the evolutionary process the search requirements may change, and therefore, some mutation strategies and parameter settings might be more effective than others. Accordingly, each of the mutation strategies in the set had a selection probability proportional to their success in generating promising solutions in previous generations. Consequently, the algorithm configuration was adapted in accordance to the knowledge accumulated from previous experiences. Other self-adaptive approaches include (Neri & Tirronen, 2009; Epitropakis et al., 2009). Experimental results have shown that self-adaptation significantly improves the efficiency and convergence speed rate of DE algorithms particularly on high-dimensional complex search spaces.

In the aforementioned parameter control schemes only the mutation scale factor F and the crossover rate C_R are considered liable to adaptive mechanisms, whereas the population size N_P is held as a constant value. However, this parameter is intrinsically related to DE convergence speed rate, and also to stagnation and premature convergence phenomena (Mallipeddi & Suganthan, 2008). Moreover, if one considers the evolutionary process as mainly exploratory during early generations and essentially exploitative at a later stage, in theory, the population size can be gradually decreased to reduce the number of function evaluations per iteration without drastically dropping diversity. In this regard, a population size reduction scheme was proposed in (Brest & Sepesy Maučec, 2008) to complement the self-adaptive jDE algorithm (Brest et al., 2006). Here, population size is dynamically halved after every number of iterations have elapsed to a total of $pmax$ times until a budget condition is met, e.g., maximum number of functions evaluations.

4.2.2 Variation Mechanisms

The arithmetic structure of DE mutation admits using vector differentials and base vectors as building blocks to design different variation operators according to the needs of the search. Hence, the performance of differential evolution can be further improved not only by automatically adapting DE parameters, but also by modifying the search strategy itself. This flexibility grants DE with mechanisms that permit to enhance exploration or exploitation depending on the number of parents involved in mutation and how these are chosen from the mating pool.

In their seminal work, Storn and Price already proposed a number of different mutation variants with up to two vector differentials terms for further exploration, or alternatively, locally guiding the search towards promising regions by replacing the base vector with the best solution in the current population. Similarly, Fan and Lampinen proposed a trigonometric mutation operator (TMO) to improve exploitation by locally biasing the perturbation towards the fittest out of three mutually different vectors (Fan & Lampinen, 2003). To avoid premature convergence by excessive local refinement, TMO is combined with standard mutation within a stochastic mutation framework, called TDE, controlled by an additional parameter, denoted by M_t. In (Thangaraj et al., 2010), based on the same basic structure of the original DE, five different mutation strategies were introduced using absolute differences and replacing the mutation scale factor by a random variable following a Laplace probability distribution.

In (Epitropakis et al., 2011), a proximity-based selection scheme was proposed to choose vectors involved in mutation by accounting the structure of the population as it evolves. Based on the clustering features shown by DE mutation strategies, a random selection was replaced by an affinity matrix that assigns a selection probability to each individual inversely proportional to its distance from the base vector. Experimental results on benchmark and real-world optimization problems showed that significant improvement on the function value and convergence speed can be obtained when compared to the original DE and other state-of-the-art algorithms.

4.2.3 Memetic Algorithms and Hybrid Approaches

Memetic algorithms (MAs) are EAs embedded with local search (LS) mechanisms. MAs hybrid approach aims at compensating for local exploitation deficiencies of EAs without compromising the overall global exploration performance. This is achieved by incorporating heuristics for local refinement of the most promising candidate solutions in the population, i.e., neighbourhood search around those solutions more likely to be close to an optimum. In classic DE, these exploitation deficiencies are due to a mainly exploratory framework driven by the mutation mechanism and high values of the crossover rate.

Several LS mechanisms have been explored to improve exploitation properties and whereby accelerate DE convergence. In (Noman & Iba, 2008), Noman and Iba proposed a memetic DE algorithm, called DEahcSPX, that implements an adaptive simplex crossover operator to improve the convergence speed of the standard `DE/rand/1/bin` algorithm. Experimental results show that by deterministically applying adaptive local search (AHCXLS) around the best candidate solution in the current population the overall convergence speed can be significantly improved. Other successful DE memetic approaches include DECLS for high-dimensional problems using chaotic local search and adaptive control parameters (Jia et al., 2011).

In addition to reducing the number of function evaluations needed to attain global convergence, further improvements on the DE economy have been proposed to deal with hardware limitations. In this regard, compact memetic algorithms arise as a class of estimation of distribution algorithms (EDAs) within a memetic search framework, in which, memory (hardware) restrictions are complied by storing and processing the statistical model of the population instead of the entire population and all the individuals therein. This, combined with a memetic search approach yields a light optimization algorithm with fast convergence speed in accordance to the requirements demanded by real-world applications.

With this two key aspects in mind, Neri and Minnino proposed a memetic compact DE algorithm (McDE) for the trajectory control of an industrial Cartesian robot using a single on-board micro-controller chip (Neri & Mininno, 2010). The proposed control system scheme was based on a sliding mode controller and a recurrent neural network (RNN) disturbance observer used to compensate for the steady-state errors introduced by the control law. Considering uncertainties on the payload mass, the neuron weights were adaptively updated so as to minimize the estimation error in each period of the trajectory. Therefore, a twofold DE optimization approach was suggested to cope with the hardware deficiencies, measurements noise, and real-time requirements of the control loop. First, the population was represented as a Probability Vector (PV) consisting of a matrix with mean and standard deviation values describing a Gaussian Probability Distribution Function for each design variable (neuron weight), thus, circumventing memory limitations. Second, a memetic search approach based on the integration of the exploratory framework `DE/rand/1/bin` and a stochastic local search mechanism was used to update the population distribution. Simulation results showed that McDE outperforms other state-of-the-art compact algorithms for this particular application. More recently, an alternative DE memetic approach for limited memory problems was proposed by Neri et al., and denoted by DEcDE (Neri et al., 2011). In contrast to McDE, this approach did not combine DE exploratory framework with additional LS mechanisms to improve the convergence speed. Instead, it used a multiple exploitative framework with a randomized perturbation scheme to promote global exploration.

An intuitive yet almost unexplored acceleration approach is to enhance the initialization mechanisms used to sample the search space, provided that, the fitter the initial population, the more likely would it be to find the global optimum after fewer iterations. If a priori information about the optimal location is available, this can be used as seed to guide the initialization process; however, this is seldom the case for most optimization problems. Consequently, a uniform random sampling of the feasible search space is commonly suggested. However, although random initialization provides with the necessary population diversity to start the search, the chance of getting closer to the solution can be further enhance by introducing some heuristics during the initialization process. In (Rahnamayan et al., 2007), a novel opposition-based learning initialization method was embedded into the canonical DE with the aim of obtaining fitter individuals in the initial population even when a priori information about the optimal was not available. The authors used two population sets, one containing the randomly generated points and the other containing the points opposite to that of the initial points. Finally the two populations were merged and the best points were taken to form the initial population. Numerical experiments revealed that for low-dimensional problems opposition-based learning initialization outperforms standard random initialization in convergence speed. Opposition-based learning has also been used to enhance exploitative pressure of the standard and compact DE algorithms (Rahnamayan et al., 2008; Iacca et al., 2011).

4.3 A Migration-based Memetic Differential Evolution (δDE)

The evolution of a population of a species halts when the genetic variation mechanisms no longer introduce significant changes among the parents and the successive offspring generations. At low genetic diversity, variation and selection reach a balance that leads to stagnation. Nevertheless, if the population is not isolated from other populations of the same species, a third mechanism might take place to stimulate evolution: migration.

Migration not only implies the movement of individuals from one region to another, but refers to gene flow, i.e., the introduction of genetic material by interbreeding. Thus, migration (or gene flow) tends to increase diversity in the population by adding new traits into the genetic pool.

Based on the same principle, migration has been widely used in genetic algorithms to avoid stagnation, premature convergence, and to improve convergence speed. In GAs, migration is either used as a random re-sampling mechanism to replace poor solutions in each generation (Moed et al., 1990), or as a restarting procedure of small populations after convergence (Goldberg, 1989). Both migration strategies are diversification mechanisms used to introduce new information into the population.

Migration has also been used in DE to improve convergence. Recently, in (Gong et al., 2010), the authors proposed a DE/BBO hybrid algorithm that combines canonical DE with biogeography-based optimization (BBO). BBO is a population-based stochastic optimizer based on the models of biological migration among habitats (Simon, 2008). In this scheme, each candidate solution represents a habitat with an associated habitat suitability index (HSI), immigration (λ) and emigration rate (μ). BBO adopts a migration operator to share information between solutions based on their migration rates (λ and μ), and a mutation operator for random perturbation of the population. Therefore, poor solutions (low HSI, low λ, and high μ) tend to admit useful information from good solutions (high HSI, high λ, and low μ). In this case, migration operates at parameter level within a single population of habitats, in contrast to the individual-level migration between sub-populations of GAs.

In summary, migration in EAs acts as a replacement mechanism aimed at improving poor solutions in a population. This replacement strategy can be tailored to promote either exploration or exploitation according to the deficiencies in the search algorithm. For instance, in crossover-based EAs exploration is usually promoted by randomly sampling the search space; whereas in mutation-based EAs migration often enhances exploitation by using the fittest individuals within the replacement strategy. Based on these notions, we propose a migration mechanism to improve the convergence speed of standard DE. We adopt the standard `DE/rand/1/bin` scheme because of its simple mutation structure yet powerful exploration performance.

Thus, let us consider a population of N_P D-dimensional candidate solutions that evolves by means of DE iterative process of mutation, crossover, and selection. The variation mechanisms of mutation and crossover explore the search space and lead the population towards its most promising regions, whereas the selection mechanism gradually clusters the candidate solutions around the global minimizers of the objective function. To speed up this process with a migration policy, a first strategy would be to directly replace a small percentage of the worst candidate solutions (low fitness) with copies of better solutions (high fitness) in each iteration. This elitist approach increases the velocity in which the candidate solutions group around promising regions, but also decreases the diversity of the population which might lead to stagnation or premature convergence. A more general approach would be to assume that each candidate solution in the population represents the mean vector of a D-variate normal distributed sub-population with covariance matrix Σ. Therefore, poor solutions could be replaced by randomly sampling different sub-population models to produce solutions in the vicinity of better existing ones; such that, the length of the local search is controlled by Σ. The latter is the operating principle of the so called *discarding* mechanism.

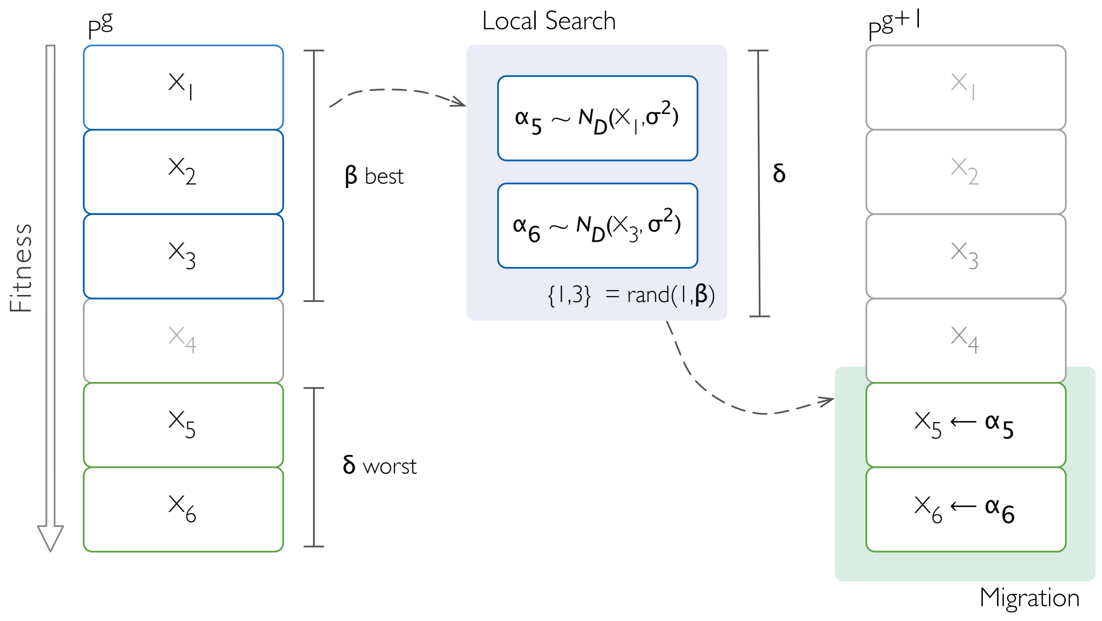
In other words, discarding combines migration and a local search strategy into a single operator. According to the rationale that supports memetic algorithms, embedding an independent local search mechanism like discarding within the evolutionary search approach of standard DE yields a memetic differential evolution hereafter denoted by δDE. In this sense, δDE strategy is closer to the memetic approach of the DEahcSPX algorithm (Noman & Iba, 2008). The proposed memetic differential evolution scheme is outlined in the pseudo code listed in Algorithm 4.1.

Algorithm 4.1 Memetic Differential Evolution (δDE)

```
Input: $D$, $N_P$, $F$, $C_R$, $\delta$, $\beta$, $\Sigma$, $\mathbf{x}^{(lo)}$ and $\mathbf{x}^{(hi)}$.
 1: $g \leftarrow 0$
 2: INITIALIZE a population with $N_P$ random candidate solutions $\mathbf{x}^g \in \left[\mathbf{x}^{(lo)}, \mathbf{x}^{(hi)}\right]$.
 3: EVALUATE fitness of each candidate solution in population.
 4: while Termination condition $\neq$ true do
 5:     for each candidate solution in current population do
 6:         MUTATION DE/rand/1.
 7:         BINOMIAL CROSSOVER.
 8:         CONSTRAINT HANDLING of out-of-bounds parameters.
 9:         EVALUATE offspring fitness.
10:         ONE-TO-ONE TOURNAMENT selection.
11:     end for
12:     DISCARDING of worst candidate solutions in current population.
13:     $g \leftarrow g + 1$
14: end while
Output: Best candidate solution in final population
```

4.3.1 Discarding Mechanism

Discarding combines migration and local search in a unified mechanism. This is, on one part discarding acts as a migration operator since in each generation a number of the worst fitted candidate solutions are replaced by new ones. And on the other hand, it uses a simple local search heuristic to perturb the best performing elements in the population to generate new ones. By combining these two mechanisms we overcome exploitation deficiencies without deteriorating the exploration performance of mutation, and thus, increasing convergence speed.



P^g
Fitness
x_1
x_2
x_3
x_4
x_5
x_6
β best
δ worst
Local Search
$\alpha_5 \sim N_D(x_1, \sigma^2)$
$\alpha_6 \sim N_D(x_3, \sigma^2)$
$\{1,3\} = \text{rand}(1,\beta)$
δ
P^{g+1}
$x_5 \leftarrow \alpha_5$
$x_6 \leftarrow \alpha_6$
Migration

Figure 4.1: Scheme of the discarding mechanism. The δ worst elements are replaced by adding a random noise to δ candidate solutions randomly selected from the β best in the current population.

Algorithm 4.2 Discarding Mechanism

```
Input: SORT $P^g = \{\mathbf{x}_1, \ldots, \mathbf{x}_\imath, \ldots, \mathbf{q}_{N_P}\}$, such that, $f(\mathbf{x}_\imath) \leq f(\mathbf{x}_{\imath+1})$
1: for $\imath = N_P - \delta + 1$ to $N_P$ do
2:     SELECT randomly an index $b$ from $\{1, \ldots, \beta\}$
3:     $\mathbf{x}_\imath \leftarrow \mathcal{N}(\mathbf{x}_b, \sigma^2)$
4: end for
```

Consider a population P^g of N_P D-dimensional candidate solutions, where g is the current generation. Let us also assume that each of them represents the mean vector of a D-variate normal distribution with covariance matrix Σ. Now, consider two sub-populations from P^g, one containing the β best performing solutions (B^g), and another with the δ worst candidate solutions (W^g), such that $\delta + \beta \leq N_P$.

According to discarding, in each generation every candidate solution in W^g is replaced with candidate solutions randomly sampled from B^g. However, instead of simply duplicating good solutions from B^g, discarding performs local refinement of the best fitted individuals in the current population. Therefore, for each candidate solution in W^g one is randomly (uniformly) selected from B^g, here denoted by $\mathbf{x}_b$ ($b = 1, \ldots, \beta$). Then, a new candidate solution is produced by sampling a D-variate normal distribution with mean $\mathbf{x}_b$ and covariance matrix Σ. This local search approach is based on a well-known local improvement process technique (LIP), called Solis-Wets method or random optimization (Solis & Wets, 1981). To get a rotationally invariant operator the variance is set to a scalar constant value σ^2 equal for all parameters and all candidate solutions in P^g. Discarding is schematically illustrated in Figure 4.1, and the pseudo-code is presented in Algorithm 4.2.

Discarding Parameters

Discarding replacement strategy aims at reinforcing exploitation by in each generation steering the population towards the current best solutions. However excessive exploitation also needs to be avoided specially for small-sized populations, otherwise, the search could prematurely converge to a local optimum. Further, the role of δ and σ is to regulate the amount and length of local search performed by discarding, and consequently, both are directly related to the good convergence behaviour of the acceleration mechanism.

Parameter σ determines the local distribution of the offspring around their parents, and thus controls the local search length. A small value of σ produces candidate solutions densely distributed around their parents which might fail at significantly improving the search quality. But, on the other hand, large values of σ yield sparsely distributed offspring which might consume unnecessary additional fitness evaluations.

The discarding size δ controls the migration rate in each generation, this is, it represents the number of individuals that are discarded in each generation to be replaced with better solutions. For convenience, this value is usually defined as a percentage of the total population size, thus,

$$\delta = \frac{N_D}{N_P} \times 100 \tag{4.1}$$

with N_D is the size of a sub-population containing the worst candidate solutions in P^g. If Lamarckian learning is assumed, one can easily deduced that only after N_P/N_D generations discarding replaces an entire population by local search alone.

Therefore, it is clear that population diversity would rapidly decrease with large values of δ which might lead to stagnation and premature convergence. Additionally, the population size (N_P) has to be large enough to ensure diversity but not too large to increase the computational cost.

Empirical results also suggest that the size of the elite pool (β) should be sufficiently large to guarantee diversity, i.e., $\beta \ggg \delta$, with $\beta = N_P/2$ being a good initial value. Additionally, the population size (N_P) has to be sufficiently large to assure diversity but not too large to increase the computational cost.

Similarities and Differences with BBO

Since discarding introduces a migration mechanism to enhance DE intensification features, we can draw an analogy between our approach and the one of DE/BBO (Gong et al., 2010). In this analogy, the HSI value would be represented by the fitness f of each candidate solution, such that, in each generation the likelihood of being replaced increases as the cost value does. However, in contrast to λ and μ, the discarding migration rate δ is a constant value that does not depend on the fitness value. In addition, the replacement strategy of discarding implements a local refinement process of the best performing candidate solutions to completely replace poor ones, instead of inheriting parameters from the elite solutions as in DE/BBO. In these aspects, the discarding approach is closer to the approach of memetic algorithms earlier described.

4.3.2 Example I: Goldstein-Price's function

Before further analysing δDE convergence performance, let us first illustrate its operating mechanism with some examples. First, consider the Goldstein-Price's function, shown in Figure 4.2, as the objective function in our minimization problem, defined as

$$\begin{aligned} f_1(\mathbf{x}) = &\left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3_2^2\right)\right] \cdot \\ &\left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right], \end{aligned} \tag{4.2}$$

where $\mathbf{x} \in \mathbb{R}^2$ is bounded by $-2 \leq \mathbf{x} \leq 2$. The optimization model thus defined has one global optimal solution located at $\mathbf{x}^* = (0, -1)$, that yields a minimum

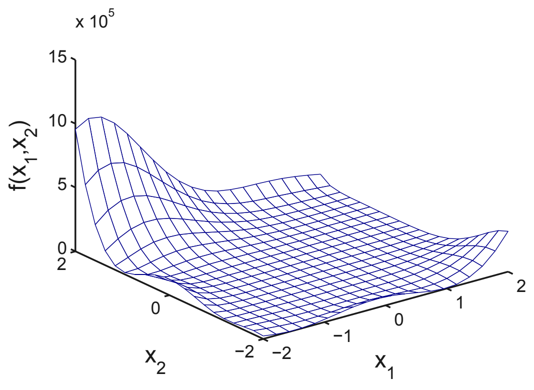


Figure 4.2: Goldstein-Price's function.

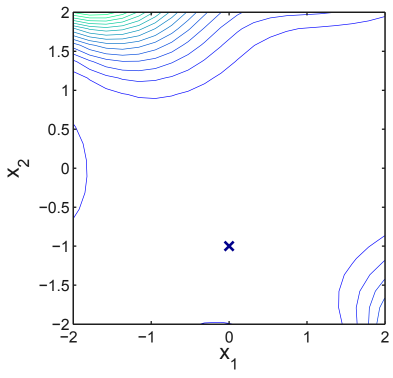
Table 4.1: δDE control parameters values for the Goldstein-Price's, six-hump camel back, and Rastrigin's functions.

	N_P	F	C_R	σ	δ	β
f_1				0.01	2 %	2 %
f_2	50	0.5	0.8	0.1	10 %	20 %
f_3				0.01	10 %	20 %

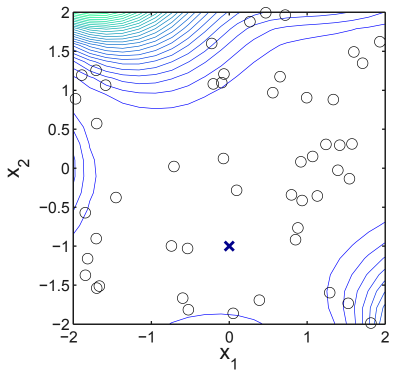
value of $f(\mathbf{x}^*) = 3$. For illustration purposes only, δDE control parameters were set to the values in Table 4.1. This means that in each generation the worst candidate solution will be replaced by a solution normally distributed around the best solution with variance of 1×10^{-4}. Figure 4.3 shows the first iteration of δDE algorithm. After initialization, the population undergoes mutation, crossover, and selection to obtain a trial population P_u. Then, replaces the worst candidate solution with one sampled from the neighbourhood of the best solution in the trial population. Finally, the modified trial population is promoted to the next generation $P^{g=2}$.

4.3.3 Example II: Six-hump camel back function

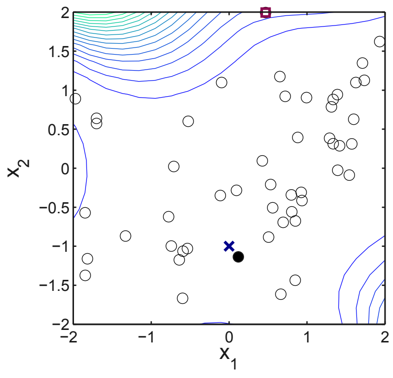
As second example, consider the multimodal benchmark function shown in Figure 4.4, also known as six-hump camel back function. This is a two-dimensional



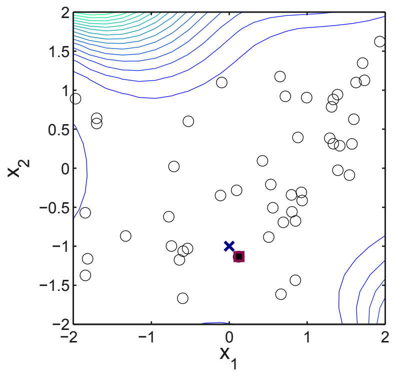
(a) Optimal solution $\mathbf{x}^*$.



(b) Initial population $P^{g=0}$.



(c) Before discarding $P_u^{g=1}$.



(d) After discarding $P^{g=2}$.

Figure 4.3: Phase portraits of the first iteration of δDE for the Goldstein-Price's function. (a) Contour plot and optimal solution. (b) Initial population. (c) Trial population before discarding, best (●) and discarded (□) candidate solutions. (d) Current population after discarding, best (●) and migrant (□) candidate solutions.

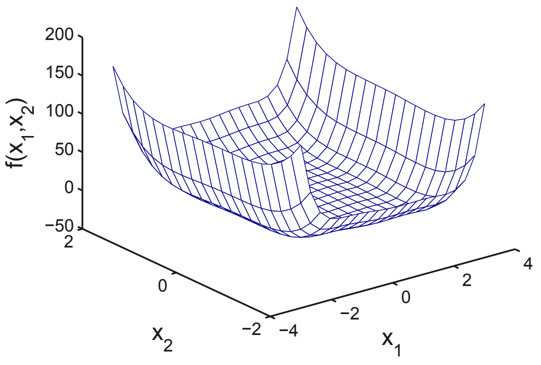


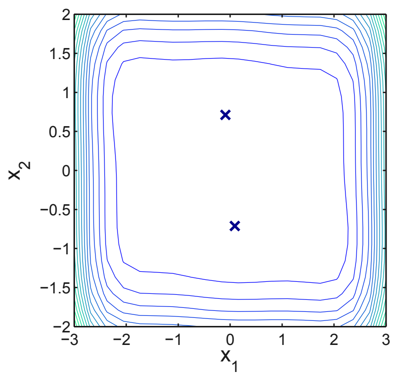
Figure 4.4: Six-hump camel back function.

objective function defined as

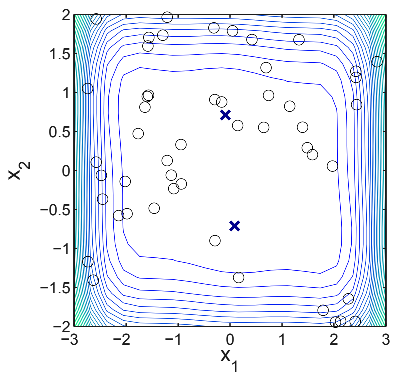
$$f_2(\mathbf{x}) = \left(4 - 2.1x_1^2 + \frac{1}{3}x_1^4\right) x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right) x_2^2, \tag{4.3}$$

where $\mathbf{x} \in \mathbb{R}^2$ is usually defined within the rectangle $-3 \leq x_1 \leq 3$ and $-2 \leq x_2 \leq 2$. Within this area (4.3) has six local minima, two of them are global ones located at $\mathbf{x}^* = (-0.0898, 0.7126)$ and $\mathbf{x}^* = (0.0898, -0.7126)$ that yield a minimum value of $f(\mathbf{x}^*) = -1.0316$.

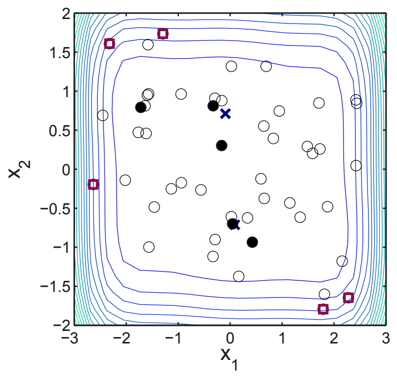
In this case, the problem's landscape demands a more efficient search strategy to avoid local minima while rapidly converging to one of the global optimal solutions. Therefore, for the same values of N_P, F and C_R we intuitively increased the discarding parameters to further enhanced local search around good candidate solutions. Thus, in each generation 10 % of the poorer solution in the population are replaced by sampling the vicinity of random solutions selected from the 20 % fittest candidate solutions. Figure 4.5 portraits the population before and after discarding during the first generation of δDE. Further, a comparison between the convergence behaviour of standard DE and δDE shows that the discarding simple strategy can significantly improve the convergence speed without failing convergence to the global minima (see Figure 4.6).



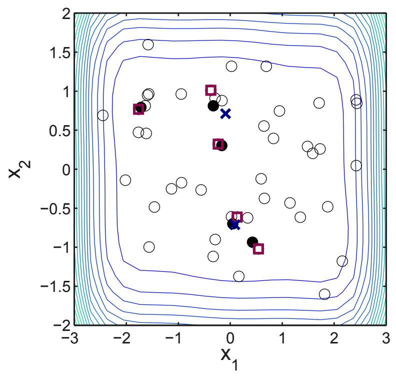
(a) Optimal solutions $\mathbf{x}^*$.



(b) Initial Population $P^{g=0}$.

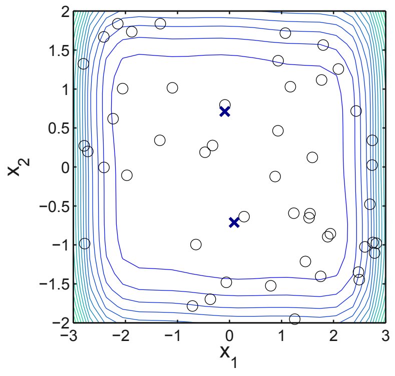


(c) Before discarding $P_u^{g=1}$.

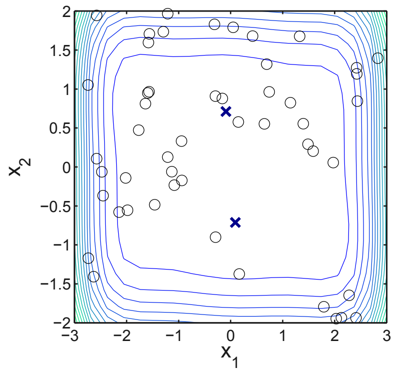


(d) After discarding $P^{g=2}$.

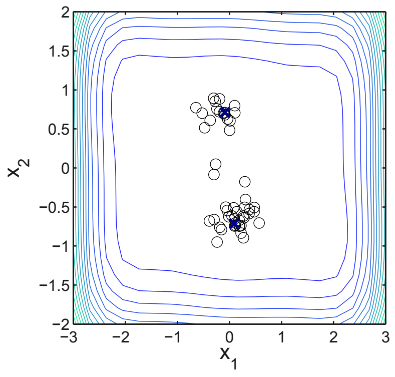
Figure 4.5: Phase portraits of the first iteration of δDE for the six-hump camel back function. (a) Contour plot and optimal solutions location. (b) Initial population. (c) Trial population before discarding, best (●) and discarded (□) candidate solutions. (d) Current population after discarding, best (●) and migrant (□) candidate solutions.



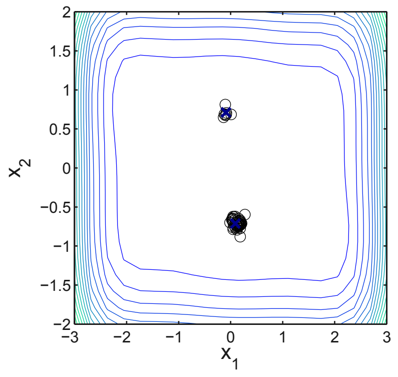
(a) DE initial population.



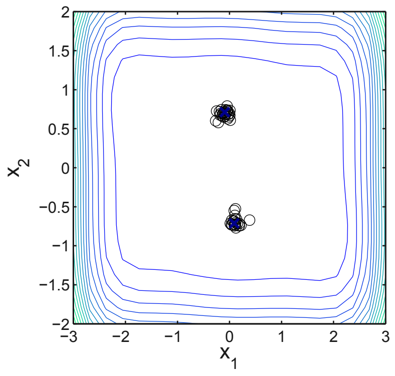
(b) δDE initial population.



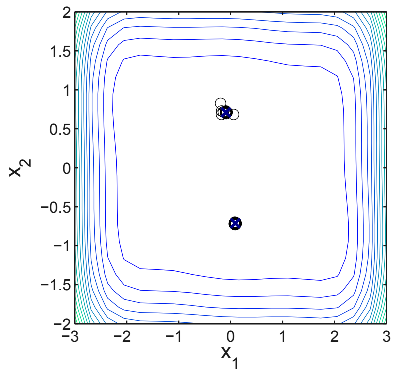
(c) DE after ten generations.



(d) δDE after ten generations.



(e) DE after twenty generations.



(f) δDE after twenty generations.

Figure 4.6: Comparison of the evolutionary behaviour of DE (left) and δDE (right) for the six-hump camel back function.

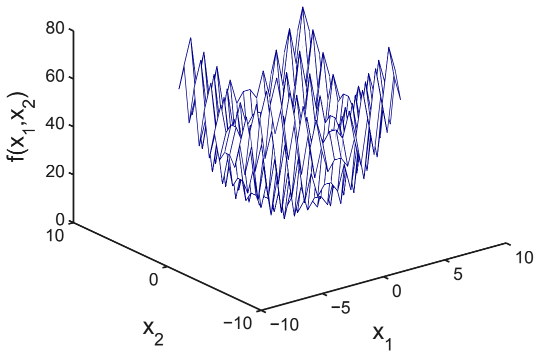


Figure 4.7: Two-dimensional Rastrigin's function.

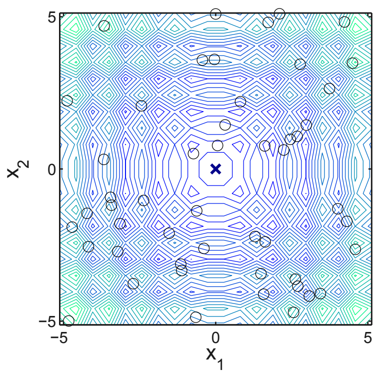
4.3.4 Example III: Rastrigin's function

Finally, let us consider a third benchmark function called the Rastrigin's function. As shown in Figure 4.7 for the two-dimensional case, this function is highly multimodal with several local minima regularly distributed within the search space. For $\mathbf{x} \in \mathbb{R}^2$ this function is defined as

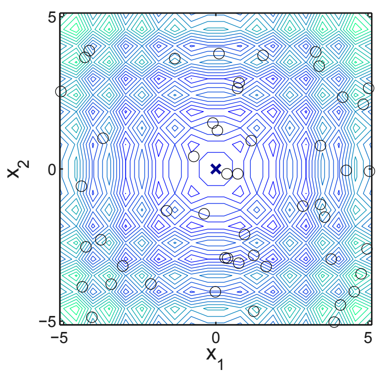
$$f_3(\mathbf{x}) = 20 + \left(x_1^2 - 10\cos\left(2\pi x_1\right)\right) + \left(x_2^2 - 10\cos\left(2\pi x_2\right)\right) \tag{4.4}$$

with the search space usually constrained to $-5.12 \leq \mathbf{x} \leq 5.12$. This function yields its minimum global value of zero at $\mathbf{x}^* = (0, 0)$.

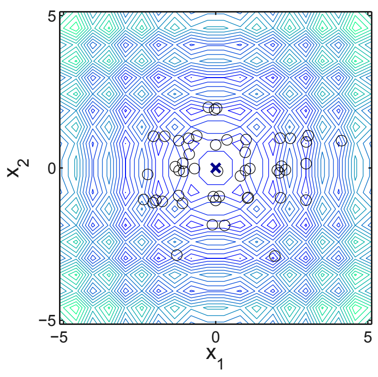
Once again, we kept the same values of N_P, F, and C_R to show how discarding influences DE search performance. The high multimodality of this function requires a careful balance between exploring the search space and reinforcing good solutions by local search. Thus, σ, δ, and β were set to relatively small values to avoid deteriorating diversification while promoting intensification on the most elitist subpopulation. Control parameter values are summarized in Table 4.1. δDE resulting convergence behaviour is shown in Figure 4.8.



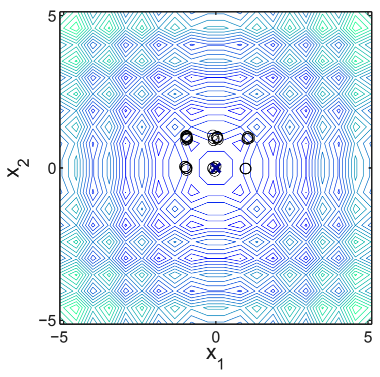
(a) DE initial population.



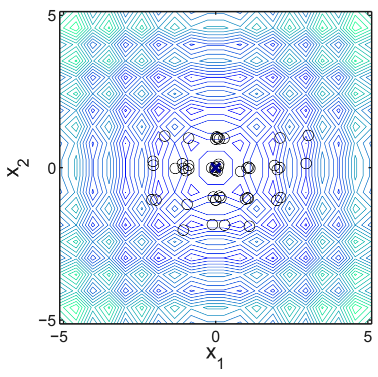
(b) δDE initial population.



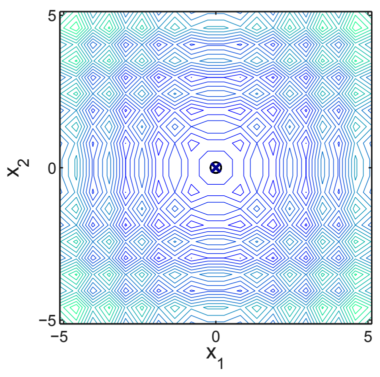
(c) DE after ten generations.



(d) δDE after ten generations.



(e) DE after twenty generations.



(f) δDE after twenty generations.

Figure 4.8: Comparison of the evolutionary behaviour of DE (left) and δDE (right) for the Rastrigin's function.

4.4 Summary

In this chapter we have introduced a memetic differential evolution scheme denoted by δDE. A memetic algorithm is an evolutionary algorithm embedded with a local search mechanism. Similarly to any other acceleration mechanisms, such as, parameter control and variation mechanisms, MAs strategy seeks to balance the exploration and exploitation features of the search to speed up convergence.

Therefore, our memetic approach aims at balancing the outstanding exploratory behaviour of the standard DE scheme with an intensification operator called *discarding*. Discarding combines *migration* with a *local search* strategy to enhance intensification of good solutions without decreasing robustness and accuracy of the standard DE. This is, on the one hand migration introduces new information into the population by replacing poor performing solutions in each iteration; and on the other, local search provides with a replacement strategy to enhance exploitation in the neighbourhood of the fittest solutions in the population. Synthesizing both mechanisms yields in a balanced exploration/exploitation behaviour.

Moreover, discarding introduces three new parameters, namely: discarding rate (δ), elite pool size (β), and local search length (σ). Adding these parameters to the standard DE increases its complexity, and their interaction with the population size (N_P), mutation scale factor (F), and crossover rate (C_R) has yet to be formally studied. However, using three benchmark optimization functions we have shown how much potential discarding has to significantly improve the convergence speed of standard DE.

Chapter 5

δDE-based Inverse Kinematics: Problem Formulation

The inverse kinematics represents one the fundamental problems in the kinematics control of robot manipulators. It formulates the question of how to determine the joint configuration that places the end-effector at a desired pose within the robot workspace. Although closed-form analytical solutions might be available for some open kinematics chains, numerical methods are often used to iteratively solve the inverse kinematics problem of more complex kinematics structures. These methods can be classified into (i) Jacobian-based methods and (ii) optimization-based methods. Their advantages and disadvantages were already discussed in Chapter 2. In this chapter, we further investigate alternative optimization-based solutions to the inverse kinematics problem, with the aim to overcome the performance deficiencies of Jacobian and other optimization-based kinematics inversion approaches.

5.1 Optimization Model: Inverse Kinematics

Consider a serial robot manipulator with N degrees of freedom. At any arbitrary joint configuration ($\mathbf{q}$) the end-effector's position and orientation can be fully described by the pose coordinates $\xi = (\mathbf{p}, \mathbf{R})$, where the position $\mathbf{p} \in \mathbb{R}^n$ and orientation $\mathbf{R} \in SO(n)$ are defined relative to an inertial reference frame fixed at the base of the robot.

Given a target pose ξ_d, consider the problem of finding at least one feasible joint configuration vector that locates the end-effector at the desired pose coordinates. Without loss of generality, it is also assumed that it is not possible to find an analytical closed-form solution to the inverse kinematics problem, and therefore, a numerical approach is required.

Under these premises let us propose a numerical optimization-based approach to solve the inverse kinematics problem of robot manipulators. To this end we first define a metric E to express the error between ξ_d and any arbitrary pose ξ within the robot workspace. This function is defined such that it vanishes when ξ_d and ξ are equal. Thereby, any joint vector $\mathbf{q}^*$ whose corresponding pose coordinates ξ^* minimize E could be considered the solution to the inverse kinematics problem.

The above formulation defines two different search spaces for the optimization problem (i) the error space defined in terms of the pose coordinates in the task space, and (ii) the configuration space defined as the set of feasible joint configuration vectors of the robot. In this context, and evoking the terminology introduced for evolutionary algorithms, the latter defines the representation space whereas the former defines the solution space. Thus that, the search mechanisms are activated on the representation space, but the proximity to the optimum is evaluated on the solution space where the error is defined. Therefore, it becomes necessary to define a mapping function to link both spaces, such that, any feasible joint vector can be transform onto its corresponding pose coordinates. For robot manipulators that would be the forward kinematics equation (fk), that in contrast to the original inverse kinematics problem can be analytically obtained in closed form for most robot manipulators (Denavit & Hartenberg, 1955). As a result, the inverse kinematics problem is transformed into an alternative mapping problem via the forward kinematics of the robot.

Further, the error metric should measure the discrepancies of the end-effector pose relative to a target in the robot's task space, but also should be defined to meet different requirements of the task. For instance, for a certain task it could only be necessary to satisfy a target position, whereas another could exclusively demand a desired orientation of the end-effector. Therefore, let us define a pose error function as the weighted sum of the position error ($E_P \in \mathbb{R}$) and orientation error ($E_O \in \mathbb{R}$), written as

$$E = w_p E_P + w_o E_O, \tag{5.1}$$

where the weighting factors w_p and w_o are constant normalization values defined as

$$w_p = 1$$
$$w_o = \frac{\sqrt{\lambda_{KC} \cdot \|\mathbf{p}_d\|}}{\pi}$$

with λ_{KC} a coefficient that depends on the dimensions of the robot (Tabandeh et al., 2010). Furthermore, by setting to zero any of the weighting factors the error function can be simplified to comply with particular task's requirements.

Finally, the optimization model for the inverse kinematics problem is obtained as

$$\min_{\mathbf{q}\in\mathcal{Q}} E \quad \text{subject to:} \quad \begin{aligned} &\xi = fk(\mathbf{q}) \\ &\mathcal{Q} = \{\mathbf{q} \mid G(\mathbf{q}) \geq 0 \wedge H(\mathbf{q}) \geq 0\} \end{aligned} \tag{5.2}$$

where G and H represent the physical constraints imposed by the mechanical structure of the robot or the environment. Assuming a free-of-obstacle environment, the joint upper and lower bounds are set as constraints. Given that, the joint limits define the robot manipulator workspace and more importantly represent configuration values of reduced dexterity that should be avoided during the execution of a task, thus

$$G(\mathbf{q}) = \mathbf{q} - \mathbf{q}^{(lo)} \tag{5.3}$$
$$H(\mathbf{q}) = \mathbf{q}^{(hi)} - \mathbf{q} \tag{5.4}$$

with $\mathbf{q}^{(lo)}$ and $\mathbf{q}^{(hi)}$ the lower and upper joint limits, respectively.

In comparison to the Jacobian-based methods, the advantages of formulating the inverse kinematics problem as an optimization model (5.2) are the following:

- The solution is directly obtained in terms of joint configuration variables and not velocities.

- The singularities issues associated to the Jacobian matrix are avoided.

- The objective function an be easily modified to meet different task requirements.

5.1.1 Position Error

Robot accuracy measures the ability of the robot to position its wrist end at a desired target point within the workspace, and it is usually defined in terms of the spatial resolution of the control system. Therefore, the lower is the distance between the final position of the end-effector and the target position, the higher is the accuracy of the robot.

Similarly, in the inverse kinematics problem the position error can be defined as the distance from the desired location of the end-effector ($\mathbf{p}_d$) to any point within the robot's workspace ($\mathbf{p}$), as depicted in Figure 5.1. This distance can be computed as the length of the residual error vector $\mathbf{p}_e$, as

$$E_P = \|\mathbf{p}_e\| = \|\mathbf{p}_d - \mathbf{p}\|, \tag{5.5}$$

where $\|\cdot\|$ denotes a norm function on $\mathbb{R}^n$.

The norm function specifies a scalar metric over the elements of a vector space, and thus, admits different definitions. The most common norm on $\mathbb{R}^n$ is the *Euclidean* or ℓ_2-norm, defined for the residual error vector as

$$\|\mathbf{p}_e\|_2 = \sqrt{\sum_{i=1}^{n} |p_{e_i}|^2}. \tag{5.6}$$

Another useful norm on $\mathbb{R}^n$ is the *Manhattan* or ℓ_1-norm, defined as

$$\|\mathbf{p}_e\|_1 = \sum_{i=1}^{n} |p_{e_i}|. \tag{5.7}$$

The ℓ_1-norm and ℓ_2-norm have been widely used as loss functions in the minimization of the residual error between measured and estimated data in parameter estimation problems. In a recent study by Moreno et al., the ℓ_1-norm and ℓ_2-norm were tested and compared as loss functions in an optimization-based filter for global localization of mobile robots (Moreno et al., 2011). Experimental results revealed

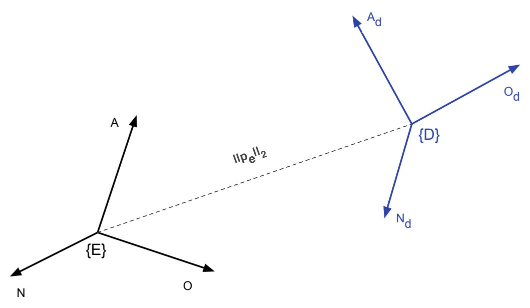


Figure 5.1: Position error (E_P).

that the ℓ_1-norm showed a superior performance in robustness and accuracy on the estimation of the robot's pose under different levels of noise contamination in the sensor data. Nonetheless, the ℓ_2-norm exhibited better convergence speed and success ratio for relatively high values of input noise variance.

In a real robot manipulator the uncertainties on the inverse kinematics optimization problem (5.2) are derived from the forward kinematics model due to mechanical inaccuracies induced by the physical properties of the structure. Assuming small uncertainties on the kinematics model, in this work the ℓ_2-norm is adopted as metric for the position error since it provides faster convergence than the ℓ_1-norm.

Definition 1 (Position Error). *Let* $\mathbf{p}_d$ *and* $\mathbf{p}$ *be the desired end-effector position and an arbitrary point in the Cartesian space, respectively. The position error is defined using the* ℓ^2*-norm as*

$$E_P = \|\mathbf{p}_d - \mathbf{p}\|_2 \tag{5.8}$$

5.1.2 Orientation Error

The spatial orientation of a rigid body admits multiple analogous representations being the Euler angles the most commonly used. Nonetheless, Euler angles provide an ambiguous metric for the orientation error since they suffer from representation singularities. Therefore, following a similar approach to that proposed in (Tabandeh et al., 2010) we adopt the unit quaternions representation to define the orientation error given their compact and numerically stable representation.

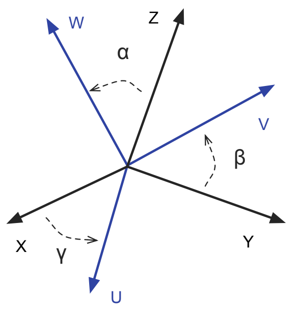


Figure 5.2: Orientation error between two frames with coicident origin ($\mathbf{R}_e$).

Hence, let OXYZ and OUVW denote two different frame axes with coincident origin, but with OUVW rotated respect to OXYZ. The rotation necessary to align frame OUVW can be regarded as a measure of the orientation error between both frames, as illustrated in Figure 5.2.

Definition 2 (Error Rotation Matrix)**.** *Consider a rotation matrix* $\mathbf{R}$ *that describes an arbitrary orientation of the end-effector in the robot's workspace. Given a target orientation* $\mathbf{R}_d$*, the orientation error of* $\mathbf{R}$ *relative to* $\mathbf{R}_d$ *can be computed as*

$$\mathbf{R}_e = \mathbf{R}_d\mathbf{R}^{-1} = \mathbf{R}_d\mathbf{R}^{\top}. \tag{5.9}$$

Matrix $\mathbf{R}_e$ describes the rotation necessary to attain the desired end-effector orientation from an arbitrary orientation in the robot's workspace. In order to comply with the definition of E in (5.1) we should find a scalar function to express the end-effector's orientation error described by $\mathbf{R}_e$.

According to the Euler's angle theorem, any orientation $\mathbf{R} \in SO(n)$ is equivalent to a rotation about a fixed axis $\mathbf{k} \in \mathbb{R}^n$ through an angle $\phi \in [0, 2\pi)$. To find the equivalent axis representation of $\mathbf{R}_{e,}$ let us use the unit quaternion expression

$$\mathbf{Q}_e = \mathbf{Rot}\left(\phi_e, \mathbf{k}_e\right) = \left(\cos\left(\frac{\phi_e}{2}\right), \sin\left(\frac{\phi_e}{2}\right)\mathbf{k}_e\right), \tag{5.10}$$

where $q_0 = \cos\left(\frac{\phi_e}{2}\right)$ is the scalar component, and $\mathbf{q}_e = \sin\left(\frac{\phi_e}{2}\right)\mathbf{k}_e$ is the vector component of the unit quaternion. In (5.10), the angle ϕ_e represents the error

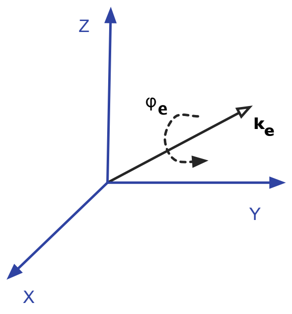


Figure 5.3: Euler angle-axis representation.

angle and $\mathbf{k}_e$ the axis of rotation (see Figure 5.3).

Further, ϕ_e amounts for the absolute error angle of $\mathbf{R}$ relative to $\mathbf{R}_d$ represented in $\mathbf{R}_e$. Thus, it can be taken as a measure of the orientation error. To extract ϕ_e from $\mathbf{R}_e$, the expression for the scalar part of a unit quaternion gives

$$q_0 = \frac{1}{2}\sqrt{n_{e_x} + o_{e_y} + a_{e_z} + 1}, \tag{5.11}$$

where n_{e_x}, o_{e_y}, and a_{e_z} are components of the square matrix $\mathbf{R}_e$. It follows that the error angle can be derived from (5.10) and (5.11) as

$$\phi_e = 2\cos^{-1}\left(\frac{1}{2}\sqrt{n_{e_x} + o_{e_y} + a_{e_z} + 1}\right). \tag{5.12}$$

Definition 3 (Orientation Error). *Let us consider the equivalent axis representation of the orientation error matrix $\mathbf{R}_e$, such that, the orientation is represented as the rotation about an axis $\mathbf{k}_e$ by an angle ϕ_e. The orientation error E_O can thus be defined as the error angle ϕ_e, obtained from $\mathbf{R}_e$ as*

$$E_O = 2\cos^{-1}\left(\frac{1}{2}\sqrt{n_{e_x} + o_{e_y} + a_{e_z} + 1}\right). \tag{5.13}$$

where n_{e_x}, o_{e_y}, and a_{e_z} are the diagonal components of matrix $\mathbf{R}_e$.

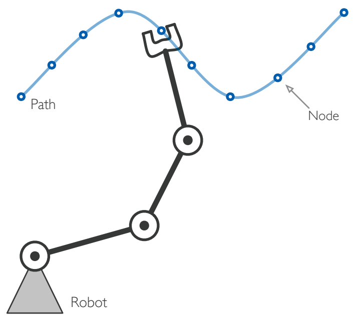


Figure 5.4: Robot geometric path.

5.2 Optimization Model: Path Generation Problem

Robot tasks are commonly described as end-effector trajectories that specify the position, velocity, and acceleration required to follow a desired motion. Similarly, trajectories can also be defined in the configuration space in terms of joint variables. A trajectory becomes a *path* when the velocity and acceleration are ignored, therefore, a path gives a purely geometrical description of the task in terms of pose or joint coordinates (see Figure 5.4).

Let us consider a desired Cartesian path defined as a discrete sequence of N_k end-effector pose coordinates (nodes). The path generation problem thus consists in mapping the desired motion in the Cartesian space into its corresponding joint path in the configuration space. Therefore, solving the path generation problem is equivalent to solving the inverse kinematics for each target node in the end-effector path. Consequently, the intuition introduced above for the kinematics inversion problem can be consistently tweaked to extend the optimization model from one to a set of target pose coordinates.

Two optimization frameworks can be proposed to solve the path generation problem. A first approach is to define a *path* joint configuration space — of dimension $N * N_k$ — to find a solution that minimizes the pose error of all nodes altogether (González et al., 2009). In the second approach the path generation problem is divided into N_k cycles —one for each node in the path —and in each cycle, the inverse kinematics problem is solved for a single node (González & Blanco, 2013).

The main advantage of the first approach is that a solution is obtained for all nodes at once by performing the search over an extended configuration space; however, the complexity of the problem also increases with the dimension of the search space. Further, this strategy yields a rigid framework that does not allow on the fly adjustments of the task due to sudden changes on the environment. In contrast, in the second strategy a global solution is obtained after N_k runs of the optimization process. Nonetheless, after each cycle the constraints on the task —and the task itself —can be updated to set new targets to the onwards nodes. This affords flexibility to the optimization framework, and for this reason, it will be considered for the path generation problem.

A path defines an ordered sequence of intermediaries locations of the end-effector within the workspace. By minimizing the pose error, we are able to find the joint configuration that yields the target end-effector pose in each node. This implies finding one of the multiple possible solutions of the inverse kinematics problem. Without imposing additional constraints over the solution space this might lead to abrupt joint displacement from one node to the next. Thus, to guarantee smooth joint transitions, a regularization term is included in the objective function to minimize the joint displacement between adjacent nodes. This is expressed in the configuration space as

$$\Gamma_k = \frac{1}{2\pi}\|\mathbf{q}_k - \mathbf{q}_{k-1}\|_2, \quad k = 1, 2, \ldots, N_k. \tag{5.14}$$

Therefore, for the kth node the objective function is expressed as the weighted sum of the pose error E_k (5.1) and the total joint displacement Γ_k (5.14) as

$$F_k = wE_k + (1 - w)\Gamma_k, \tag{5.15}$$

with $w \in (0, 1]$. For high-accuracy solutions the value of the weighting factor w should be set close to one, to emphasize the error minimization on the end-effector pose.

Thus, given a target path in the Cartesian space defined as a set of N_k pose nodes

$$\Omega_d = \left(\xi_{d_1}, \ldots, \xi_{d_k}, \ldots, \xi_{d_{N_k}}\right),$$

for the kth node, a constrained optimization model can be written as:

$$\min_{\mathbf{q}_k \in \mathcal{Q}} F_k \quad \text{subject to:} \quad \begin{array}{c} \xi_k = fk(\mathbf{q}_k) \\ \mathcal{Q} = \{\mathbf{q_k} \mid G(\mathbf{q_k}) \geq 0 \wedge H(\mathbf{q}_k) \geq 0\} \end{array}, \qquad k = 1, \ldots, N_k. \tag{5.16}$$

where G and H are constraints defined as in (5.3) and (5.4).

5.3 Solution to the Inverse Kinematics Problem

In this section, a numerical optimization framework based on the δDE scheme is proposed to solve the inverse kinematics problems formulated in the optimization models (5.2) and (5.16). The main advantage of using an optimization framework for kinematics inversion, instead of the damped least-squares methods, is the lack of use of the Jacobian matrix as mapping function. Thereby, numerical instability in the vicinity of singular joint configurations is completely overcome.

δDE is a hybrid population-based direct search approach. δDE combines the powerful exploratory behaviour of the standard DE scheme (`DE/rand/1/bin`) with a migration operator, called discarding, to enhance the local search features of the algorithm. The aim is to improve the convergence speed and solution accuracy of standard DE by balancing its exploratory and exploitative search behaviour.

δDE performs an iterative process of mutation, recombination, selection, and discarding over a population to produce a solution with the minimum fitness value. To counteract the mainly exploratory features of the variation mechanisms of mutation and recombination, discarding implements a twofold strategy: migration and local search. Thereby, in each generation a small percentage of the worst solutions in the population are replaced (migration) by solutions sampled from the vicinity of good ones (local search).

Discarding combines migration and local search into a single operator controlled by three parameters, namely the discarding size (δ), elite pool size (β), and standard deviation (σ). The first controls the size of migration and the last two regulate the elitism and length of the local search. The discarding size and the elite pool size are usually set as a percentage of the population size, whereas the standard

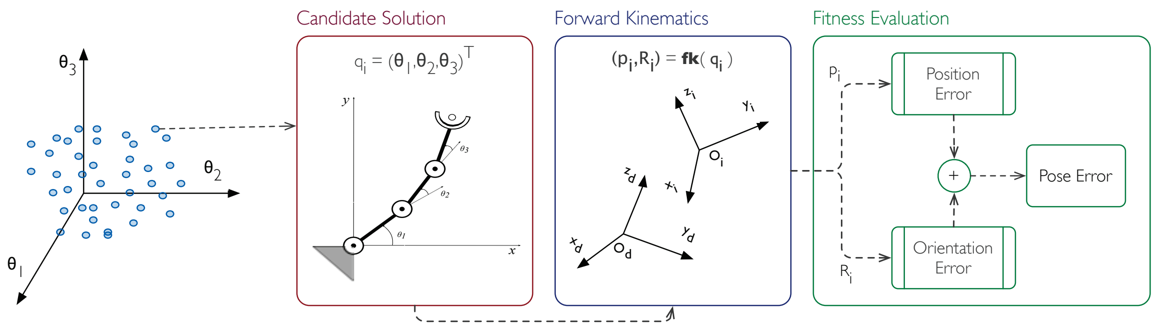


Figure 5.5: Fitness evaluation scheme.

deviation is defined in the units of the search variables. Further, by setting σ to zero, discarding can be simplified into a migration-only algorithm.

In this scheme, each candidate solution in the population represents a joint configuration vector coded as a floating-point array. The pose error function (5.1) defines a fitness value that measure the proximity to the optimal solution. Such that, as a candidate solution approaches the optimum, its fitness decreases towards zero. For the path generation problem this function is replaced with the objective function in (5.15). Note that the fitness function is defined on the task space, whereas the population is defined as a subset of the configuration space of the robot. Therefore, the fitness evaluation would comprise two steps: (i) each candidate solution is mapped into end-effector coordinates in the task space via forward kinematics, and (ii) the corresponding pose error is computed relative to the target position and orientation coordinates. This is schematically illustrated in Figure 5.5, where the arrows heads indicate the direction of the evaluation process. Finally, the solution is obtained as the joint configuration with the lowest fitness in the population after convergence.

5.4 Summary

The inverse kinematics refers to the problem of determining the joint configuration vector that yields a desired end-effector position and orientation in the robot's workspace. Although, analytical solutions to the inverse kinematics problem might be available for a class of open kinematics chains, more complex kinematics structures often require numerical methods to iteratively solve their inverse kinematics.

In this chapter, we have tackled this fundamental kinematics problem from a numerical optimization approach. In this formulation, the optimization model defines a direct search over the configuration space to minimize the end-effector pose error in the robot's task space. However, instead of relying on the Jacobian matrix as mapping function between the task and configuration space, the mapping is done via the forward kinematics equations. Thereby, the numerical instability of the Jacobian matrix around singular joint configurations is overcome. This formulation was further extended to address the path generation problem of robot manipulators.

Chapter 6

δDE-based Inverse Kinematics: Simulation Results

In this chapter, the δDE algorithm is tested against the standard DE as kinematics inversion methods. We have considered the kinematics of three benchmark robot manipulators moving in a free-of-obstacles workspace, thus that, the only constraints on the solutions space are the lower and upper joint displacement limits. Further, to examine the influence of *discarding* on the convergence behaviour of δDE, different parameter settings have been tested in a simulation environment.

6.1 Simulation Setup

The convergence behaviour has been examined under three different simulation setups programmed in MATLAB®:

- **Workspace.** This simulation experiment examined the performance of the algorithm over the reachable workspace of the robot. The test consisted of solving the inverse kinematics problem of N_ξ target poses randomly generated from the robot workspace.

- **Pose Testbed.** In a second simulation scenario, a testbed with four known target pose coordinates was considered. The aim was to set different search landscapes for the optimization problem to examine the algorithm repeatability. For each experimental point ξ_i $(i = 1, \ldots, 4)$, N_t independent trials were

Table 6.1: Simulation configuration for the IK problem.

Test	N_t	N_ξ	N_k	ϵ_P	ϵ_O
Workspace	1	1000	–	0.1mm	0.1°
Pose Testbed	100	4	–	0.1mm	0.1°
Path Generation	100	–	21	1mm	–

conducted on the simulation environment.

- **Path Generation.** A third simulation setup consisted of solving the path generation problem of a task. The path was defined as a set of N_k consecutive nodes that geometrically described the target end-effector pose. In this case, only the end-effector position has been considered, thus neglecting the orientation. The inverse kinematics was thus solved sequentially for each node, by minimizing the end-effector position error (E_P), for positioning accuracy of the robot, and the joint displacement from the previous node (Γ), for smoothness in the joint transitions between adjoining nodes. The termination criteria for each cycle was set to a maximum position error tolerance, equal to 1mm. Further, to emphasize accuracy along the path, the weighting factor w in equation (5.15) was set to a value of 0.8. The parameter configuration for each simulation setup is listed in Table 6.1.

The evaluation criteria followed to assess the performance of the proposed kinematic inversion methods, for all simulation setups, have been the following:

- **Errors:** At each run of the algorithm the minimum total error E, position error E_P, and orientation error E_O, obtained after $g_{\max}$ generations were recorded. Once the simulation experiment was completed, the overall average errors (AVG) and standard deviations (STD) were computed.

- **Success rate (SR):** Number of success trials obtained after a number of runs of the algorithm, expressed as a percentage value. Any trial is considered a success, if after a predefined number of generations ($g_{\max}$), the best joint configuration vector yields a position and orientation errors below a predefined accuracy tolerance values, denoted by ϵ_P and ϵ_O, respectively. Results are reported following the notation AVG $\pm$ STD (SR).

- **No. generations (GEN):** Average number of generations required to reach the position and orientation error thresholds, ϵ_P and ϵ_O.

- **Acceleration rate (AR):** Ratio of improvement on the number of generations (GEN) expressed as a percentage value.

- **Population Diversity**. Measures the *genetic* variation among the candidates solutions of the same population. It is defined as the sum of the distance from the average point ($\bar{q}$) to each individual of the population, that is

$$\text{Diversity} = \frac{1}{|N_P| \cdot \|\mathbf{q}^{(hi)} - \mathbf{q}^{(lo)}\|_2} \cdot \sum_{i=1}^{N_P} \sqrt{\sum_{j=1}^{D} (q_{ij} - \bar{q}_j)^2}.$$

6.2 Control Parameters

For fair comparison, the population size (N_P), crossover rate (C_R), and mutation scale factor (F) of both DE schemes were set to the same values. Accordingly, the mutation scale factor ($F = 0.5$) and crossover rate ($C_R = 0.8$) were set based following the guidelines suggested in (Ronkkonen et al., 2005). Note that a large value of C_R yields a mainly exploratory search dynamic controlled by the value of F. In addition, the population size was adjusted, from its nominal value according to the problem dimension ($D = N$), to meet a trade-off between the population diversity and the computational cost. Table 6.2 summarizes the parameter setup values used for each robot manipulator.

Furthermore, since we lack of a theoretical analysis on the dynamic of δ, σ, and β, these parameters have been tuned based on empirical data. Preliminary results suggested that large values of δ noticeable increase the convergence speed at the expense of increasing computational cost; whereas parameters σ and β control the intensification pressure around the best solutions. Thus, δ, σ, and β have been set to balance speed rate, accuracy, and efficiency of the memetic search scheme. For readability, the parameter setting of δDE will be denoted by the triple (δ, β, σ).

Table 6.2: Control parameter setting of δDE.

DOF	F	C_R	N_P	$g_{\max}$
3			50	100
6	0.5	0.8	150	300
7			250	500

6.3 Planar Robot Manipulator

Consider the planar robot manipulator schematically illustrated in Fig. 6.1. Each angle θ_j represents a joint variable in the configuration space, such that, $\mathbf{q} = (\theta_1, \theta_2, \theta_3)^\top$ denotes the configuration vector of the robot ($N = 3$). The end-effector motion is constrained to the two-dimensional Cartesian space, thus, the position vector $\mathbf{p} = (p_x, p_y)$ and the angle φ respect to the horizontal axis completely describe the end-effector pose ($m = 3$). Table 6.3 summarizes the D-H kinematic parameters and joint limits.

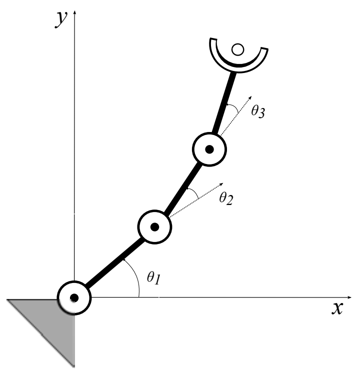


Figure 6.1: Planar robot manipulator with revolute joints (3 DOF).

Table 6.3: Kinematic parameters and joint limits of the planar robot manipulator.

Joint	$\mathbf{a}_j$	α_j	$\mathbf{d}_j$	θ_j	$\mathbf{q}_j^{(lo)}$	$\mathbf{q}_j^{(hi)}$
1	0.20m	0°	0m	θ_1	-90°	90°
2	0.15m	0°	0m	θ_2	-90°	90°
3	0.10m	0°	0m	θ_3	-90°	90°

6.3.1 Workspace

For this simulation setup the inverse kinematics was solved for N_ξ different target coordinates, previously sampled from the workspace of the robot. The goal was to examine the overall performance of δDE over the reachable workspace of the robot manipulator under different parameter settings.

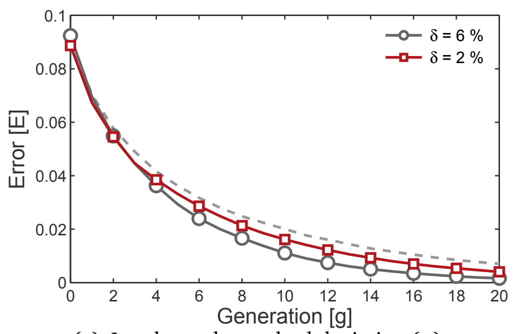
The discarding mechanism of δDE, combines a migration strategy with local search. To test both features, a migration-only scheme was first considered by setting to zero the standard deviation of the local search (σ). To further stress an elitist replacement strategy, the discarding size (δ) and the elite pool size (β) were set to 6 % and 10 % of the population size, respectively. Thereby, in each new generation the worst three candidates were replaced by copies of candidate solutions randomly sampled from the five best fitted elements in the population. Simulation results are reported in Table 6.4.

According to these results, the migration-only strategy significantly improves the convergence speed by duplicating good solutions in each generation; however, it also promotes a lower diversity among the candidate solutions. The loss of diversity negatively impacts the exploration features of the search, and consequently, the efficiency of the algorithm. This is observed as a slight decrease in the success ratio of δDE. However, these values are still competitive in comparison to the standard DE performance.

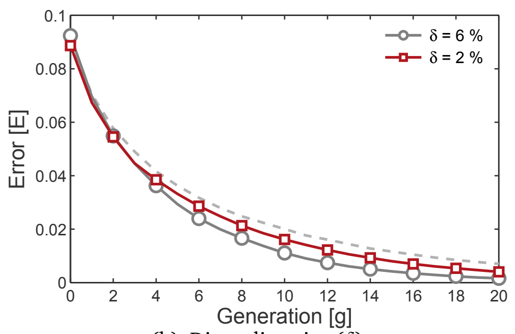
With the aim to examine the influence of the local search in the overall improvement of the search performance of δDE, the standard deviation (σ) was increased to 0.2° while keeping the values of δ and β unaltered. Thus this time, a relatively small number of individuals were replaced in each generation by densely

Table 6.4: Simulation results of standard DE and δDE as kinematics inversion methods over the workspace of the planar robot manipulator.

	DE	**δDE (δ,β,σ)**		
		(6 %, 10 %, 0°)$_{(1)}$	(6 %, 10 %, 0.2°)$_{(2)}$	(2 %, 10 %, 0°)$_{(3)}$
E	2.32E-05 ± 2.46E-04(98.9)	1.58E-05 ± 2.59E-04(98.7)	1.47E-05 ± 2.61E-04(99.7)	**2.19E-06 ± 7.06E-06(99.9)**
$\mathbf{E_P}$ (mm)	1.05E-02 ± 2.52E-02(98.9)	1.54E-02 ± 2.58E-01(98.7)	1.45E-02 ± 2.61E-01(99.7)	**1.47E-03 ± 6.33E-03(99.9)**
$\mathbf{E_O}$ (deg.)	5.62E-03 ± 1.10E-01(99.9)	1.48E-04 ± 2.83E-03(100)	**8.05E-05 ± 2.72E-04(100)**	3.04E-04 ± 8.28E-04(100)
GEN	62.03 ± 13.05	**31.43 ± 5.82**	33.56 ± 5.96	46.85 ± 8.03
AR (%)		**49.33**	45.90	24.47



(a) Local search standard deviation (σ).



(b) Discarding size (δ).

Figure 6.2: Convergence curves of DE (dashed line) and δDE as inverse kinematics methods for a planar robot manipulator. Curves correspond to the average error obtained from one thousand trials over the reacheable workspace of the robot. The discarding mechanism was set to different values of the (a) standard deviation (σ), and (b) discarding size (δ) to test different features of the search.

distributed candidate solutions around a small set of the top best solutions in the current population.

According to the results in Table 6.4, a better overall accuracy performance is obtained when combining migration with local search, without considerably deteriorating the convergence speed improvement provided by migration. Figure 6.2(a) illustrates the marginal impact that the increase of the local search length distribution has on the convergence speed during the early stages of the optimization process.

Although large values of the discarding size (δ) can significantly improve the convergence speed (up to 50 % respect to the standard DE), it also increases the computational cost by proportionally increasing the number of function evaluations in each generation. Therefore, we ran a third simulation experiment in which the value of δ was reduced to 2 % of the population size, while the other parameters were kept unaltered ($\beta = 10$ % and $\sigma = 0°$). Results in Table 6.4 show that by decreasing half of the total number of discarded solutions in each generation, the convergence speed decreased from 50 % ($\delta = 6$ %) down to 27.47 % ($\delta = 2$ %) of average acceleration rate (see Figure 6.2(b)). These preliminary results would suggest that in a migration-only scheme the impact of the discarding size value is almost inversely proportional to the number of generations required to converge. Noteworthy, in this case a lower value of the discarding size improved the solution

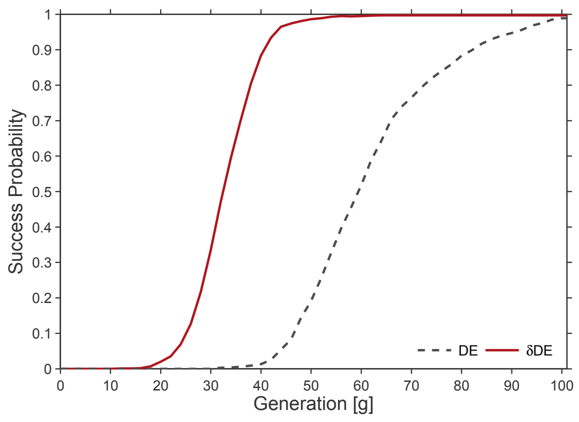


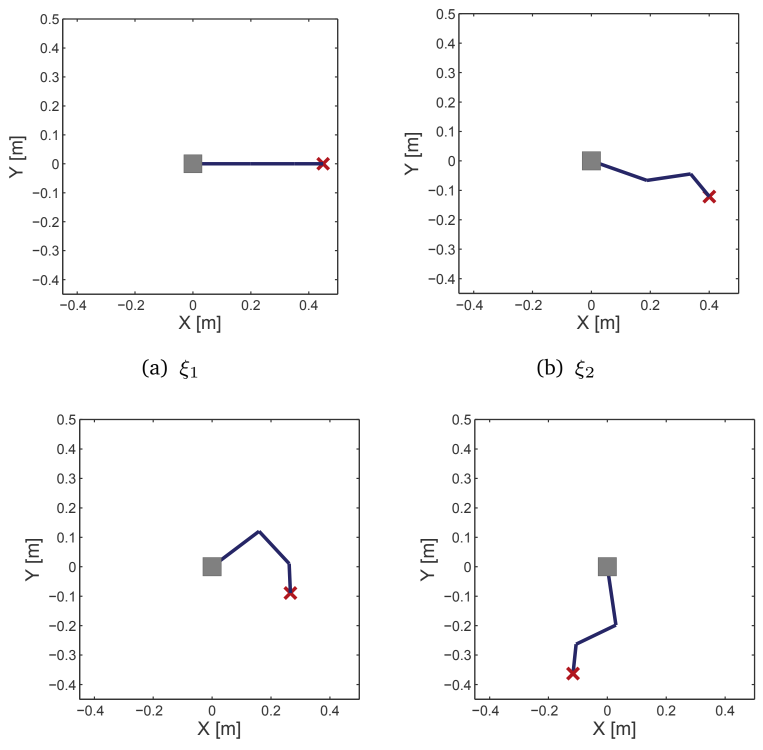
Figure 6.3: Probability of success of standard DE (dashed line) and δDE (solid line) over successive generations.

accuracy and success ratio performance.

These simulation results showed that δDE outperforms standard DE in convergence speed. It exhibits an improvement of almost 50 % in the average number of iterations required to reach a position and orientation error below the predefined threshold values. Moreover, δDE also provides competitive accuracy values in comparison to the standard DE while improving its convergence success ratio. Figure 6.3 compares the probability of success of DE and δDE (6 %, 10 %, 0.2°) during the evolutionary process. It can be observed that δDE substantially improves the efficiency of the search, such that, by the fiftieth generation the probability of convergence is almost of 100 %, compared to the approximately 20 % probability of the standard DE.

6.3.2 Pose Testbed

The inverse kinematics of the planar robot manipulator admits two solutions for any end-effector pose in the workspace, commonly knowns as elbow-up and elbow-down configurations. The exception occurs when the robot is at a singular joint configuration, and thus, the inverse kinematics only has one solution. This feature is examined by solving the inverse kinematics problem of four different end-effector pose coordinates, three of which correspond to non-singular joint configurations (ξ_2, ξ_3, ξ_4), and a one to a singularity (ξ_1). The target end-effector coordinates and one of the two possible joint configuration solution are shown in the Figure 6.4. The



(a) ξ_1 (b) ξ_2 (c) ξ_3 (d) ξ_4

Figure 6.4: Simulation testbed for the inverse kinematics of the three DOF robot manipulator.

performance of δDE was again tested for the same parameter settings considered in the previous simulation experiment.

The simulation results reported in Table 6.5 are consistent with the convergence performance observed in the previous simulation experiment. As in that case, the migration-only strategy provided a noticeable 46 % improvement on the convergence speed, in comparison to standard DE and the other two δDE configurations; but it also deteriorated the efficiency of the search by decreasing its success ratio. Again, the best overall performance was obtained combining migration and local search.

These results allow us to highlight two important advantages of δDE as kinematics inversion method. Firstly, the performance of population-based optimization methods does not degrade around singularities. And secondly, in comparison to standard DE, δDE provides better accuracy (lower errors) and competitive success

Table 6.5: Simulation results of the standard DE and δDE as kinematics inversion methods of a planar robot manipulator.

	DE	**δDE** (δ,β,σ)		
		(6 %, 10 %, 0°)$_{(1)}$	(6 %, 10 %, 0.2°)$_{(2)}$	(2 %, 10 %, 0°)$_{(3)}$
ξ_1		$\mathbf{p}_1$ = (0.45,0)m ϕ_1 = 0°		
E	7.71E-07 ± 5.40E-07(100)	4.00E-06 ± 2.08E-05(98)	2.74E-07 ± 1.67E-06(100)	**2.49E-07** ± **1.45E-07(100)**
E_P (mm)	4.65E-04 ± 4.15E-04(100)	3.91E-03 ± 2.05E-02(98)	1.96E-04 ± 1.35E-03(100)	**1.52E-04** ± **1.22E-04(100)**
E_O (deg.)	1.24E-04 ± 1.28E-04(100)	3.62E-05 ± 1.88E-04(100)	**3.09E-05** ± **1.31E-04**(100)	3.89E-05 ± 3.29E-05(100)
GEN	48.72 ± 5.61	**26.52** ± **5.73**	28.24 ± 6.51	36.83 ± 5.75
AR (%)		**45.57**	42.04	20.30
ξ_2		$\mathbf{p}_2$ = (0.4004,-0.1213)m ϕ_2 = -50.6°		
E	2.42E-05 ± 2.70E-05(100)	4.16E-06 ± 3.62E-05(99)	**5.68E-07** ± **1.62E-06(100)**	4.44E-06 ± 8.90E-06(100)
E_P (mm)	1.68E-02 ± 2.09E-02(100)	4.11E-03 ± 3.61E-02(99)	**3.34E-04** ± **7.54E-04(100)**	4.11E-03 ± 3.61E-02(99)
E_O (deg.)	3.10E-03 ± 4.83E-03(100)	**1.77E-05** ± **1.61E-04(100)**	9.68E-05 ± 3.81E-04(100)	6.50E-04 ± 1.88E-03(100)
GEN	72.36 ± 10.52	**31.77** ± **3.88**	34.82 ± 5.77	49.84 ± 8.82
AR (%)		**56.09**	51.88	31.12
ξ_3		$\mathbf{p}_3$ = (0.2661,-0.0893)m ϕ_3 =-87.34°		
E	6.34E-07 ± 4.99E-07(100)	**3.35E-11** ± **6.35E-11(100)**	8.75E-08 ± 6.20E-08(100)	1.98E-07 ± 1.53E-07(100)
E_P (mm)	4.15E-04 ± 4.00E-04(100)	**2.30E-08** ± **4.08E-08(100)**	6.27E-05 ± 5.30E-05(100)	1.46E-04 ± 1.06E-04(100)
E_O (deg.)	1.11E-04 ± 1.39E-04(100)	**5.33E-09** ± **1.78E-08(100)**	1.26E-05 ± 1.42E-05(100)	2.63E-05 ± 4.32E-05(100)
GEN	54.66 ± 5.26	**31.04** ± **3.94**	32.72 ± 4.95	43.61 ± 4.79
AR (%)		**43.21**	40.14	20.22
ξ_4		$\mathbf{p}_4$ =(-0.1166,-0.3626)m ϕ_4 = -95.98°		
E	7.39E-07 ± 4.78E-07(100)	1.72E-07 ± 1.72E-06(100)	**1.04E-07** ± **5.69E-08(100)**	2.41E-07 ± 1.39E-07(100)
E_P (mm)	4.98E-04 ± 3.74E-04(100)	1.21E-04 ± 1.21E-03(100)	**6.93E-05** ± **4.61E-05(100)**	1.66E-04 ± 1.16E-04(100)
E_O (deg.)	1.05E-04 ± 1.11E-04(100)	2.26E-05 ± 2.26E-04(100)	**1.49E-05** ± **1.59E-05(100)**	3.30E-05 ± 3.39E-05(100)
GEN	52.22 ± 5.42	**30.95** ± **5.56**	32.59 ± 5.15	42.21 ± 5.11
AR (%)		**40.73**	37.59	19.17

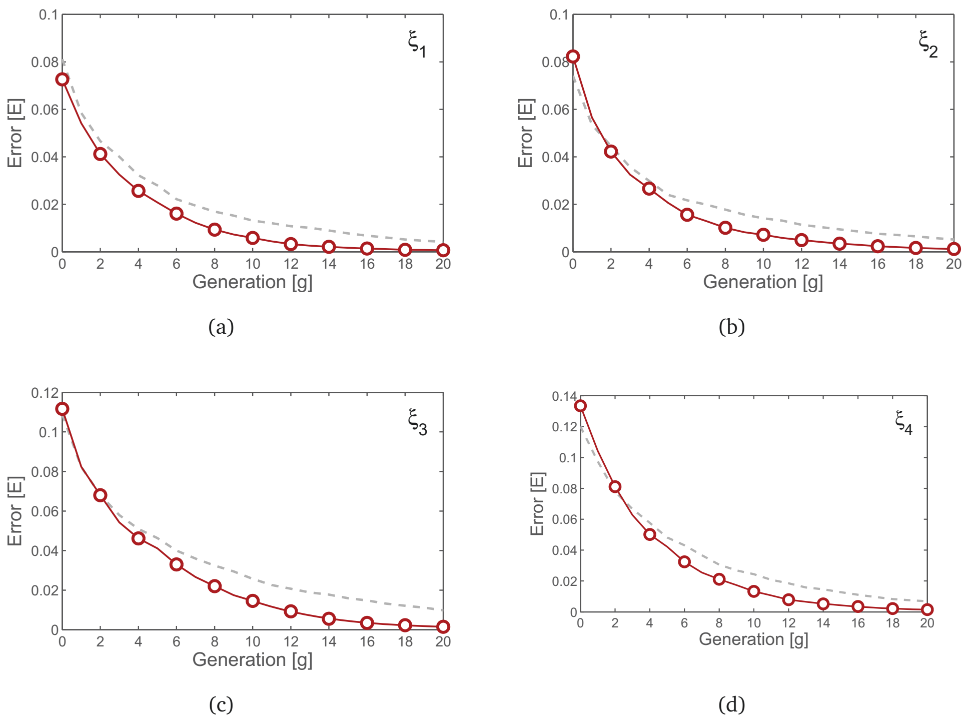
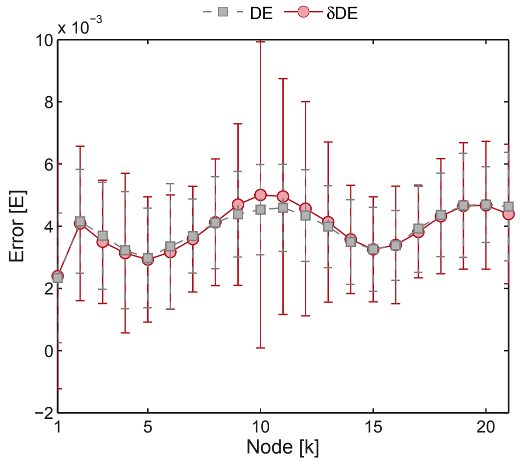


Figure 6.5: Convergence curves of δDE (solid line) and standard DE (dashed line) on the kinematics inversion of four different target pose vector of a planar robot manipulator. Curves correspond to the average fitness over one hundred trials for each target point in the testbed, and discarding parameters (6 %, 10 %, 0.2°).

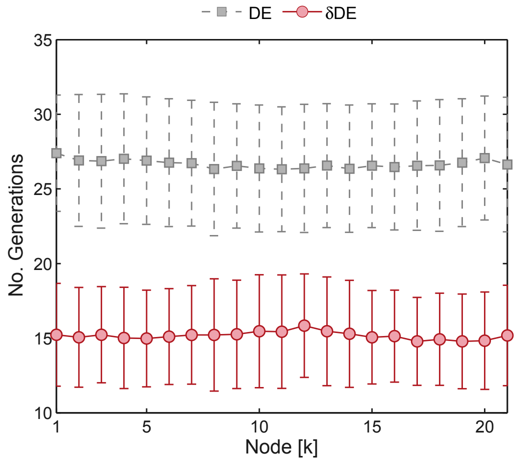
rates values. Furthermore, experimental results also revealed a remarkable 46.40 % acceleration improvement on the convergence speed (see Figure 6.5). This shows that our memetic approach is able to enhance the local search capabilities of DE without deteriorating its global search performance.

6.3.3 Path Generation

A third simulation experiment was set to test the flexibility of the δDE scheme as kinematic inversion method. A joint path was solved from a sequence of target nodes that defined a circular path, with origin at (0.2, 0.2)m from the robot base and a radius of 0.05m. For each node of the path, the inverse kinematics was solved with a tolerance of 1mm. Also, it was assumed that the robot manipulator was initially at its zero configuration, i.e., $\mathbf{q}_{k=0} = (0^\circ, 0^\circ, 0^\circ)^\top$. Based on the previous



(a) Average position error.

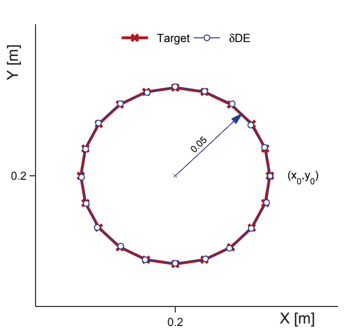


(b) Average number of generations.

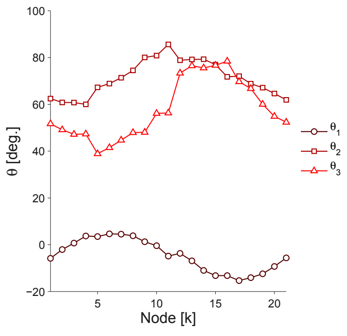
Figure 6.6: Comparison of the accuracy and convergence speed of standard DE (dashed line) and δDE (solid line) at each node of a circular target path by a planar robot manipulator. (a) Position accuracy. (b) Number of generations for convergence.

simulation experiments, δDE was set to (6 %, 10 %, 0.2°) as discarding parameters. Results correspond to one hundred independent trials.

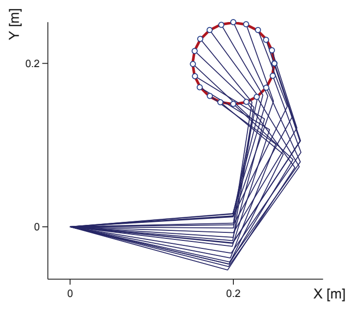
Simulation results shows competitive values between both algorithms on the overall position accuracy and joint total displacement along the path. In this case, δDE produced an average position error of 1.53E+01mm, which gives an average accuracy of approximately 0.73mm for each node. The largest errors occurred between the tenth and fifteenth nodes (third quadrant of the circumference) due to the proximity of the second and third joints to the upper joint limit (+90°). However, this could be circumvented by increasing the value of the weighting factor w in (5.16), and thus, allowing a greater displacement of the first joint, yet still generating smooth joint transitions. Further, the most significant improvement was observed in the convergence speed, outperforming DE with a 43.13 % of acceleration rate. A comparison on the distribution of the position error and the number of generations in each node of the path of standard DE and δDE is illustrated in Figure 6.6. Moreover, in Figure 6.7 the target circular path and the solution joint path obtained for a single run of δDE are illustrated. In this figure it can be observed that the obtained joint path solution imposes a smooth transition between nodes while following the target path with high-accuracy. These results verify the flexibility of



(a) Target path.



(b) Joint path.



(c) Planar robot manipulator.

Figure 6.7: δDE solution to the path generation problem for a planar robot manipulator.

optimization-based inverse kinematics approach.

6.4 Anthropomorphic robot manipulator

In this section, we consider the kinematics inversion problem of an anthropomorphic robot manipulator. The PUMA robot manipulator is a benchmark robotic platform widely used in industrial and research applications. The anthropomorphic kinematics of the arm features six links and six rotational joints (6 DOF), arranged such that the first three joints (waist - shoulder - elbow) position the end-effector and the last three (wrist) orient it (see Figure 6.8). Kinematic D-H parameters and joint limits of the PUMA robot manipulator are listed in Table 6.6.

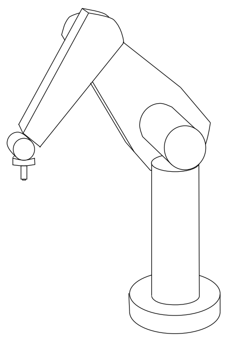


Figure 6.8: Antropomorphic robot manipulator with revolute joints (6 DOF).

Table 6.6: Kinematic parameters and joint limits of the six DOF robot manipulator (Elgazzar, 1985).

Joint	$\mathbf{a}_j$	α_j	$\mathbf{d}_j$	θ_j	$\mathbf{q}_j^{(lo)}$	$\mathbf{q}_j^{(hi)}$
1	0m	90°	0m	θ_1	-160°	160°
2	0.4318m	0°	0m	θ_2	-225°	45°
3	0.0191m	-90°	0.1254m	θ_3	-45°	225°
4	0m	90°	0.4318m	θ_4	-110°	170°
5	0m	-90°	0m	θ_5	-100°	100°
6	0m	0°	0m	θ_6	-266°	266°

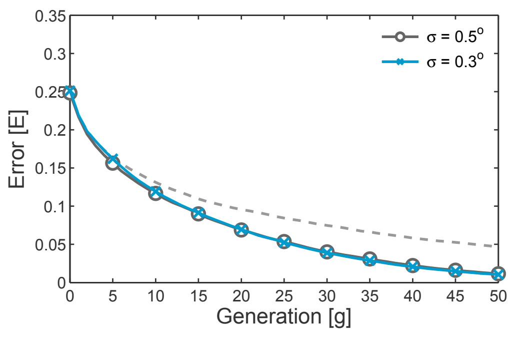
6.4.1 Workspace

We tested the convergence behaviour of δDE for different parameter settings. A preliminary study on the configuration of the discarding parameters revealed that a fair trade-off between accuracy, speed, and convergence efficiency was obtained with parameters (10 %, 75 %, 0.5°). Note that, a large value of β implies that the objective function demands a higher diversity in the elite parent pool. To further investigate the influence of the discarding size on the search dynamic, this value was decreased from 10 % to 5 %. Finally, a third parameter setting was considered by decreasing the value of the standard deviation from 0.5° to 0.3 ° . Lower standard deviation values produce more densely distributed candidates around good solutions, and thus, it is related to the local search length. Simulation results are reported in Table 6.7.

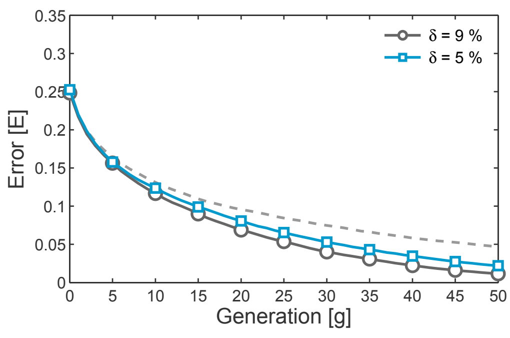
Simulation results verify that δDE outperforms DE in convergence speed, solution accuracy, and success ratio. Furthermore, δDE was, in average, around 36 %

Table 6.7: Simulation results of standard DE and δDE as kinematics inversion methods over the workspace of the anthropomorphic robot manipulator.

	DE	**δDE (δ,β,σ)**		
		(10 %, 75 %, 0.5°)$_{(1)}$	(5 %, 75 %, 0.5°)$_{(2)}$	(10 %, 75 %, 0.3°)$_{(3)}$
$\mathbf{E}$	1.55E-04 ± 9.74E-04(90.6)	2.20E-04 ± 2.25E-03(98.1)	1.58E-04 ± 1.35E-03(97.4)	**1.29E-04** ± **1.37E-03(98.1)**
$\mathbf{E_P}$ (mm)	8.71E-02 ± 6.41E-01(92.7)	7.10E-02 ± 1.37E+00(99.0)	8.08E-02 ± 9.13E-01(98.4)	**2.18E-02** ± **4.22E-01(99.1)**
$\mathbf{E_O}$ (deg.)	5.21E-02 ± 3.75E-01(93.9)	1.04E-01 ± 1.22E+00(99.1)	6.19E-02 ± 7.86E-01(98.9)	**7.83E-02** ± **9.50E-01(99.0)**
GEN	205.28 ± 43.56(90.6)	127.51 ± 25.04(98.1)	147.27 ± 28.93(97.4)	**117.55** ± **21.83(98.1)**
AR (%)		37.88	28.26	**42.74**



(a) Local search standard deviation (σ).



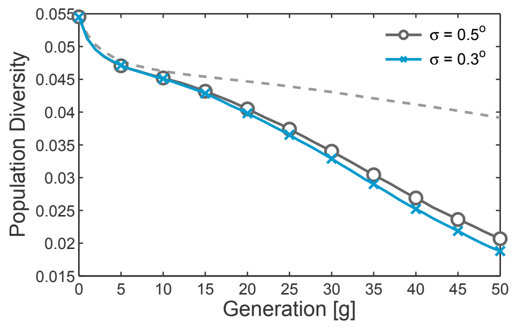
(b) Discarding size (δ).

Figure 6.9: Convergence curves of DE (dashed line) and δDE as inverse kinematcis methods for a anthropomorphic robot manipulator. Curves correspond to the average error obtained from one thousand trials over the reacheable workspace of the robot. The discarding mechanism was set to different values of the (a) standard deviation (σ), and (b) discarding size (δ) to test its robustness.

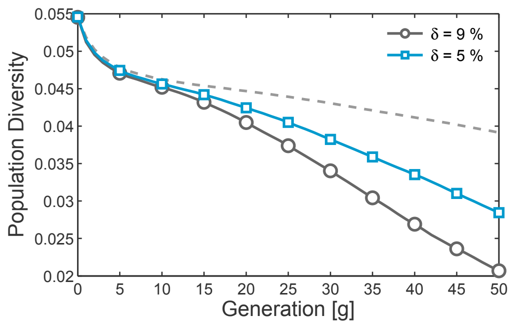
faster that standard DE in all discarding configurations. Although outstanding, these acceleration rates are considerably lower than the almost 50 % reached by δDE on the inverse kinematics of the planar robot manipulator. This is related to the parameter setting of δDE, as candidate solutions were replaced with sparsely distributed solutions in a more complex multimodal search landscape.

In general, the discarding size has a major impact on the performance of the algorithm, particularly, on its convergence speed. However, these results suggest a more complex dynamic with the other parameters. In fact, it was necessary to finely tune the local search length ($\sigma = 0.3^\circ$) to obtain a substantial improvement on the speed rate (42.74 %). In contrast, a lower value of the discarding size ($\delta = 5$ %) not only decreased convergence speed, but also deteriorated the overall δDE performance. This contrasts with the behaviour observed for the much simpler kinematics structure of the planar robot manipulator. Figure 6.9 illustrates the effect of discarding parameters on the convergence behaviour of δDE as kinematics inversion method.

The intuition behind the discarding mechanisms is based on the idea that by continuously replacing poor solutions with fitted elements, the population can be steered more rapidly towards the optimal solution. Thus far, simulation results have supported this intuition and they also have suggested that the amount in which



(a) Local search standard deviation (σ).



(b) Discarding size (δ).

Figure 6.10: Comparison of the population diversity of standard DE (dashed line) and different parameter configutarions of δDE (solid line). (a) Local search standar deviation, and (b) discarding size.

this process enhances the convergence speed is directly proportional to the value of the discarding size δ. However, seeding the population with a large number of similar copies of good solutions in each generation might deteriorate the population diversity.

In (Zaharie, 2003), the author argued that the exploration power of DE intrinsically depends on the diversity of the population. This is, the algorithm can explore more efficiently the search space as more diverse is the information contained therein the population. In a mainly exploratory scheme as the standard DE, the population diversity gradually decreases as the population converges. Conversely, a sharp decrease during early generations might indicate excessive local search and poor exploration. Therefore, the population diversity represents a good indicator of the equilibrium between intensification and diversification mechanisms.

Since discarding implements a local search-based migration strategy to overcome the exploitation deficiencies of standard DE, it would normally decrease the population diversity after each generation. Indeed, as observed in Figure 6.10, the population diversity dropped more steeply as the exploitation pressure on the population increases (large δ and a small σ). A further study on the convergence failures of δDE and standard DE provided insight on the effect that low population diversity has on the convergence performance of the algorithm. In this context, a solution was considered a convergence failure if after a number of generations ($g_{\max}$), its position error and/or orientation error was greater than the predefined

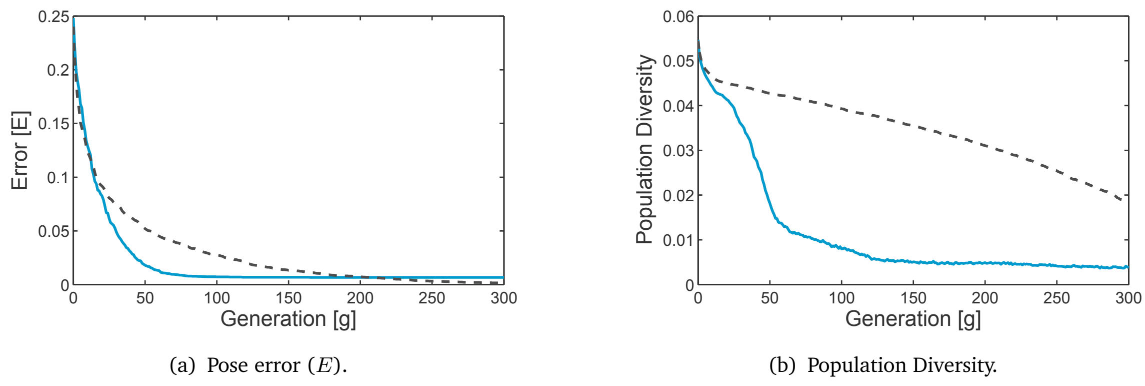


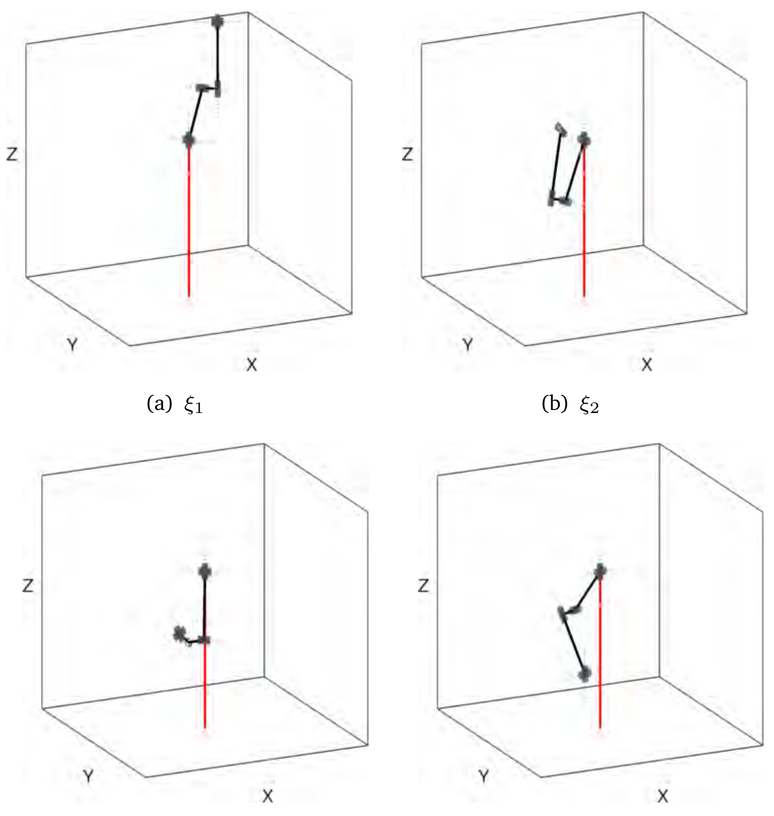
Figure 6.11: Comparison of the average convergance failure curves of standard DE (dashed line) and δDE (solid line) with parameters (10 %, 75 %, 0.3 °).

minimum error thresholds. Figure 6.11 compares the average error convergence curves and population diversity of standard DE and δDE failed solutions. As it can be observed, the steep descent in population diversity leads δDE to stagnates after few generations, whereas DE slowly approaches the minimum error value by gradually decreasing the population diversity. In both cases, the convergence behaviour is undesirable since exploitation and exploration are not balanced.

6.4.2 Pose Testbed

For this simulation experiment, a tesbed with four target points have been considered. Each point represents a desired position and orientation of the end-effector within the reachable workspace of the robot manipulator. Accordingly, the target pose will be represented by the position coordinates $\mathbf{p} = (p_x, p_y, p_z)$ and the minimal set of Euler angles $\Phi = \{\phi, \gamma, \psi\}$. The inverse kinematics of any feasible pose admits at least eight joint postures solutions. The target end-effector coordinates and one possible joint configuration solution are shown in Figure 6.12.

The higher multimodality of the search landscape — compared to the planar robot arm — compelled to modify the discarding parameter setting to adapt the mechanism to the search space characteristics. In this case, the problem demanded to increase exploitation, but within a broader elite pool to maintain diversity. Thus, as in the previous simulation experiment three parameter settings were considered for this simulation setup, namely (10 %, 75 %, 0.5°), (10 %, 75 %, 0.3°), and (5 %,



(a) ξ_1 (b) ξ_2 (c) ξ_3 (d) ξ_4

Figure 6.12: Simulation testbed for the 6 DOF PUMA robot manipulator.

75 %, 0.5°). For each experimental point in the testbed and parameter setting, one hundred independent runs of the algorithm were carried out on the simulation environment. And in every run the algorithm iterated until a maximum number of generations ($g_{\max}$ = 300). Simulation results are summarized in the Table 6.8.

Simulation results verify the superior performance of δDE in comparison of standard DE in terms of accuracy, convergence speed, and success rate. An analysis on the effect of the discarding parameters over the convergence behaviour of δDE suggested the influential role of the local search length on the solution accuracy, but more importantly on the improvement of the convergence speed. Further, by decreasing the standard deviation value from 0.5° to 0.3° , while discarding 10 % of the population in each generation, the acceleration rate was increased by more than a third to an average of 44 %. This is crucial in terms of the computational cost,

Table 6.8: Simulation results of DE and δDE as kinematics inversion methods of a six degrees of freedom robot manipulator.

	DE	δDE (δ,β,σ)		
		(10 %, 75 %, 0.5°)$_{(1)}$	(5 %, 75 %, 0.5°)$_{(2)}$	(10 %, 75 %, 0.3°)$_{(3)}$
ξ_1		$\mathbf{p}_1$ =(0.34, 0.18, 0.69)m Φ_1 =(-114.93, 36.50,-41.43)°		
E	1.08E-06 ± 1.39E-06(100)	3.78E-07 ± 6.46E-07(100)	3.90E-07 ± 5.86E-07(100)	**2.99E-07 ± 5.19E-07(100)**
E_P(mm)	5.27E-04 ± 8.11E-04(100)	1.81E-04 ± 2.88E-04(100)	1.87E-04 ± 2.91E-04(100)	**1.45E-04 ± 2.80E-04(100)**
E_O(deg.)	3.33E-04 ± 4.88E-04(100)	1.19E-04 ± 2.44E-04(100)	1.23E-04 ± 2.13E-04(100)	**9.35E-05 ± 1.79E-04(100)**
GEN	173.76 ± 18.69	125.48 ± 17.09	138.56 ± 16.63	**118.15 ± 15.68**
AR (%)		27.79	20.26	**32.00**
ξ_2		$\mathbf{p}_2$ =(-0.19, 0.03, 0.08)m Φ_2 =(138.60, 50.79, -50.17)°		
E	2.05E-05 ± 7.02E-05(96)	**5.76E-08 ± 1.22E-07(100)**	2.59E-07 ± 1.40E-06(100)	7.43E-08 ± 1.70E-07(100)
E_P(mm)	9.57E-03 ± 3.29E-02(96)	**2.92E-05 ± 6.20E-05(100)**	1.00E-04 ± 3.96E-04(100)	3.85E-05 ± 1.02E-04(100)
E_O(deg.)	1.27E-02 ± 4.64E-02(97)	**3.31E-05 ± 7.91E-05(100)**	1.85E-04 ± 1.24E-03(100)	4.19E-05 ± 8.97E-05(100)
GEN	208.25 ± 36.56	113.79 ± 15.45	135.16 ± 21.55	**107.29 ± 16.38**
AR (%)		45.36	35.10	**48.48**
ξ_3		$\mathbf{p}_3$ =(-0.37, -0,30, -0,295)m Φ_3 =(173.12, 116.37, -9.12)°		
E	4.95E-04 ± 1.73E-03(78)	4.88E-05 ± 4.87E-04(99)	**1.10E-07 ± 4.03E-07(100)**	4.88E-05 ± 4.87E-04(99)
E_P(mm)	2.57E-01 ± 1.06E+00(80)	**2.09E-05 ± 5.82E-05(100)**	6.32E-05 ± 2.68E-04(100)	2.76E-05 ± 8.34E-05(100)
E_O(deg.)	1.70E-01 ± 5.43E-01(82)	3.49E-02 ± 3.49E-01(99)	**3.33E-05 ± 1.02E-04(100)**	3.49E-02 ± 3.49E-01(99)
GEN	238.53 ± 30.64	122.21 ± 16.45	148.59 ± 23.29	**115.28 ± 16.19**
AR (%)		48.76	37.70	**51.67**
ξ_4		$\mathbf{p}_4$ =(-0.13, 0,01, -0,65)m Φ_4 =(41.0, 174.0, 154.0)°		
E	5.48E-05 ± 1.28E-04(90)	5.25E-05 ± 3.92E-04(98)	**2.33E-06 ± 1.07E-05(100)**	3.29E-05 ± 2.29E-04(98)
E_P(mm)	2.29E-02 ± 5.57E-02(93)	**1.79E-04 ± 4.30E-04(100)**	1.40E-03 ± 6.85E-03(100)	2.19E-04 ± 1.41E-03(100)
E_O(deg.)	2.11E-02 ± 5.76E-02(94)	3.46E-02 ± 2.59E-01(98)	**6.09E-04 ± 2.57E-03(100)**	2.16E-02 ± 1.51E-01(98)
GEN	224.63 ± 34.59	137.92 ± 21.81	163.80 ± 29.57	**127.70 ± 16.52**
AR (%)		38.60	27.08	**43.15**

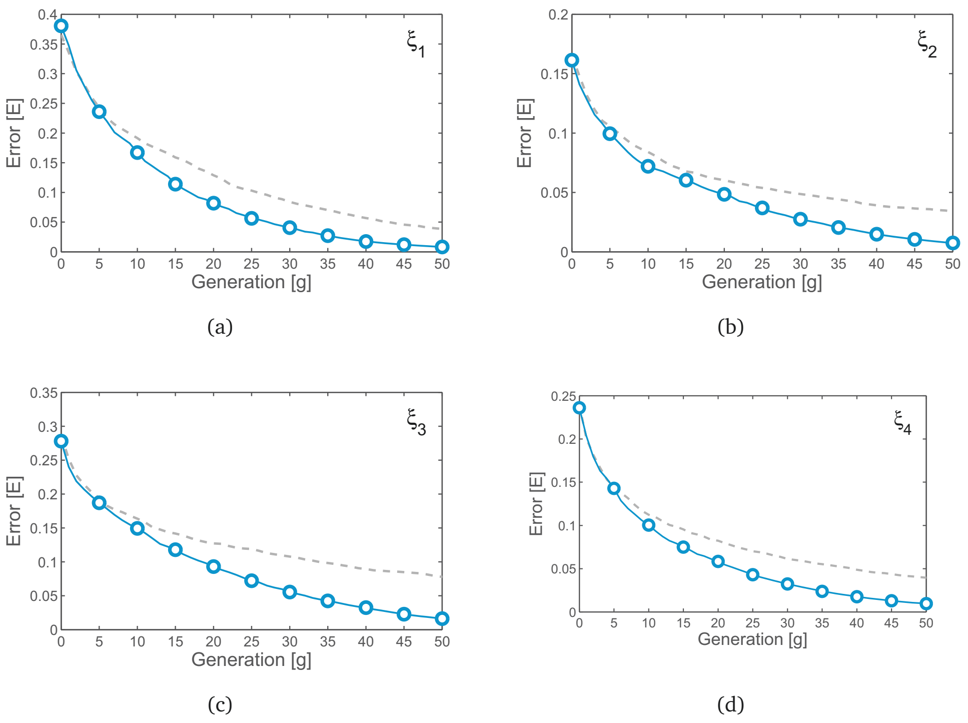


Figure 6.13: Convergence curves of δDE (solid line) and standard DE on the kinematics inversion of four different target pose vector of an anthropomorphic robot manipulator. Curves correspond to the average pose error (E) over one hundred trials for each target point in the testbed, and δDE (10 %, 75 %, 0.3°).

since contrary to the discarding size, modifying σ does not involve additional function evaluations in each generation. Figure 6.13 shows the average convergence curves of the position and orientation error obtained for each point.

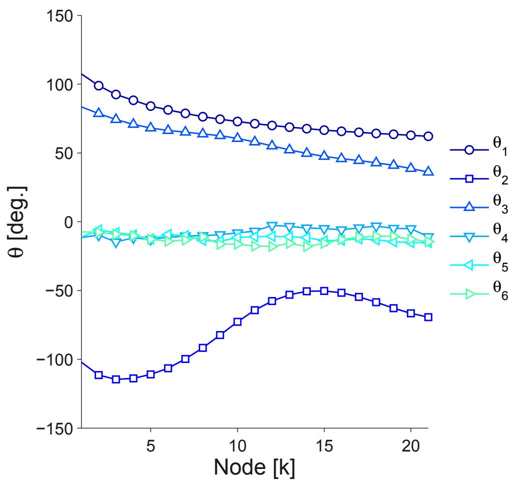
Excessive exploitation also has a negative impact on the efficiency of the algorithm, as observed for test points ξ_3 and ξ_4. Further study showed that convergence failures occurred due to premature convergence to local minima. This could be partly explained by the fact that there exist out-of-bounds solutions to the inverse kinematics close to frontier of the feasible solution space delimited by the joint limits (constraints). The population is driven towards these suboptimal regions by the exploration mechanisms, but it gets stuck in the local minima due to the rapid decrease of population diversity caused by the intensive exploitation of discarding. These empirical results imply that exists a critical dependency on the tuning of discarding parameters as the dimensionality and complexity of the landscape increase.

6.4.3 Path Generation

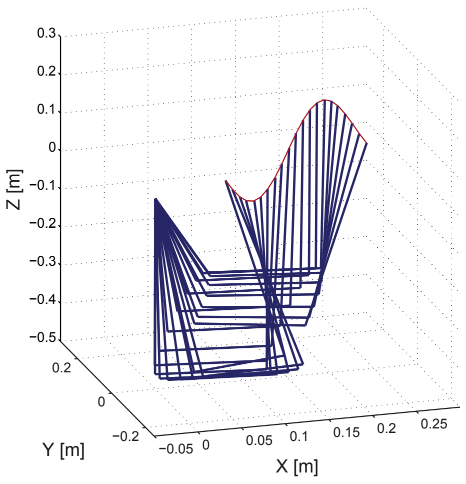
A virtual target path was defined within the robot workspace by sampling N_k consecutive points from the following sinusoidal parametric equation

$$p_z = A \sin\left(B\sqrt{p_x^2 + p_y^2}\right)$$

with with A = 0.1m and B = 1350°/m. The path generation problem consisted of finding the joint configuration of each target node with a tolerance of 1mm in position error. In addition, the robot was assumed to be initially at rest position with all of its joints set to zero, i.e., $\mathbf{q} = (0°, 0°, 0°, 0°, 0°, 0°)^\top$. For this simulation environment the parameter configuration of δDE was set to (10 %, 75 %, 0.3°). Figure 6.14 shows the results obtained for the path generation problem after a single run of δDE.



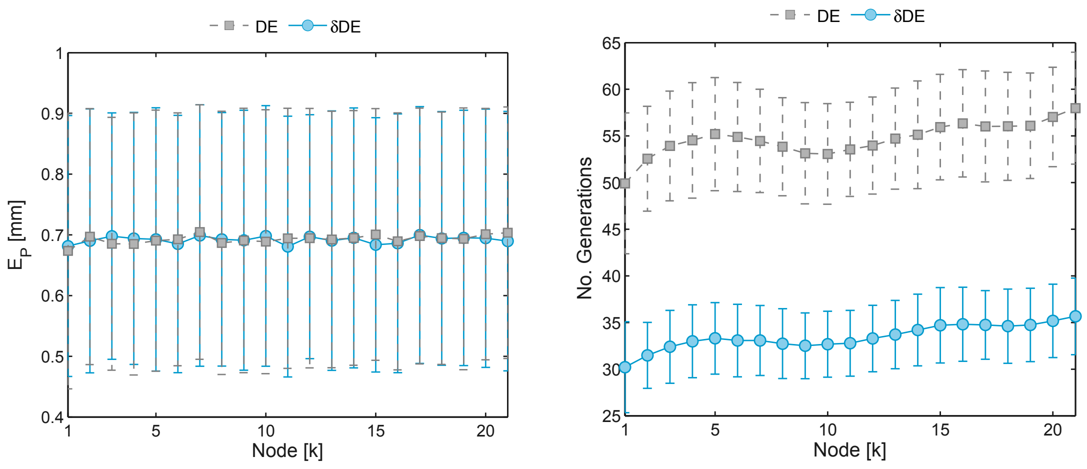
(a) Joint path.



(b) 6 DOF robot manipulator.

Figure 6.14: δDE solution to the path generation problem of an antropomorphic robot manipulator.

After N_t independent runs the average of the total position error obtained for the path was of 0.145mm, which gives an average of 6.90E-03mm of error in each node. These values are similar to the errors obtained with standard DE. However, δDE was able to find an overall solution for the path after a total of 702.87 generations in comparison to the 1148.40 generations required by DE. This represents an average



(a) Average position error. (b) Average number of generations.

Figure 6.15: Comparison of the accuracy and convergence speed of standard DE (dashed line) and δDE (solid line) at each node of a sinusoidal target path by an anthropomorphic robot manipulator. (a) Position accuracy. (b) Number of generations for convergence.

acceleration rate of 38.8 % (see Figure 6.15). This shows that δDE provides a balanced trade-off between convergence speed and accuracy.

6.5 Redundant Robot Manipulator

In this section, we consider the inverse kinematics problem of a redundant robot manipulator with seven degrees of freedom. The Mitsubishi PA10-7C is a general-purpose robot manipulator widely used in demanding tasks, such as surgical tool placement or teleoperated soft tissue manipulation, thanks to its backdrivability, accurate positioning and zero backlash. The arm's serial kinematics structure features seven degrees of freedom (7 DOF) provided by seven revolute joints, arranged as shown in the Figure 6.16. The redundancy entailed by the seventh joint affords the robot with additional dexterity to meet secondary tasks, such as obstacle and singularities avoidance.

The joint configuration vector is defined as $\mathbf{q} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)^\top$ where θ_j represents the angular displacement of joint j. Accordingly, the end-effector position is described by the rectangular Cartesian coordinates $\mathbf{p} = (p_x, p_y, p_z)$ and the orientation by the minimal set of Euler angles $\Phi = \{\phi, \gamma, \psi\}$. Kinematic D-H

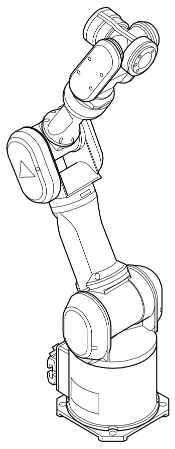


Figure 6.16: PA10 redundant robot manipulator with revolute joints (7 DOF).

Table 6.9: Kinematic parameters and joint limits of the PA10 redundant robot manipulator.

Joint	$\mathbf{a}_j$	α_j	$\mathbf{d}_j$	θ_j	$\mathbf{q}_j^{(lo)}$	$\mathbf{q}_j^{(hi)}$
1	0m	-90°	0.3170m	θ_1	-180.00°	180.00°
2	0m	90°	0m	θ_2	-101.05°	101.05°
3	0m	-90°	0.4500m	θ_3	-180.00°	180.00°
4	0m	90°	0m	θ_4	-153.73°	153.73°
5	0m	-90°	0.4800m	θ_5	-270.00°	270.00°
6	0m	90°	0m	θ_6	-180.00°	180.00°
7	0m	0°	0.0700m	θ_7	-360.00°	360.00°

parameters and joint limits of the PA10 robot arm are listed in Table 6.9.

The inverse kinematics of redundant robot manipulators has infinite solutions, since the number of independent joints ($N = 7$) exceeds the number of task variables in the Cartesian space ($m = 6$). Conventionally, the strategy followed to solve the redundant inverse kinematics problem involves formulating an optimization problem by defining additional constraints in the configuration or task space of the robot. To take full advantage of the redundancy these constraints are usually defined to enhance the dexterity of the robot during the execution of task, e.g., collision avoidance with static or dynamic obstacles. Yet multiple solutions would still exist to any target end-effector pose. Since our aim here is to examine the performance of the proposed memetic approach in comparison to the standard DE scheme on a highly multimodal search landscape, only the joint limits were considered as constraints on the configuration space.

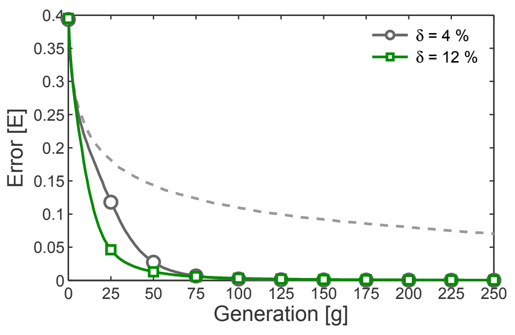
6.5.1 Workspace

In this section, we test the versatility of the proposed optimization-based formulation of the inverse kinematics, and the performance of two differential evolution schemes in finding a feasible solution. The convergence features of both search schemes are examined as kinematics inversion methods by solving the inverse kinematics of one thousand target pose coordinates randomly sampled from the robot's workspace.

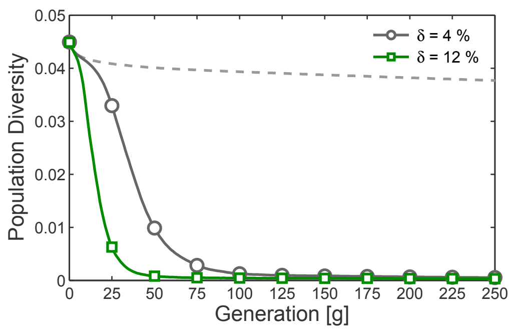
Multimodal optimization problems demand robust diversification mechanisms to explore the search space efficiently and to avoid local minima; however, a mainly exploratory behaviour also slows the convergence of the algorithm. To speed up the convergence it becomes necessary to meet a trade-off between exploring different regions of the optimization landscape and exploiting promising ones. Successfully meeting this balance is particularly challenging in highly multimodal optimization problems as the inverse kinematics of redundant robot manipulators.

We compare the performance of the standard DE and different parameter settings of δDE to highlight the aforementioned features of the search. On the one hand, the slow but robust exploratory behaviour of standard DE; and on the other hand, the faster but less reliable intensification effect provided by discarding to δDE. The latter is further investigated by considering different values of the discarding size, with the aim to increase the intensification pressure over the population, and therefore, to accelerate the convergence speed. Two different values of the discarding size were considered in a highly elitist δDE configuration, .i.e., small elite pool size (β = 12 %). To further enhance intensification, the local search length was also set to a small value ($\sigma = 0.1^\circ$). This means that discarding produced densely distributed solutions around the top best elements in the current population. In the first setup a relatively small value of the discarding size ($\delta = 5$ %) was considered to examine the effect of this parameter on the search dynamic. In a second parameter setup this parameter was increased ($\delta = 12$ %) to further enhance the intensification on the search.

Figure 6.17 compares the evolution of the pose error of the best candidate solution and the population diversity during the first two hundred and fifty generations



(a) Pose error (E) vs. generations.



(b) Population diversity vs generations.

Figure 6.17: Comparison of the convergence behaviour of standard DE (dashed line) and δDE (solid lines) as inverse kinematics methods of an redundant robot manipulator. Curves correspond to the average values obtained from one thousand trials over the reachable workspace of the robot. The discarding mechanism was set to different values of the discarding size (δ) to show its impact over successive generations. (a) Pose error. (b) Population diversity.

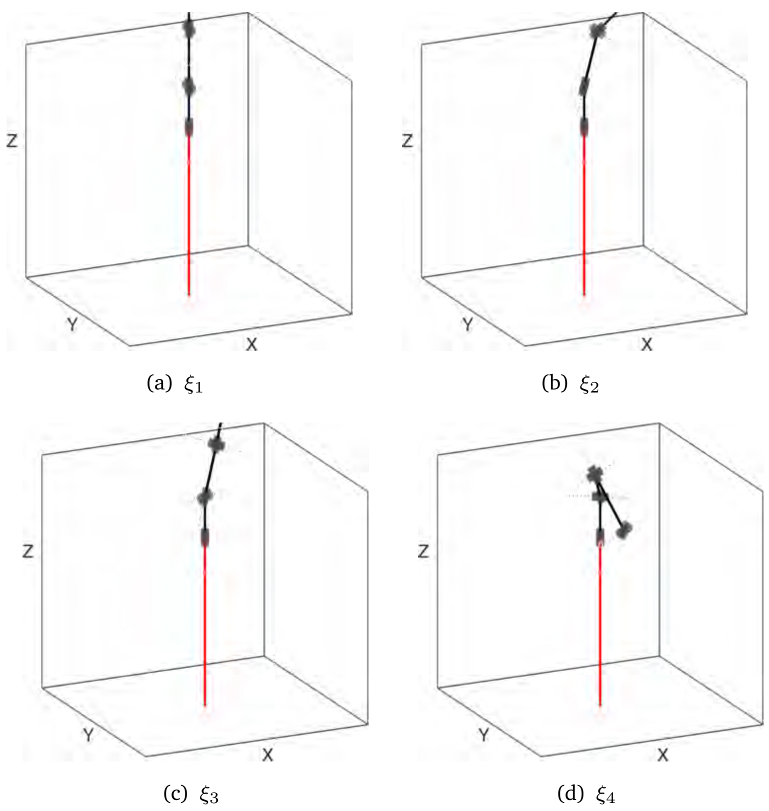
of the search process. The strong exploratory tendency of standard DE becomes evident on the high diversity exhibited by the population over the successive generations, that translates into a slow convergence speed. In contrast, both configurations of δDE considerably increase the convergence speed at expense of drastically decreasing the population diversity.

The average accuracy, convergence speed and success ratio of each algorithm are summarized in Table 6.10. Noteworthy, standard DE completely failed to solve the inverse kinematics of the redundant robot manipulator. By increasing the number of generations to three thousand, standard DE yielded better results in terms of accuracy and success rate, but at the expense of prohibitive slow convergence speed. In contrast, δDE was able to find high-accuracy solutions in position and orientation considerably faster than standard DE, reaching acceleration rates of 88.8 % (δ = 4 %) and 91.60 % (δ = 12 %), respectively.These preliminary results suggest a significant improvement on the overall performance of the search by the discarding strategy in highly multimodal optimization problems.

Table 6.10: Simulation results of standard DE and δDE as kinematics inversion methods over the workspace of the redundant robot manipulator.

	DE[1]	DE[2]	δDE (δ,β,σ)[1]	
			(4 %, 12 %, 0.1°)$_{(1)}$	(12 %, 12 %, 0.1°)$_{(2)}$
E	4.00E-02 ± 2.29E-02(0.0)	1.30E-05 ± 2.62E-04(99.1)	**1.08E-05 ± 1.66E-04(99.7)**	9.84E-05 ± 1.42E-03(99.0)
$\mathbf{E_P}$ (mm)	2.05E+01 ± 1.43E+01(0.0)	1.12E-02 ± 2.57E-01(99.1)	**8.60E-03 ± 1.66E-01(99.7)**	8.06E-02 ± 1.33E+00(99.1)
$\mathbf{E_O}$ (deg.)	6.71E+00 ± 4.85E+00(0.2)	5.07E-04 ± 9.56E-03(99.8)	**7.20E-04 ± 8.49E-04(100)**	5.32E-03 ± 1.41E-01(99.8)
GEN	–	1446.85 ± 428.08	162.00 ± 40.43	**121.59 ± 54.21**
AR (%)	–	–	88.8	**91.60**

[1] $g_{\max}$ = 500.
[2] $g_{\max}$ = 3000.



(a) ξ_1 (b) ξ_2

(c) ξ_3 (d) ξ_4

Figure 6.18: Pose testbed for the 7 DOF PA10 robot manipulator.

6.5.2 Pose Testbed

A simulation testbed with four target pose coordinates has been considered for this simulation experiment. Figure 6.18 shows one of the multiple solutions to the

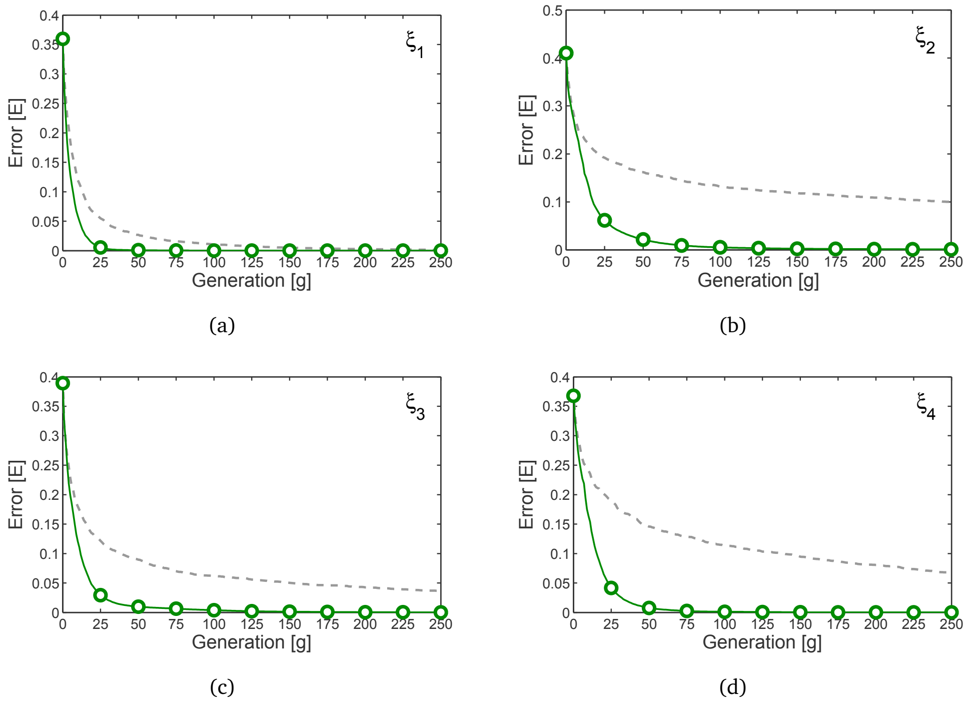


Figure 6.19: Convergence curves of δDE (solid line) and standard DE (dasehed line) on the kinematics inversion of four different target pose vector of a redundant robot manipulator. Curves correspond to the average pose error (E) over one hundred trials for each target point in the testbed, and δDE (12 %, 12 %, 0.1°).

inverse kinematics of each target pose of the end-effector proposed in the testbed. The aim is to asses the repeatability of each algorithm given a highly multimodal optimization problem. Simulation results reported in Table 6.11 correspond to the average value obtained after one hundred independent runs of the DE and different parameter configurations of δDE.

These results are consistent with the convergence behaviour observed in the previous experiment. Once again, after five hundred generations, standard DE failed to converge in almost all runs of the algorithm. The exception occurred in the first test point (ξ_1), that corresponds to the fully extended arm configuration (singularity). In this case, DE was able to find a solution with errors below the predefined thresholds. In contrast, by introducing a simple migration/local search mechanism, δDE was able to provide high-accuracy solutions to the redundant inverse kinematics problem of all target pose coordinates in the test suit.

Table 6.11: Simulation results of DE and δDE as kinematics inversion methods of a seven degrees of freedom robot manipulator.

	DE[1]	**DE**[2]	**δDE** (δ,β,σ)[1]	
			(4 %, 12 %, 0.1°)$_{(1)}$	(12 %, 12 %, 0.1°)$_{(2)}$
ξ_1	$\mathbf{p}_1$ =(0, 0, 1.317)m Φ_1 =(0,0,0)°			
E	4.38E-05 ± 3.36E-05(100)	1.2131E-15 ± 1.0396E-14(100)	8.82E-08 ± 2.93E-07(100)	**2.93E-08 ± 7.35E-08(100)**
E_P (mm)	1.70E-02 ± 1.65E-02(100)	1.2131E-12 ± 1.0396E-11(100)	3.29E-05 ± 1.22E-04(100)	**1.30E-05 ± 3.64E-05(100)**
E_O (deg.)	7.32E-03 ± 6.45E-03(100)	3.4151E-08 ± 2.4026E-07(100)	1.51E-05 ± 5.09E-05(100)	**4.51E-06 ± 1.09E-05(100)**
GEN		361.86±43.71	90.79 ± 15.62	**57.65 ± 20.95**
AR (%)			74.95	**84.33**
ξ_2	$\mathbf{p}_2$ =(0.4877, 0.0458, 0.9880)m Φ_2 =(-140.2346 ,49.6398,132.3117)°			
E	7.16E-02 ± 1.69E-02(0)	1.8540E-05 ± 1.6831E-04(99)	**5.65E-06 ± 4.40E-06(100)**	6.12E-06 ± 1.67E-05(99)
E_P (mm)	3.59E+01 ± 1.56E+01(0)	1.3463E-02 ± 1.2747E-01(99)	**2.71E-03 ± 2.17E-03(100)**	3.31E-03 ± 1.09E-02(99)
E_O (deg.)	1.07E+01 ± 4.68E+00(0)	1.5167E-03 ± 1.2329E-02(99)	8.80E-04 ± 8.56E-04(100)	**8.42E-04 ± 1.81E-03(100)**
GEN	–	1764.59 ± 407.41	161.61 ± 39.47	**137.35 ± 60.55**
AR (%)			90.84	92.22
ξ_3	$\mathbf{p}_3$ =(0.0316, -0.3476, 1.2422)m Φ_3 =(159.8530, -6.0872, -5.7301)°			
E	2.18E-02 ± 4.92E-03(0)	1.2370E-05 ± 9.5972E-05(99)	**5.84E-06 ± 2.79E-06(100)**	6.15E-06 ± 6.12E-06(100)
E_P (mm)	1.14E+01 ± 4.18E+00(0)	5.4412E-03 ± 3.8981E-02(99)	**2.85E-03 ± 1.63E-03(100)**	3.27E-03 ± 3.69E-03(100)
E_O (deg.)	2.88E+00 ± 1.22E+00(0)	1.9136E-03 ± 1.5777E-02(99)	8.27E-04 ± 5.02E-04(100)	**7.94E-04 ± 7.90E-04(100)**
GEN	–	1849.57 ± 418.86	178.93 ± 37.25	**149.35 ± 55.87**
AR (%)			77.35	91.93
ξ_4	$\mathbf{p}_4$ =(-0.0227,-0.4768,0.1346)m Φ_4 =(146.2806,-16.5334,155.3686)°			
E	3.02E-02 ± 1.19E-02(0)	1.2829E-09 ± 5.6540E-09(100)	3.14E-06 ± 2.91E-06(100)	**2.05E-06 ± 1.75E-06(100)**
E_P (mm)	1.58E+01 ± 8.48E+00(0)	8.8368E-07 ± 4.1306E-06(100)	1.69E-03 ± 2.16E-03(100)	**1.05E-03 ± 1.02E-03(100)**
E_O (deg.)	6.43E+00 ± 3.59E+00(0)	1.7783E-07 ± 9.6952E-07(100)	6.47E-04 ± 5.40E-04(100)	**4.47E-04 ± 3.98E-04(100)**
GEN	–	1247.33 ± 285.52	151.39 ± 25.53	**100.98 ± 27.05**
AR (%)			77.11	91.90

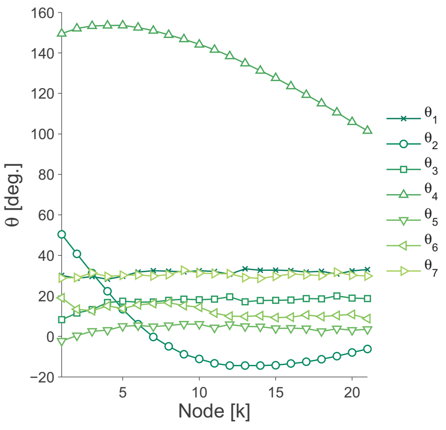
[1] $g_{\max}$ = 500.
[2] $g_{\max}$ = 3000.

Moreover, δDE was significantly faster than standard DE while still achieving high-accuracy solutions. More importantly, these simulation results verify the almost proportional relationship between the discarding size and the convergence speed, but also its impact on the robustness of the algorithm. Thus, comparing the average number of generations required to reach the error thresholds respect to standard DE, δDE provided an average of 80.06 % of acceleration with a 4 % of discarding size, while a 90.09 % was obtained increasing the number of discarded solutions up to 12 % of the population size in each generation. Figure 6.19 illustrates the acceleration effect of discarding ($\delta = 12$ %).

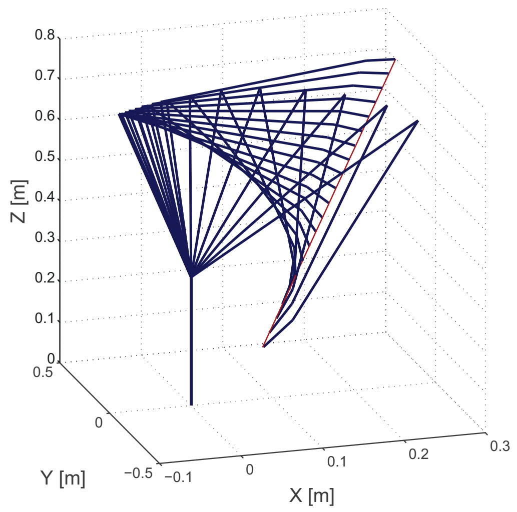
6.5.3 Path Generation

In this section, the performance of the proposed δDE kinematics inversion scheme was examined for the path generation problem of a redundant robot manipulator. The commanded task consisted of following a straight line segment, defined in the robot's reachable workspace as a set of consecutive desired position coordinates ($N_k = 21$), with initial and final position at $\mathbf{p}_0 = (0.1, 0.1, 0.1)$m and $\mathbf{p}_{N_k} = (0.3, 0.4, 0.7)$m, respectively. It was also assumed that the robot was initially at rest position with all of its joints at zero degrees.

This path generation problem was solved sequentially for each node by minimizing the end-effector position error and the joint displacement from the previous node. The results obtained after one hundred independent runs of δDE (12 %, 12 %, 0.1°) provided an average total position error of 14.29mm, which represents an average error of 0.6803mm per node on the path. Moreover, for the same task δDE was able to find a solution 68.60 % faster than standard DE, with an average total of 651.42 generations in comparison to the 2074.43 required by DE. Figure 6.20 shows the solution joint path for the 7 DOF redundant robot manipulator, and Figure 6.21 illustrates the distribution of the position error and the number of generations along the target path.

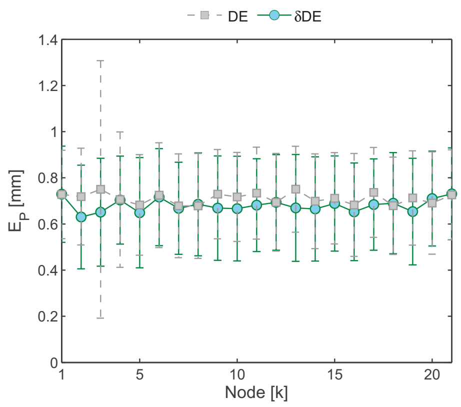


(a) Joint path.

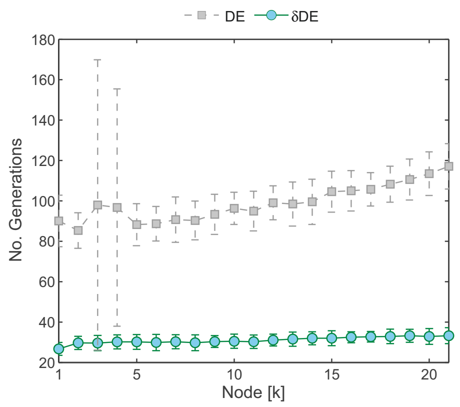


(b) 7 DOF robot manipulator.

Figure 6.20: δDE solution to the path generation problem of a 7 DOF robot manipulator.



(a) Average position error.



(b) Average number of generations.

Figure 6.21: Comparison of the average error and number of generations of standard DE (dashed line) and δDE (solid line) along a straight line path.

6.6 Computational Cost

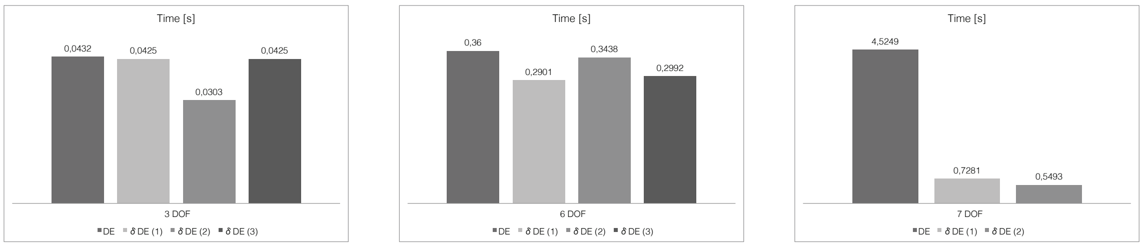
The previous simulation results showed the positive impact of discarding on the convergence speed of standard DE. Thus far, these improvements have been expressed in terms of the number of generations required to reach a predefined objective function value. Nonetheless, a reduction in the number of generations might not necessarily yield a lower computational cost, if it does not entail a reduction on the number of function evaluations.

In general, the evaluation of the objective function demands most of the computational effort in each generation of the evolutionary process. This holds particularly true for the inverse kinematics problem, since the evaluation of the pose error for each candidate solution involves a two-step procedure, namely mapping via forward kinematics and computing the pose error. Considering that the discarding mechanism increases the number of function evaluations in proportion to the discarding size (δ), we can directly deduce that a single iteration of δDE is actually computationally more expensive than one of the standard DE.

With the aim of estimating the computational cost of δDE for the inverse kinematics problem, the execution time was measured in all the simulation experiments. These measurements are based on simulations run on a MacBook Pro 2.5 GHz Intel Core I5. The additional computational cost introduced by discarding was computed considering the worst case scenario, in this case, the largest value of discarding size (δ) for each robot.

Taking as reference the execution time of standard DE for a fixed number of generations and neglecting the initialization time, it was estimated that the cost in seconds associated to the discarding in each generation was of 0.230ms ($\delta = 6$ %), 0.863ms ($\delta = 10$ %), and 0.996ms ($\delta = 12$ %) for the three, six, and seven DOF robot manipulators, respectively. This represents a total increase in time of approximated 35 %, 52 %, and 29 % respect to the standard DE for the same number of generations.

The improvement in time carried by a decrease in the number of iteration was also computed. For this, the best result obtained in convergence speed for each robot manipulator was compared to the time of convergence of standard DE. Results show that, although the acceleration rates obtained with δDE were outstanding



(a) 3 DOF planar robot arm. (b) 6 DOF robot manipulator. (c) 7 DOF robot manipulator.

Figure 6.22: Time of convergence of standard DE and δDE for the inverse kinematics problem.

in terms of number of generations, the gains in time are rather modest. Thus, an acceleration rate of 49.33 % only represents an improvement of 1.6 % for the planar robot, and a 42.74 % translate into a 16 % for the PUMA arm. The exception occurred with the 7 DOF robot manipulator since that a 91.60 % of acceleration rate, provided a gain in time of approximately 88 %.

Figure 6.22 shows the computational time required for convergence (error threshold) by standard DE and δDE for the inverse kinematics of robot manipulators. In this figure δDE (1), δDE (2), and δDE (3) correspond to parameter settings used in the previous sections to test the performance of δDE in the order that they appear (from left to right) on the Tables 6.4, 6.7, and 6.10.

6.7 Summary

In this chapter we have presented a simulation-based study on the performance of the memetic differential evolution scheme (δDE) as kinematics inversion method. δDE combines the powerful exploration features of standard DE search scheme with a local search operator to enhance its convergence speed rate. Based on the simulation experiments performed on three benchmark robot manipulators it can be summarized that:

- The optimization-based approach provides a flexible framework for the inverse kinematics problem. It readily admits redefining the optimization criteria according to the commanded task.

- The memetic approach exhibits an overall satisfactory performance for the inverse kinematics problem of robot manipulators. Results revealed a consistent convergence behaviour on different robotic platforms and degrees of freedom.
- Discarding provides an efficient local refinement strategy without disrupting the overall global performance of standard DE.
- In general, δDE generates inverse kinematics solutions with lower average error values in position and orientation than those obtained with DE.
- δDE yields a substantial improvement on the convergence speed rate in terms of the average number of iterations required to reach a desired end-effector accuracy. The replacement strategy of discarding leads the population to rapidly group around the global minimizers.
- A trade-off among accuracy, success, computational cost and convergence speed hinge on a fine tuning of control parameters δ, β, and σ.
- Although increasing the value of δ significantly upgrades convergence speed, this might also be disadvantageous since excessive exploitation might lead to premature convergence.
- Large values of δ proportionally increases the number of function evaluations in each iterations and consequently the actual computational cost.

Chapter 7

δDE Control Parameters

The convergence performance of δDE hinges on the well-balanced behaviour between exploration and exploitation of the search space. As it can be easily deduced from the simulation results obtained in the previous chapter, control parameters play a key role at modifying the diversification and intensification features of the search, according to the requirements of the objective function landscape.

Nonetheless, parameter setting is not a straightforward task and by itself represents a multiobjective optimization problem. Moreover, as the number of parameters grows, it becomes more difficult to keep track on the individual influence of each parameter over the search dynamic. In practice, the parameter setting is usually approached as a trial and error process based on either reference values suggested by benchmark studies, or ad-hoc empirical data. In the best scenario, this process would provide a sufficiently good trade-off in accuracy, efficiency, and convergence speed; but more often, further tuning would be necessary to adapt the parameters to the requirements of the optimization problem in hand.

δDE has six different control parameters, namely population size (N_P), mutation scale factor (F), crossover rate (C_R), discarding size (δ), elite pool size (β), and local search length (σ). The first three represent diversification parameters since they control the exploratory search scheme; whereas the last three are intensification parameters that regulate the discarding mechanism. All control parameters are constant values set by the user.

In this chapter, we address the parameter setting of δDE for the inverse kinematics problem of robot manipulators, based on empirical data obtained from simulation experiments. The aim is to gain further insight on the influence of control parameters on the convergence behaviour of δDE, rather than to provide a set of rules for their tuning. Otherwise, it would be necessary a thorough testing on a wider range of functions, which is out of the scope of this dissertation.

The study is divided in two parts. In the first part we have focused on the discarding parameters, by independently varying their values while the diversification parameters are set constant. The idea is to examine how much improvement (or not) can be obtained on the performance of the algorithm on the basis of a rough tuning of DE parameters. In the second part, the setting of all parameters are treated as a direct search problem, such as to obtain an (almost) optimum convergence performance for the inverse kinematics of a redundant robot manipulator.

7.1 Discarding Parameters

In this section, we seek to investigate the role that each of the discarding parameters plays in the convergence behaviour of δDE as kinematics inversion method. The study has a dual purpose: (i) getting a snapshot of the expected convergence behaviour of δDE when the intensification pressure varies, and (ii) quantifying the amount of improvement that can be obtained on the performance of the algorithm when the diversification parameters are roughly tuned. The discarding parameters and their operating values are briefly summarized as follows:

- **Discarding Size (δ).** The discarding size or discarding rate represents the number of candidate solutions that are discarded in each generation, and therefore, it controls the migration rate of δDE. For uniformity on the analysis across different applications, this parameter is usually defined as a percentage value of the population size rather than as an exact number of discarded solutions in each generation. For its analysis, we have considered values that range between zero and fifty percent of the total population size. Note that a zero percent of discarding rate implies that none of the candidate solutions are discarded whatsoever, i.e., the standard DE scheme.

- **Elite Pool Size (β).** The replacement strategy of the discarding mechanism is based on the local refinement of candidate solutions sampled from a pool with the best fitted individuals. The elite pool size (β) represents the number of candidate solutions contained therein this elite sub-population. Thereby, small values of β promote an elitist local search, whereas large values increase the diversity of the seed pool. Further, to avoid excessive exploitation, this parameter is usually set greater than the discarding size (δ); however, for the purposes of this study this condition has been relaxed to allow different sampling scenarios. To assess the influence of the elite pool size on the search, its value has been conveniently varied from $(1/N_P) * 100$ to $(100 - \delta)$ percent of the population size. That is, we have considered highly elitist scenarios, in which only the best candidate solution is used as seed for replacement, as well as highly diverse scenarios, in which almost all individuals in the population belong to the replacement parent pool.

- **Local Search Length (σ).** Discarding implements a local search mechanism based on the Solis-Wets local improvement process. For it, we assume that every candidate solution in the population represents the mean vector of a D-variate normal random distribution with a common standard deviation σ. Therefore, the local search mechanism of discarding is just a sampling process modelled by a normal random function, whereby new solutions are generated in the vicinity of existing ones. In this scheme, the parameter σ determines the local distribution of the new candidate solution around the mean vector, and therefore, it controls the local search length. To study its influence on the search dynamic of δDE, we have considered values of σ that range from zero to twenty degrees. Note that when the value of σ is set to zero, the discarding mechanism transforms into a migration-only operator.

The role of each of the discarding parameters was examined in independent simulation tests with the aim of keeping track of the individual impact of their values on the convergence behaviour of δDE. The simulation test was designed so that only one discarding parameter was iteratively varied while the other two and the diversification parameters, namely the population size, mutation scale factor, and crossover rate, were kept constant at their reference values (see Table 7.1). To provide a broad assessment on the performance of the memetic algorithm under

Table 7.1: Reference values of the δDE control parameters.

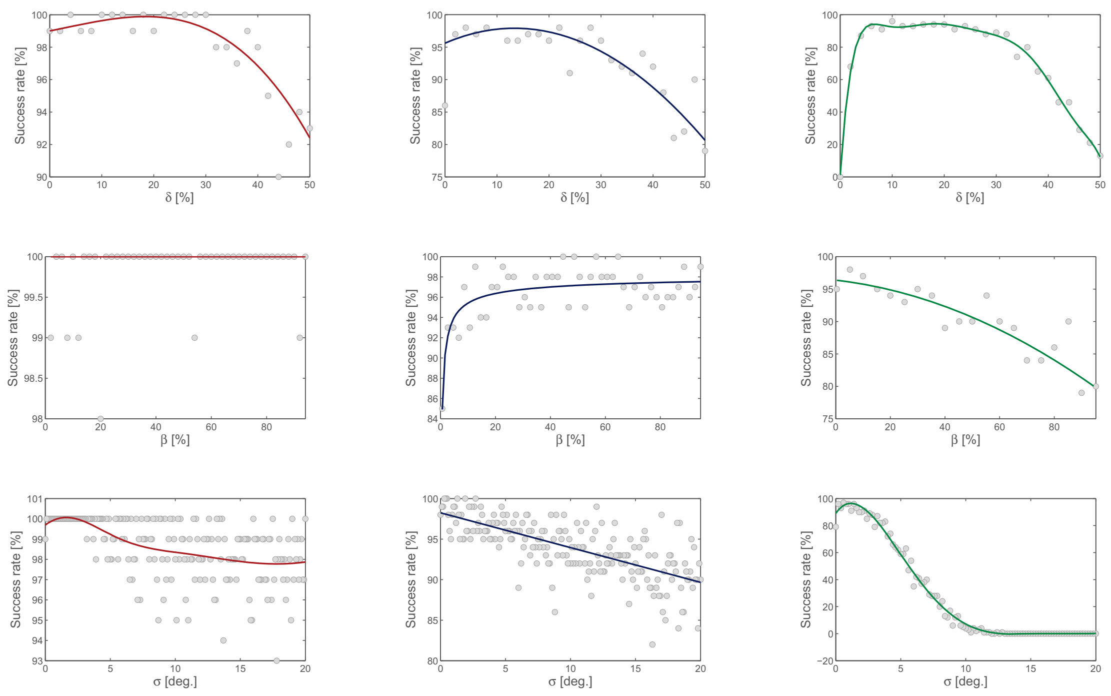
DOF	F	C_R	N_P	δ	β	σ	$g_{\max}$
3			50				100
6	0.5	0.8	150	5 %	50 %	2°	300
7			250				500

different parameter settings, each iteration consisted of one hundred independent trials. At each trial, the inverse kinematics was solved for a target pose randomly sampled from the robot reachable workspace. The (near-) optimal solution was then obtained as the joint configuration vector that yielded the minimum pose error after that a maximum number of generations had elapsed. For each parameter setting (iteration), the overall performance of the algorithm was evaluated based on the average values of the accuracy, success rate, and convergence speed obtained from these trials. To further examine the scalability of this analysis on the dimension of the search space, the tests on the discarding parameters were repeated for three different robot manipulators with three, six, and seven degrees of freedom, respectively.

7.1.1 Success Rate

The success rate of δDE was estimated as the ratio of trials that converge to the true optimal solution. Inasmuch as the resolution of sensors and the control system of robot manipulators impose a maximum end-effector positioning accuracy, the optimality convergence condition was relaxed to accept near-optimal solutions for the inverse kinematics problem. In this regard, the convergence of any trial was considered a success if the inverse kinematics solution afforded a position and orientation error below a predefined tolerance error value; otherwise, the trial was considered a failure. Although in practice the positioning accuracy varies among robot manipulators, for the sake of the comparison, the error tolerance was set to 0.1mm in position and 0.1° in orientation for all robots.

Figure 7.1 illustrates the effect of the discarding parameters on the search efficiency of δDE as kinematics inversion method. Simulation results are represented for each control parameter by the scatter plot of the estimated criteria and a fitted trend curve (solid line) that describes the overall tendency of the average. Results



(a) 3 DOF planar robot manipulator. (b) 6 DOF PUMA robot manipulator. (c) 7 DOF PA-10 robot manipulator.

Figure 7.1: Success rate of δDE as a function of the discarding parameters: discarding size (top), elite pool size (middle), and local search length (bottom).

can be summarized as follows:

δ The response to the discarding rate can be divided into three well differentiated regions. The first encompasses values of the discarding rate lower than 10 %, the second spans between approximately 10 % and 30 %, and the third corresponds to values greater than 30 %. In the first region, the efficiency of the search steadily improves as the migration rate increases from zero (i.e., δDE without discarding). Around 10 %, the curve reaches its maximum value and then it flattens, that is, any further increase below 30 % neither significantly enhances nor worsen the success rate of δDE. Finally, a noticeable decline on the success rate is observed for values greater than 30 % of the discarding rate.

β Simulation results show that the response to the elite pool size is more dependent to the problem landscape than the other discarding parameters. With the exception of the inverse kinematics of the planar robot manipulator, increasing the value of the replacement pool size has contrasting effects on the performance of the algorithm. On the one hand, for the non-redundant PUMA robot arm the convergence performance noticeable improves as the diversity of the replacement pool increases. On the other hand, the inverse kinematics problem of the redundant PA-10 robot demands a more elitist replacement strategy with lower values of β.

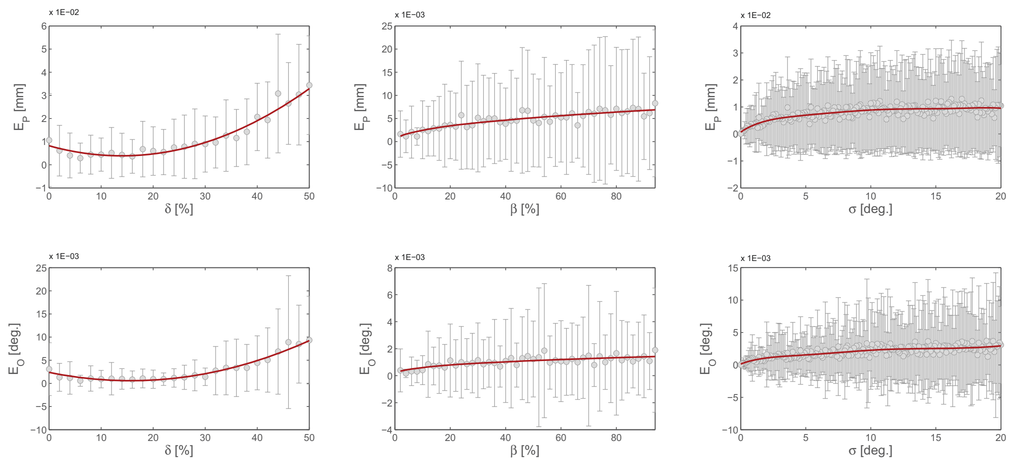
σ A steady improvement on the success rate is obtained with small values of standard deviation; however, this is followed by a marked decline as its value increases. According to the simulation data the best results are obtained with values below 5° .

Having regard the characteristics of the inverse kinematics problem, these results suggest that as the dimension of the search space grows, a good performance of δDE is obtained increasing the intensification pressure but with a lower number of migrant candidate solutions.

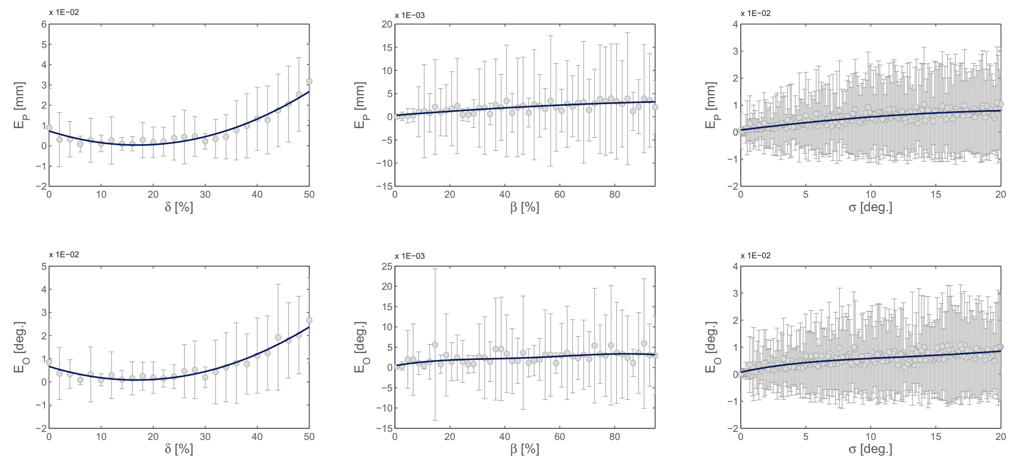
7.1.2 Accuracy

The accuracy of the algorithm was defined as the position and orientation error yielded by the solution in each trial. Aside from analysing the overall performance in accuracy of each parameter setting, we were interested in studying the convergence behaviour of the algorithm related to its success ratio. Therefore, for the purpose of the analysis, solutions were classified as "success" or "failure" depending on whether or not their position and orientation errors fell below the predefined tolerance accuracy values (0.1mm and 0.1°).

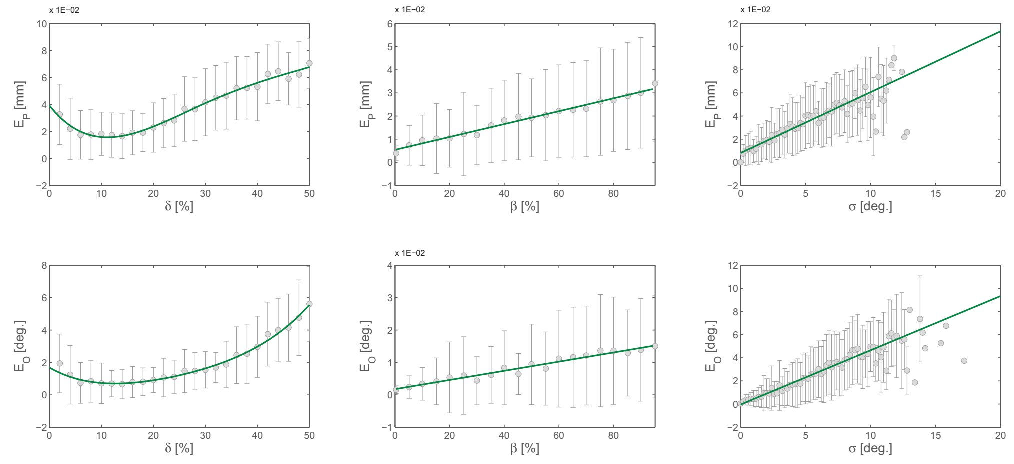
Figure 7.2 illustrates the distribution and orientation error of the "success" optimal solutions found for different settings of the control parameters. Based on the simulation data it can be highlighted that:



(a) 3 DOF planar robot manipulator.



(b) 6 DOF PUMA robot manipulator.



(c) 7 DOF PA-10 robot manipulator.

Figure 7.2: Average position and orientation errors of "success" trials as a function of the discarding parameters: discarding size (left), elite pool size (center), and local search length (right).

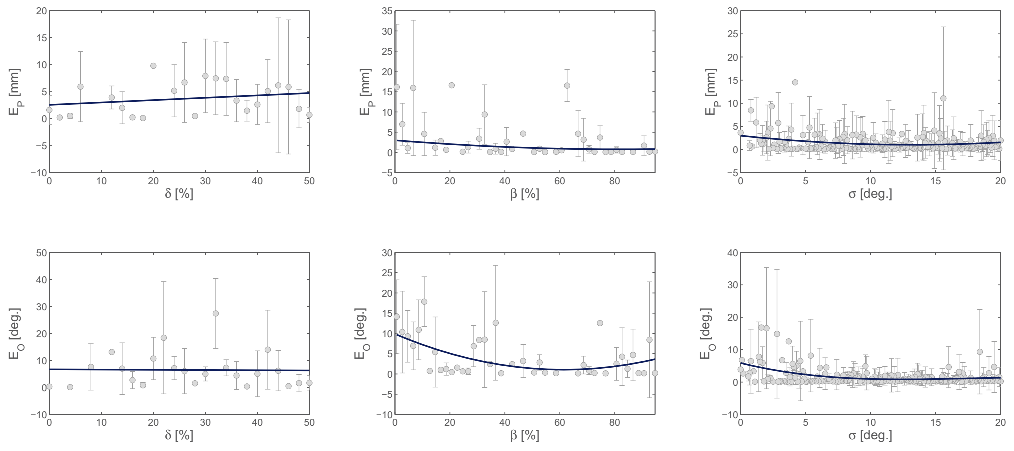
δ The error smoothly decreases, in average and standard deviation, until it reaches a minimum value around 15 % – 20 % of discarding rate. For values above this range, migration disrupts the exploration of the search space and the error starts increasing. The U-shaped curve described by the accuracy and success rate responses, suggest that the amount of improvement that can be obtained with migration strategies is limited.

β, σ The error grows linearly and become more sparsely distributed as their values increase. In the case of the 7 DOF redundant robot manipulator the linear tendency shows a steep slope, since the high-multimodality of the inverse kinematics problem demands small values of both parameters. Note that for values of σ greater than 15° no data is available since the success rate of δDE drops to zero (see Figure 7.1(c)).

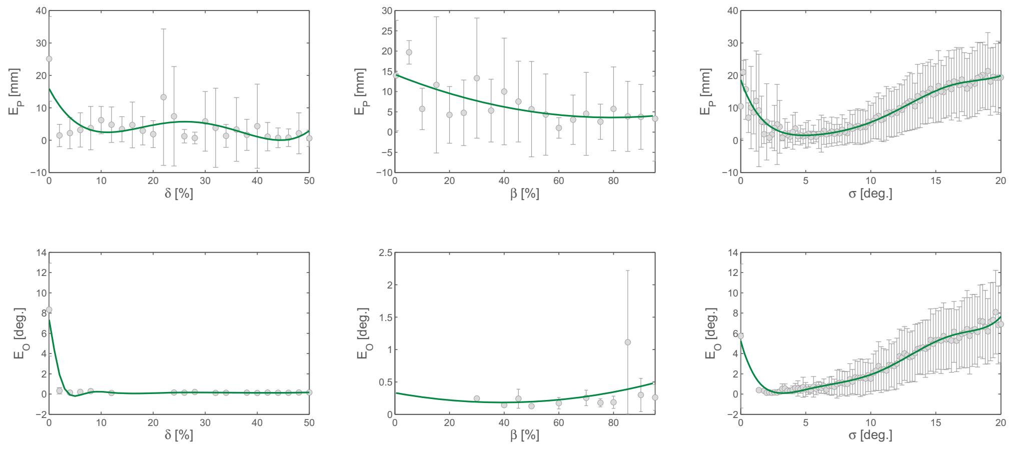
With the aim to understand how the algorithm fails, and therefore, to gain useful insight on the convergence behaviour of δDE under different parameter settings, the average errors of the "failure" solutions were also examined. Figure 7.3 illustrates the average position and orientation errors of the solutions that failed to converge according to the considered accuracy tolerance value. In this figure we have omitted the results obtained for the kinematics inversion of the 3 DOF planar robot manipulator, since in this case, it did not provide significant data for the analysis. Therefore, for each control parameter the results were the following:

δ As the value of the discarding rate increases, the algorithm fails more often and with larger errors, particularly in position. In fact, for values greater than 20 % the behaviour of δDE is considerable poorer in comparison to the standard DE ($\delta = 0$ %). The exception occurs with the orientation error of the redundant robot manipulator, but not with its position error.

β The overall effect of increasing this value exhibits a decreasing trend; however, the average error remains considerably above the tolerance threshold.



(a) 6 DOF PUMA robot manipulator.



(b) 7 DOF PA-10 robot manipulator.

Figure 7.3: Average position and orientation errors of “failure” trials as a function of the discarding parameters: discarding size (left), elite pool size (center), and local search length (right).

σ A distinct behaviour can be observed in the non-redundant and redundant inverse kinematics. Although, in the non-redundant case the average error curve shows a decreasing tendency for small values of σ, the error is almost constant for values greater than 10°. Conversely, in the redundant case the average error decreases until it reaches a minimum value and then increases almost linearly. In both, for relatively small values of σ, the local search replacement strategy yields a better performance than the migration-only scheme set by a standard deviation of zero degrees.

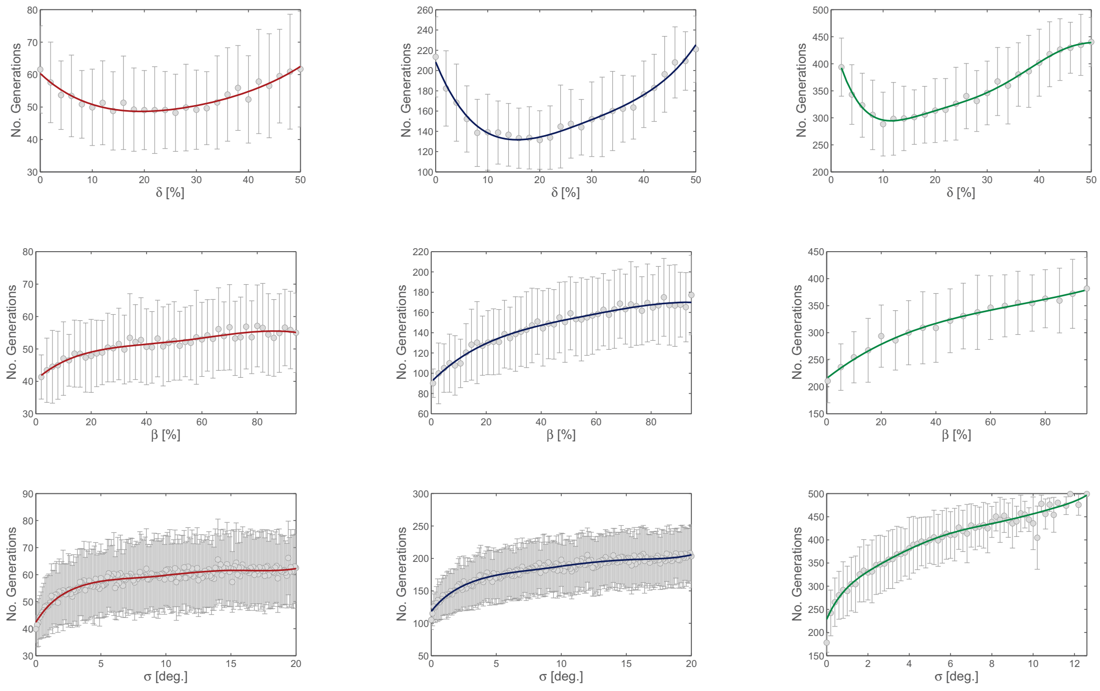
In terms of accuracy, large values of the three discarding parameters have a negative impact on the overall performance of δDE. For instance, by increasing the number of discarded solutions, the algorithm would not only be discarding poor solutions (low fitness), but also promising elements with intermediate fitness values. The derived loss of population diversity leads to convergence failure due to the disruption of the exploration mechanisms. Moreover, large values of β promote diversity in detriment to a more focused local search. Similarly, large values of σ also distort the base of the local search.

7.1.3 Convergence Speed

The convergence speed is regarded in terms of the average number of generations required to reach a (near-)optimal solution. Figure 7.4 shows the behaviour of the convergence speed for different parameter settings of the memetic algorithm:

δ Simulation results have supported the hypothesis of the inverse proportional relationship between the discarding rate and the number of generations required to reach a predefined accuracy tolerance value. However, according to these results, this would only hold for values lower than approximately 20 % of discarding rate. For greater values, the number of generations increases almost quadratically. In the case of the redundant robot manipulator, the maximum improvement occurs around five percent of discarding rate, but the performance deteriorates slower than in the non-redundant robot manipulators.

β, σ The diversification role of these parameters has the effect to slow down the search process as their values increase, and consequently, does the number of generations required for convergence (see Figure 7.4(a)).



(a) 3 DOF planar robot manipulator. (b) 6 DOF PUMA robot manipulator. (c) 7 DOF PA-10 robot manipulator.

Figure 7.4: Convergence speed of δDE as a function of the discarding parameters: discarding size (top), elite pool size (middle), and local search length (bottom).

Simulation results verify the main role of the discarding rate in the improvement of the convergence speed of δDE, although its effect on the algorithm is limited to relatively small rate values.

7.2 δDE parameters

Thus far we have advocated for the superior performance of δDE over the standard DE. This claim have been based on the belief that the DE lacks of efficient exploitation mechanisms, and instead, it excessively promotes an exploratory behaviour of the search. This deficiency is further enhanced by the large values of the mutation scale factor and crossover rate usually demanded by multimodal objective functions. The δDE memetic approach attempts to overcome this deficiency by transferring the exploitation task onto an additional local search mechanism (discarding). This approach increases the complexity of the search dynamic of differential evolution, but also provides flexibility in the configuration setting of the search.

Considering that the memetic algorithm is the result of combining the exploratory strategy of standard DE with the intensification mechanism of discarding, two forthright questions arise: (i) Would discarding have a significant impact on the search behaviour when standard DE parameters are well-tuned?, and if so, (ii) is it its implementation computationally worth it?.

The study in the previous sections shed some light on the convergence features of δDE when the intensification pressure on the search varies. In addition, it provided useful intuition on how much improvement can be obtained on the accuracy and convergence speed on the basis of roughly tuned diversification parameters.

Nonetheless, to be able to understand the true potential of discarding as acceleration mechanism, it is necessary to address the parameter setting of δDE by considering all of its control parameters. That is, finding a parameter setting that provides an almost optimal convergence performance for the inverse kinematics problem. The parameter setting is thus conceived as a multiobjective optimization problem, that is, the optimal solution must simultaneously satisfy multiple conflicting criteria, for example: high accuracy, good convergence speed, and robustness. That means finding Pareto optimal solutions that set a trade-off among these conflicting criteria according to which aspect of the performance one needs to emphasize.

Although there exists an extensive literature devoted to multiobjective optimization, in this section the parameter setting problem is tackled with a simpler approach. Basically, it is supported on the direct testing of δDE over all possible combinations of a discrete set of control parameters values. The parameters can then be chosen according to different performance criteria in an *a posteriori* analysis of the results.

The study on control parameters of δDE has been divided in two parts. The first focuses on the diversification parameters, namely population size, mutation scale factor, and crossover rate. The second part is devoted to the intensification parameters: discarding rate, elite pool size, and local search length, supported by the results obtained in the first part for the diversification parameters. Since in terms of the simulation time performance the major disparities between standard

DE and δDE have been observed in the solution of the inverse kinematics of the PA-10 robot manipulator, the following study on the control parameters will be solely restricted to the case of the redundant seven degrees of freedom robot manipulator.

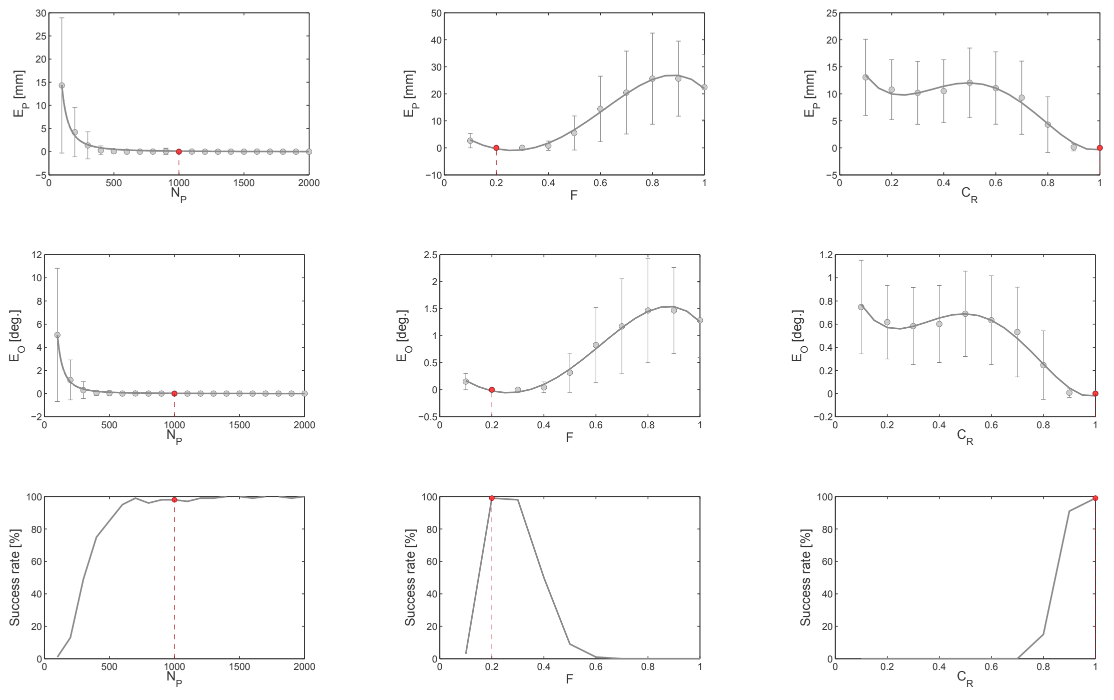
7.2.1 Diversification Parameters

Let us first consider the diversification parameters of δDE, that is, the population size (N_P), mutation scale factor (F), and crossover rate (C_R). These are regarded as diversification parameters since their values regulate the variation mechanisms of δDE, and therefore, are related to the exploratory behaviour of the search. These are also the original control parameters of the `DE/rand/1/bin` search scheme, upon which the δDE memetic algorithm is based on.

The control parameters of δDE, including the ones that regulate discarding, are constant positive values set by the user. The population size (N_P) is usually defined based on the dimension of the search space (D), such that, depending on the attributes of the objective function its value would typically vary between $2D$ and $100D$. By definition, the mutation scale factor (F) can take any value greater than zero; however, in practice, values greater than one are rarely used. In contrast, the crossover rate (C_R) should be set less than or equal to one.

Here we seek a feasible set of diversification parameters for the inverse kinematics problem that yield the *best* performance of δDE without discarding, i.e., the discarding rate set to zero. In this context, by *best* performance we understand the lowest pose error and highest success rate for a given error threshold and a maximum number of generations. Alongside with these requirements, the parameters should be finally set to the lowest possible value, particularly the population size.

The process of finding the best set of diversification parameters for the inverse kinematics problem consisted in a two-step procedure. First, the parameters were varied within a feasible range of values in a nested loop, such that, every possible combination among a discrete set was tested. In this step, the performance of the algorithm was evaluated based on the results of the average pose error and overall success rate, obtained from one hundred independent simulation trials run on the inverse inverse kinematics problem. The preliminary parameter set obtained from the first step underwent a second round of testing to further asses their robustness. Thereby, one parameter was varied around its preliminary *best* value while keeping



(a) Population size (N_P) (b) Mutation scale factor (F) (c) Crossover rate (C_R)

Figure 7.5: Accuracy and success rate behaviour of δDE without discarding as a function of the diversification parameters: population size (left), mutation scale factor (center), and crossover rate (right).

the other two parameters unaltered. Contrary to the first step, this process was repeated independently for each individual parameter. This two-step process yielded values of N_P = 1000, F = 0.2, and C_R = 1. Figure 7.5 shows the behaviour of the accuracy and success rate of δDE (without discarding) for different values of the diversification parameters.

These results provide useful information about the search dynamic required by the inverse kinematics problem of a redundant robot manipulator. Further, these results also reveal the critical dependence of differential evolution on the setting of its control parameters. This being particularly noticeable for the mutation scale factor (F) and the crossover rate (C_R) when a relatively large population size is considered (see Figure 7.5). According to the observed behaviour of the memetic algorithm one might verify that a highly multimodal inverse kinematics problem demands intensive exploitation of different regions of the search space; which is effectively yielded by a small mutation scale factor, high crossover rate (mutation-only scheme), and a relatively large population size.

7.2.2 Intensification Parameters

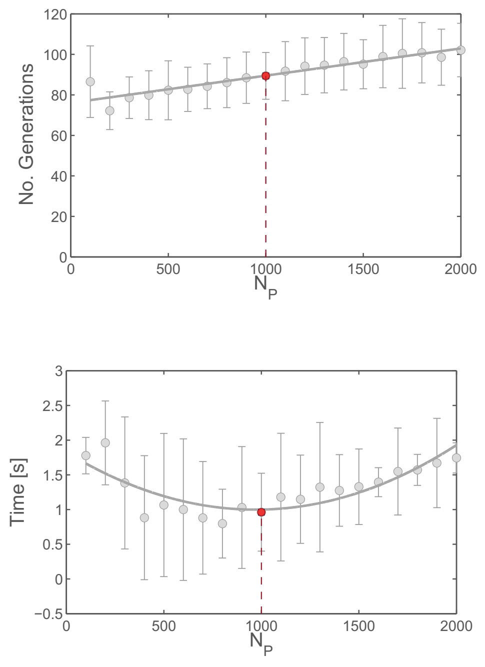
The intensification parameters of δDE are those that regulate the discarding mechanism, namely the discarding rate (δ), elite pool size (β), and local search length (σ). The first two parameters are constrained by the population size, while the third can take any value greater than or equal to zero.

Our aim was to determine how much further improvement can be obtained using discarding on the basis of optimally-tuned diversification parameters. Therefore, we sought to find a set of discarding parameters that yielded a balanced trade-off between convergence speed and accuracy, without drastically deteriorating the efficiency of δDE. Moreover, since the discarding size is directly related to the increase of the computational cost of δDE, its value shall be set such that the convergence time is considerably less than the time needed by standard DE (or δDE with the discarding size set to zero).

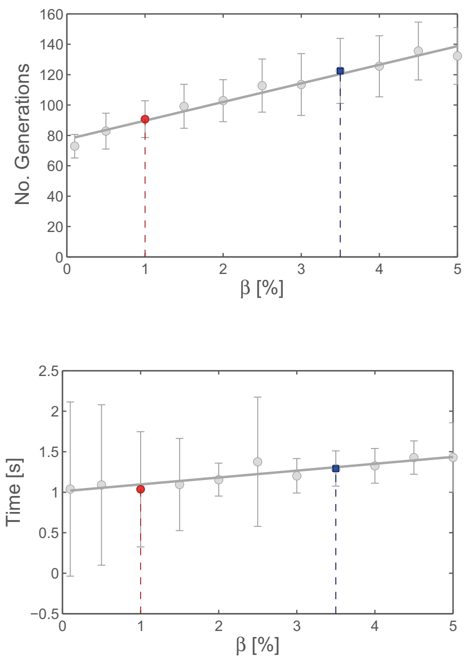
Based on this criteria, we have followed a similar simulation process to the one described for the diversification parameters. First, the parameters' range were narrowed down based on the accuracy and efficiency performance of δDE. Then, based on this reduced set of values, a second iteration of the simulation process was carried out focusing on the discarding size (δ) and the local search length (σ), given their significant impact on the convergence speed of δDE.

An initial good trade-off among accuracy, convergence speed, and computational cost was found values of $\delta = 0.5$ %, $\beta = 1$ %, and $\sigma = 2^\circ$. However, this discarding setting in combination with a small mutation factor ($F = 0.2$) would excessively promote intensification over exploration, which might eventually lead to premature convergence towards non-optimal solutions. Evidence of this behaviour is observed in the fast decline of the population diversity in early stages of the search.

To avoid the abrupt loss of population diversity during the search, the user can either increase the population (N_P) or use a larger elite pool size (β) in the local search mechanism. Nonetheless, increasing the value of any of these two parameters negatively penalizes the computational cost of the search. This is illustrated in



(a) Population size (N_P)



(b) Elite pool size (β)

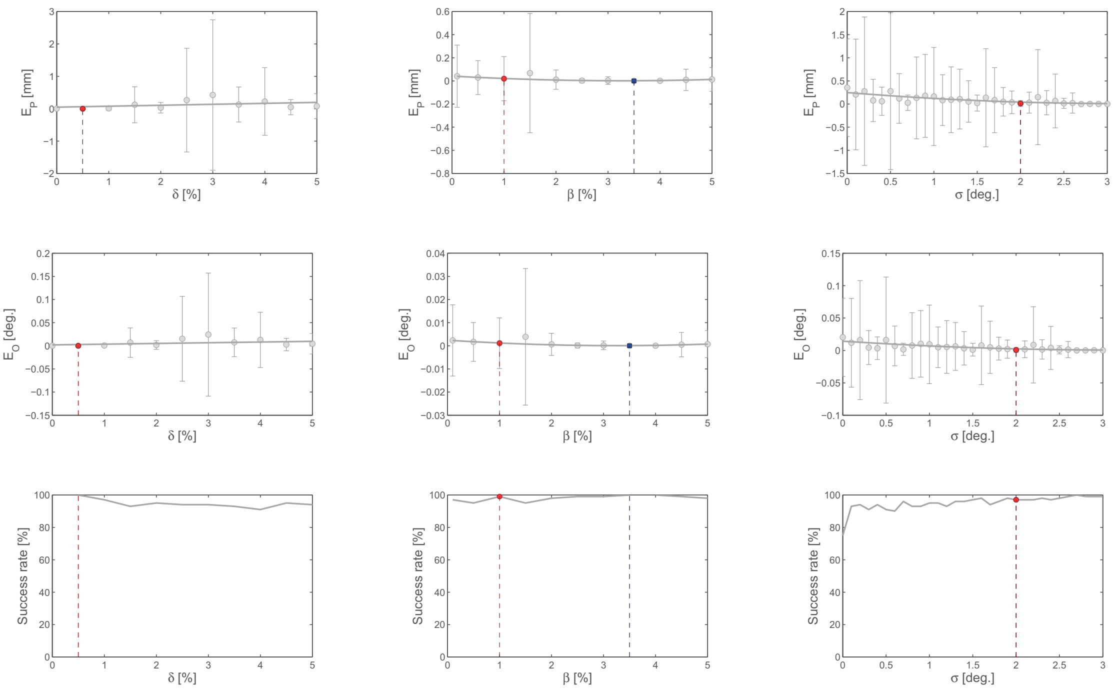
Figure 7.6: Computational cost of δDE as a function of the (a) population size (N_P) and (b) elite pool size (β).

Figure 7.6 by the average number of generations and time in seconds required for a single run of δDE to find the solution of the inverse kinematics problem.

The comparison of the two time curves shows an almost quadratic growth when the population size is increased, whereas the time curve corresponding to the elite pool size reveals a linear growth tendency. Based on these findings the elite pool size was finally set to a 3.5% of the population size ($N_P = 1000$). Figure 7.7 shows the accuracy and success rate of δDE when the intensification pressure varies.

7.2.3 Simulation Results

In this section, we evaluate the performance of δDE, with and without discarding, with the values found for the diversification and intensification parameters in sections 7.2.1 and 7.2.2. The reported results are based on one hundred independent trials on the kinematics inversion of a redundant robot manipulator (7 DOF),



(a) Discarding size (δ) (b) Elite pool size (β) (c) Local search length (σ)

Figure 7.7: Acurracy, success rate, and convergence speed of δDE as a function of the intensification parameters: discarding rate (left), elite pool size (center), and local search length (right).

Table 7.2: Comparison between standard DE and δDE as kinematics inversion methods of a redundant robot manipulator.

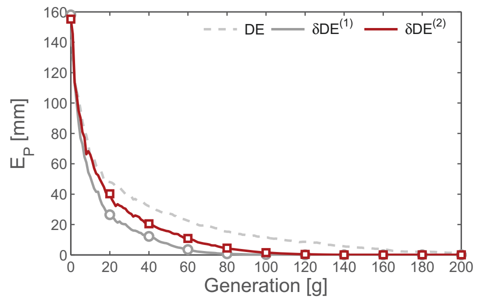
	DE	**δDE (δ,β,σ)**	
		(0.5 %, **1 %**, 2°)$^{(1)}$	(0.5 %, **3.5 %**, 2°)$^{(2)}$
$\mathbf{E}$	4.04E-08 ± 4.04E-07 (100)	3.23E-05 ± 1.92E-04 (97)	3.36E-07 ± 3.36E-06 (100)
$\mathbf{E_P}$ (mm)	3.65E-05 ± 3.65E-04(100)	2.81E-02 ± 1.73E-01(97)	3.21E-04 ± 3.21E-03(100)
$\mathbf{E_O}$ (deg.)	1.11E-06 ± 1.11E-05(100)	1.26E-03 ± 9.90E-03(100)	4.13E-06 ± 4.13E-05(100)
GEN	202.19 ± 39.87	91.80 ± 12.50	124.05 ± 18.29
AR (%)		54.60	38.65

where each trial consisted of a total five hundred generations. Table 7.2 summarizes the average accuracy, efficiency, and convergence speed of standard DE and two different parameter configurations of δDE.

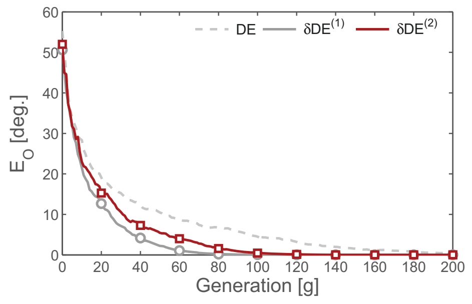
Although a small mutation scale factor (F = 0.2) produces offspring densely distributed around their parents, the additional intensification promoted by discarding during early stages of the search helps to steer the population much faster towards the optimum. Further, if the intensification pressure over the search is increased (β = 1 %) then the loss of diversity leads to premature convergence, and therefore, deteriorates the overall efficiency of the search (see Figure 7.8(c) and 7.8(d)). Figure 7.8 compares the convergence behaviour exhibited by standard DE and δDE in terms of accuracy, population diversity, and success rate.

Simulation results reveal that the discarding mechanism is able to significantly improve the convergence speed of the standard DE scheme, even when the search has been almost optimally tuned. Depending on the parameter setting of discarding, δDE reached acceleration rates of 54.60 % (β = 1 %) and 38.65 % (β = 3.5 %) in the average number of generations required by δDE to find the solution. More importantly, discarding accelerates the convergence without severely penalizing the accuracy and efficiency of the search, particularly when a relatively large elite pool size is considered (β = 3.5 %). In fact, although lower than standard DE, the position and orientation accuracies of δDE are still competitive for this particular application when contrasted to the maximum accuracies of real robot manipulators.

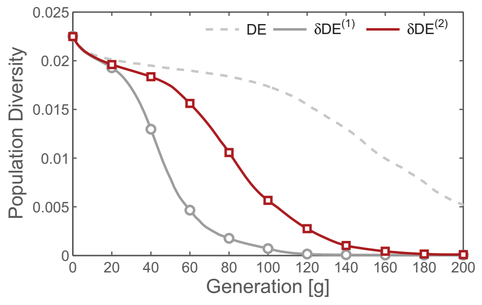
In practical terms, a single iteration of δDE is more computationally expensive than an iteration of standard DE. This is because, in each generation discarding increases the number of functions evaluations in the same proportion to the discarding rate. Thus, in addition to the number of iterations, the convergence speed



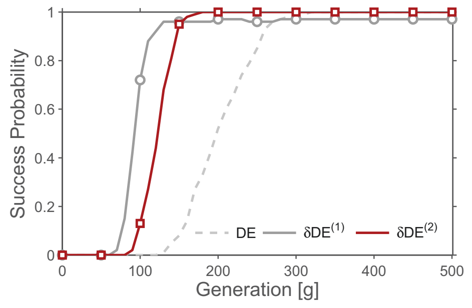
(a) Position error (E_P).



(b) Orientation error (E_O).



(c) Pupulation diversity.



(d) Success rate.

Figure 7.8: Comparison between the performance of standard DE (dashed line) and δDE (solid lines) as kinematics inversion methods. The intensification pressure of discarding is modified with two different values of the elite pool size (1 % and 3.5 %).

of each algorithm was measured as the average computation time required to find the (near-) optimal solution. Results are illustrated in Figure 7.9.

According to these results, an improvement of 54.60 % in number of generations corresponds to a 40.52 % improvement in computational time (β = 1 %); whereas, the most efficient search configuration provides a 32.54 % of improvement in time with a 38.65 % of acceleration rate in number of generations (β = 3.5 %). These values represent an important upgrade on the computational cost of standard DE; however, they are far from the approximately 88 % obtained with δDE in the previous chapter for the redundant inverse kinematics problem (see Chapter 6, Section 6.6). Obviously, by optimally adapting the parameter setting

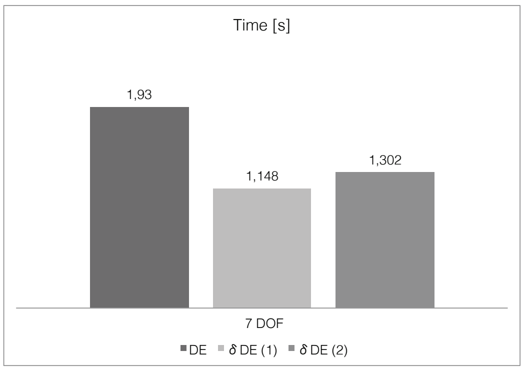


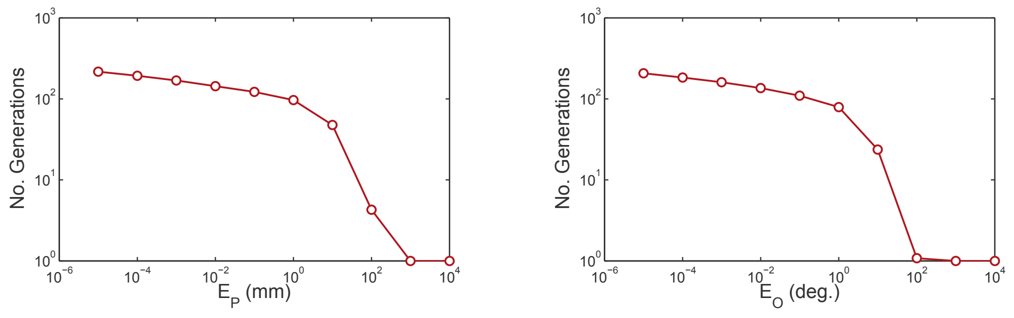
Figure 7.9: Time of convergence of standard DE and δDE.

of standard DE to the objective landscape one might expect a noticeable improvement in its performance. But it also suggests that when δDE combines into a single search strategy distinctly set diversification and intensification mechanisms, it yields a lower computational cost than optimally tuning the standard DE. Basically, since it only required a quarter of the population size needed by the standard DE optimal performance.

Throughout this study, we have demanded high-accuracy standards to evaluate the performance of δDE as kinematics inversion method. However, most benchmark robot manipulators are built to provide considerable lower positioning accuracies that are suitable for most tasks. Figure 7.10 illustrates how the convergence speed (no. of iterations) of δDE varies when considering different position and orientation thresholds as convergence conditions. These figures show that just by increasing the accuracy tolerance values to 1mm and 1°, the algorithm can obtain a further increase of 21 % and 27 % in convergence speed, respectively.

7.3 Summary

In this chapter, we have discussed the role of the control parameters on the memetic δDE algorithm. These have been classified into diversification and intensification parameters, on the one hand, to differentiate those of the `DE/rand/1/bin` scheme from those of the discarding mechanisms; and on the other hand, to characterize their main task during the search. Therefore, the diversification parameters



(a) Position error. (b) Orientation error.

Figure 7.10: Error threshold vs. convergence speed. Curves are represented in a logarithmic scale in both axes.

are those that control the mechanisms of mutation and crossover, namely the population size, mutation scale factor, and crossover rate; whereas the intensification parameters are directly related to the local search implemented by the discarding mechanism, i.e., the discarding size, elite pool size, and local search length (or standard deviation).

Based on simulation data, we have investigated how much further improvement (or not) can be obtained on the δDE performance given different diversification scenarios, in terms of success rate, accuracy, and convergence speed. The first half of the chapter was devoted to study the influence of the intensification parameters, with the premise that the diversification parameters provided a sufficiently good exploratory behaviour. The test on three different robot manipulators revealed a poor robustness of δDE to the intensification parameters, in particular to the discarding rate and the local search length. More importantly, it showed the limitations of the migration strategies as acceleration mechanisms.

In a second scenario the constraints on the mutation and crossover mechanisms as mainly diversification actors were dropped. Thus, the control parameters (diversification parameters) were allowed to optimally adapt to the landscape of the inverse kinematics problem by ignoring the discarding mechanism. That let us compared the ability of standard DE and δDE to balance the exploration and exploitation features of the search, and quantifying how much additional intensification an optimally configured standard DE accepts before deteriorating its performance.

Simulation results showed that a better trade-off between accuracy, success rate, convergence speed, and computational cost is obtained when diversification and intensification are implemented in different mechanisms (δDE).

Since the analysis on this chapter was only focused on the performance of δDE as kinematics inversion method, results should not be interpreted as a general guideline for the parameter setting of the algorithm, whatsoever; but merely as an analysis on the expected behaviour of δDE under different configurations of its parameters.

Chapter 8

Robot Hands: A higher-dimensional search space

Previous chapters have shown the performance attributes of the memetic δDE algorithm as kinematics inversion method on benchmark robot manipulators. In terms of convergence speed these findings suggest that better results are obtained on higher dimensional search spaces.

On this basis, we further investigate this feature by considering robot applications that involve a greater number of degrees of freedom. Anthropomorphic robot hands not only satisfy this requirement, but also represent a growing field of research in robotics. In the following sections we describe and test different tasks on two anthropomorphic robot hands to assess the performance of δDE on these robot applications.

8.1 Anthropomorphic Robot Hands

Human hands are extraordinary multifaceted functional mechanisms. They are endowed with *prehension* and *apprehension* abilities that allow them to grasp and dexterously manipulate different kind of objects; and also permit them to explore the environment with a fine sensory system.

Anatomically, the human hand is a multifingered extremity located at the end of the arm that includes a wrist, palm, an opposable thumb, and four fingers, called

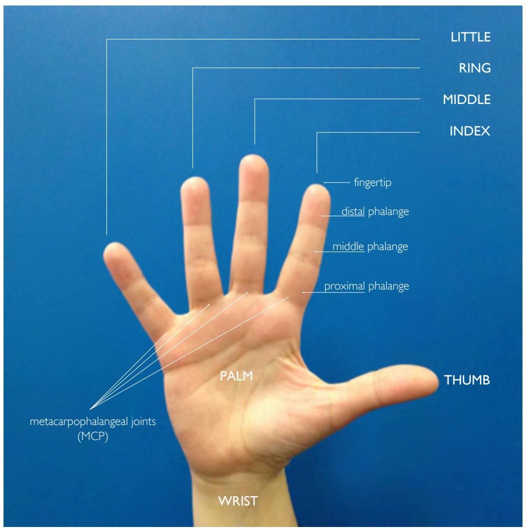


Figure 8.1: Parts of the human hand.

index, middle, ring, and little. Each finger has three phalanges known as proximal, medial, and distal. The exception is the thumb that only has a proximal and a distal phalanx. The proximal phalanx is connected to the palm by the metacarpophalangeal joint, that is approximately located at the base of the finger. The palm spans from the proximal phalanges to the wrist, which connects the hand to the forearm. Figure 8.1 identifies the anatomical parts of the human hand that will be used as reference throughout this chapter.

The human hand represents a standard paradigm for dexterous manipulation of artificial mechanisms, such as robot end-effectors and prosthetic devices. In robotics, the anthropomorphism is highly advantageous in tasks where the end-effector has to operate in human-oriented environments. The limitations of the current technology do not allow to fully mimic all the anatomical components of the human hand. Accordingly, researchers have focused on replicating the outstanding functional capabilities of the human hand in complex manipulation tasks by simplifying the kinematics structure while preserving certain degree of anthropomorphism. In the next sections, two of the most recent developments in anthropomorphic robot hand are presented.

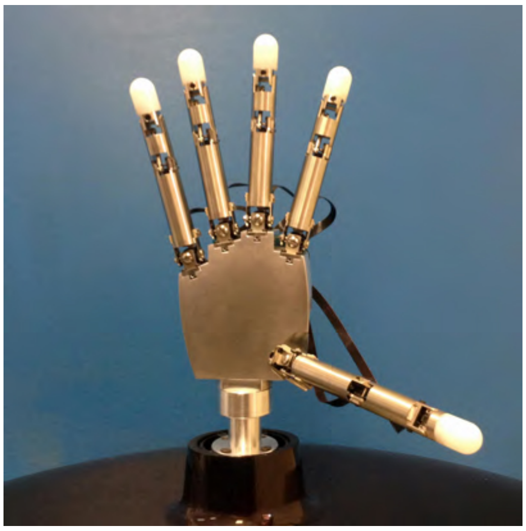


Figure 8.2: Gifu Hand III.

8.1.1 Gifu Hand III

The Gifu Hand III is a five-fingered robot hand designed to replicate the human hand in size and functionality (Mouri et al., 2002). This robot hand features 20 joints and 16 DOF distributed among four fingers and one thumb that are independently driven by built-in servomotors. The thumb has four joints with 4 DOF, and each finger has four joints with 3 DOF. The third and fourth joints of each finger are coupled as a planar four-bar linkage mechanisms, which can be approximately expressed as $q_{i_4} \approx q_{i_3}, \forall i = 2, 3, 4, 5$. The joints are arranged to allow adduction/abduction and flexion/extension motions of the thumb and each finger. Figure 8.2, shows the Gifu Hand III at the RoboticsLab of the Carlos III University (UC3M).

The kinematics of the thumb and fingers is individually defined respect to their base and transformed into the reference coordinate system of the hand located at the center of the wrist. The kinematic parameters, joint limits, and transformation matrices between the base of each finger and the wrist can be found in (Dainichi, 2004).

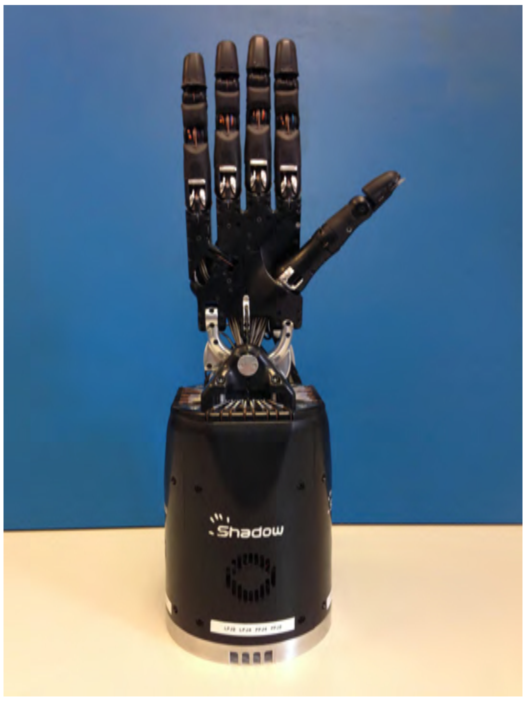


Figure 8.3: Shadow Dexterous Hand C6M. *[Courtesy of Prof. Mohamed Abderrahim]*

8.1.2 Shadow Dexterous Hand

The Shadow Dexterous Hand C6M is an anthropomorphic robot system with 20 actuated degrees of freedom. The model C6M is equipped with force and position control electronics, motor drive electronics, motor, gearbox, force sensing and communications, all integrated into a compact unit. Figure 8.3 shows the system unit of the Shadow hand at the UC3M-RoboticsLab.

The hand is provided with 24 joints distributed among an opposable thumb, four fingers of the same length, and a wrist. For convenience, the joints of the hand have been sequentially numbered starting from the wrist and ending at the distal joint of the little finger. Similarly to the Gifu Hand III, the proximal and distal interphalangeal joints of all fingers are coupled (not the thumb). This means that the distal joint is governed by the value of the proximal joint. For a detailed summary of the kinematic description of the hand and the fingers' joint limits, the reader is referred to (Shadow, 2009).

8.2 A Simple Path Generation Problem

In this section we assess the kinematics inversion performance of the δDE memetic algorithm on articulated mechanisms with a larger number of degrees of freedom. For that, we have considered a simple hand path generation problem that consisted in making the robot hand into a fist. This target path was represented as a sequence of position coordinates (nodes), defined for each fingertip respect to the wrist reference frame. The orientation of the fingers has been ignored.

Thus, each node has been mapped into a corresponding joint hand posture by minimizing the weighted sum of the position error (E_P) and the joint angular displacement (Γ), that is:

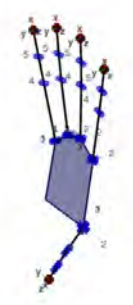
$$F_k = w \sum_{i=1}^{n_f} E_{P_{ki}} + (1-w) \sum_{i=1}^{n_f} \Gamma_{ki}, \quad k = 1, \dots, N_k \tag{8.1}$$

where k is the current node, N_k the total number of nodes, $w \in (0,1]$ is a positive weighting factor, and n_f is the number of fingers. The first term in the right hand of the equation leads the search towards the solution, whereas the second term introduces a smoothing factor of the joint configuration solution between adjoining nodes of the path.

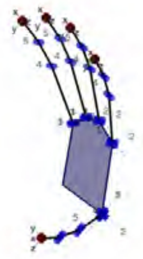
8.2.1 Simulation Setup

The same target path has been defined on the two anthropomorphic robot hands described in the previous sections. The task involves finding for each node the joint configuration of every finger of the hand, that minimizes the objective function (8.1). This poses a search problem with sixteen joint variables for the Gifu Hand III and twenty variables for the Shadow Dexterous Hand, for the latter the two wrist joints are also included. Figures 8.4 and 8.5 depict the task on the Gifu Hand III and the Shadow Dexterous Hand, respectively.

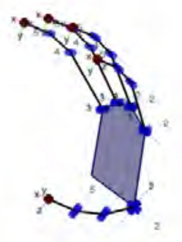
Two optimization scenarios have been examined for the path generation problem based on (8.1). First, the minimization criteria was simplified to a position error only by setting the weighting factor to one, $w = 1$. In the second scenario, the feasible solution space has been constrained to those that also provide a minimum



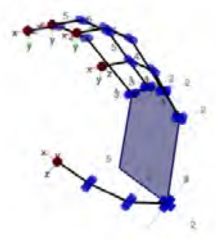
(a) Initial hand posture.



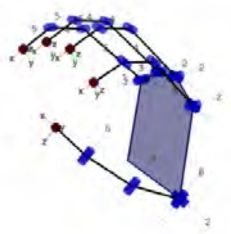
(b) k = 1.



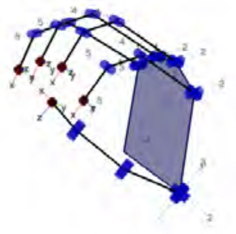
(c) k = 2.



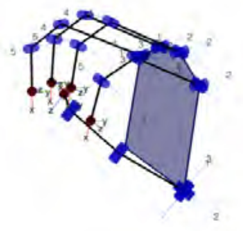
(d) k = 3.



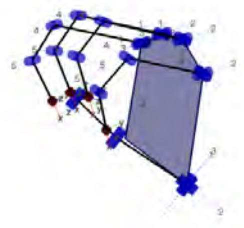
(e) k = 4.



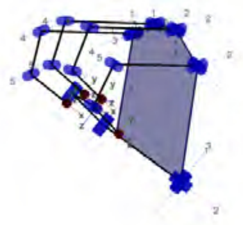
(f) k = 5.



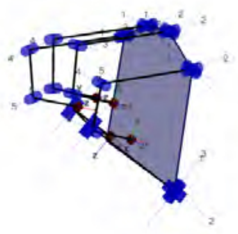
(g) k = 6.



(h) k = 7.

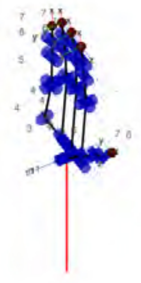


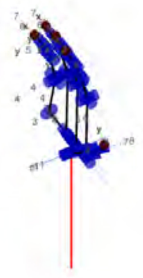
(i) k = 8.

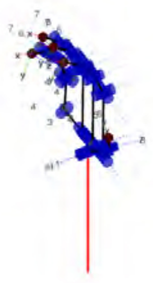


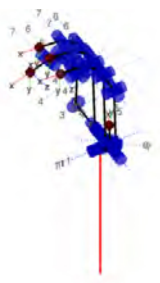
(j) k = 9.

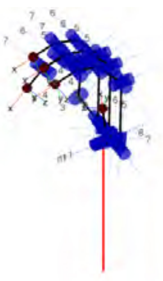
Figure 8.4: Gifu Hand III “fist” path (N_k = 9).

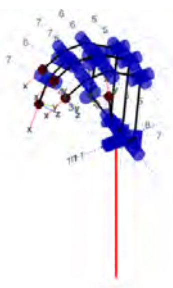

(a) Initial hand posture.


(b) k = 1.

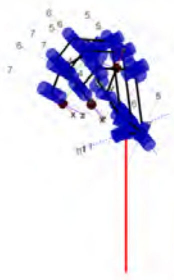

(c) k = 2.

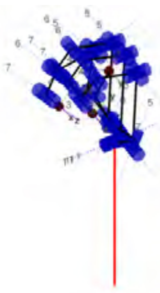

(d) k = 3.


(e) k = 4.


(f) k = 5.

(g) k = 6.


(h) k = 7.


(i) k = 8.

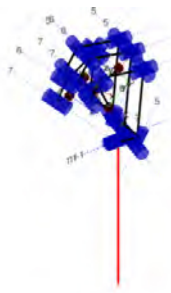

(j) k = 9.

Figure 8.5: Shadow Dexterous Hand “fist” path ($N_k = 9$).

Table 8.1: δDE control parameters for the path generation problem.

	$\mathbf{D}$	$\mathbf{N_P}$	$\mathbf{F}$	$\mathbf{C_R}$	δ	β	σ	$\mathbf{g}_{\max}$	ϵ_P	$\mathbf{N_k}$	w
Gifu Hand III	16	500	0.3	1	6 %	10 %	0.2°	100	1mm	9	0.8
Shadow Dexterous Hand	20	900	0.5	0.9	10 %	20 %	0°	200			

joint angular displacement between adjoining nodes, that is $0 < w < 1$.

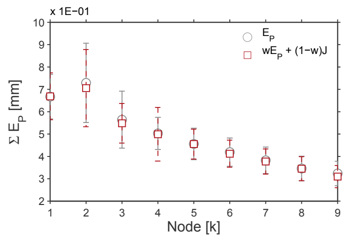
The performance of δDE has been tested on this kinematics inversion problem. Control and task parameters are listed in Table 8.1.

8.2.2 Simulation Results

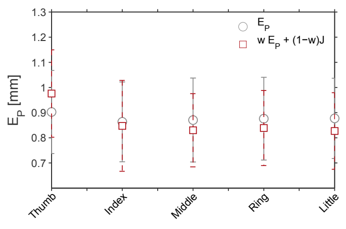
The convergence behaviour of the δDE algorithm and the accuracy of the solution have been tested on both of the proposed optimization criteria; and its analysis is based on simulation data obtained from one thousand independent trials. Figure 8.6 shows the average position error, convergence speed, and computational time obtained for the Gifu Hand III; and Figure 8.7 shows the corresponding results for the Shadow Dexterous Hand. In addition, Figure 8.8 and 8.9 compares the joint path of the thumb generated with and without joint displacement minimization.

These results reveal a good overall performance of δDE in robot applications that involve a large number of degrees of mobility. In fact, after just few generations the algorithm was able to provide high-accuracy solutions to each node of the path. For instance, it only required an average of forty five generations to find a feasible joint configuration of the Gifu Hand III that yielded less than 1mm of error in each finger. In the case of the Shadow Dexterous Hand the average increases to sixty four generations for the same error tolerance. Figure 8.6(a) and 8.7(a) illustrate the total position error obtained per node of the path. In Figure 8.6(b) and 8.7(b) the accumulated position error across the path is shown individually for each finger of the hand.

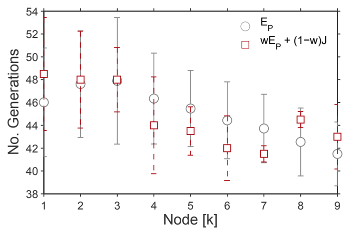
A noteworthy feature of the proposed optimization framework for kinematics inversion is its flexibility. It allows to easily define additional constraints tailored to the needs of an specific task. In this case we have considered two optimization scenarios by modifying the value of the weighting factor w in (8.1). Figures 8.8 and 8.9 compare the joint path of the thumb obtained as solution for $w = 0$, and $w = 0.8$. This figures evidence the smoothing effect introduced by the second term of (8.1).



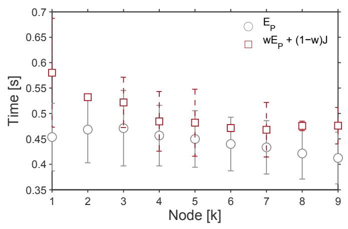
(a) Hand position error.



(b) Fingers position error.

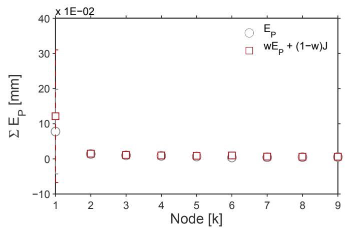


(c) Convergence speed.

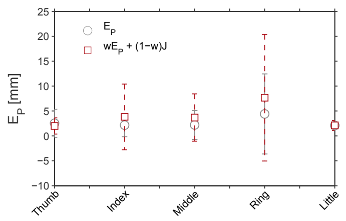


(d) Computational time.

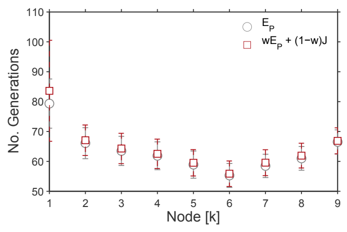
Figure 8.6: Performance of the δDE algorithm on the path generation problem for the Gifu Hand III.



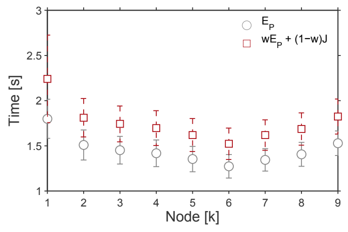
(a) Hand position error.



(b) Fingers position error.



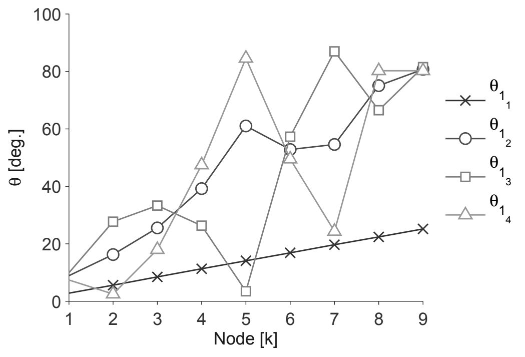
(c) Convergence speed.



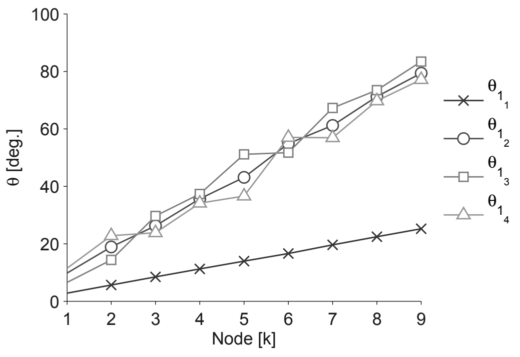
(d) Computational time.

Figure 8.7: Performance of the δDE algorithm on the path generation problem for the Shadow Dexterous Hand.

Yet, as observed in Figures 8.6 and 8.7, the convergence speed and computational time are penalized, but not the accuracy of the algorithm.

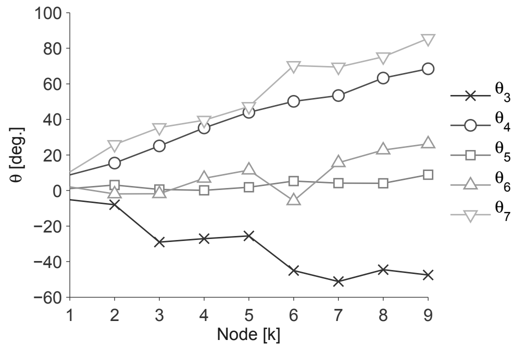


(a) $w = 1$.

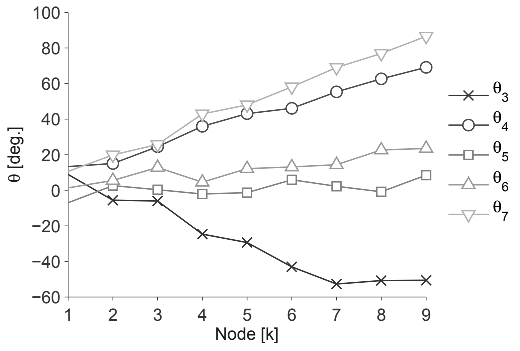


(b) $0 < w < 1$.

Figure 8.8: Comparison between the Gifu Hand III joint path solution for the thumb considering different minimization criteria. (a) Position error. (b) Position error and joint angular displacement..



(a) $w = 1$.



(b) $0 < w < 1$

Figure 8.9: Comparison between the Shadow Dexterous Hand joint path solution for the thumb considering different minimization criteria. (a) Position error. (b) Position error and joint angular dispalcement.

8.3 Hand Mobility: The Kapandji Test

The Kapandji method is a set of medical tests that evaluate the hand mobility of patients recovering from injuries or surgery. In contrast to other evaluation methods these tests do not require any measurement instrument (e.g.: goniometer) since the hand itself is taken as reference system. The test provides an overall assessment of the opposition of the thumb, flexion and extension of fingers, and grip

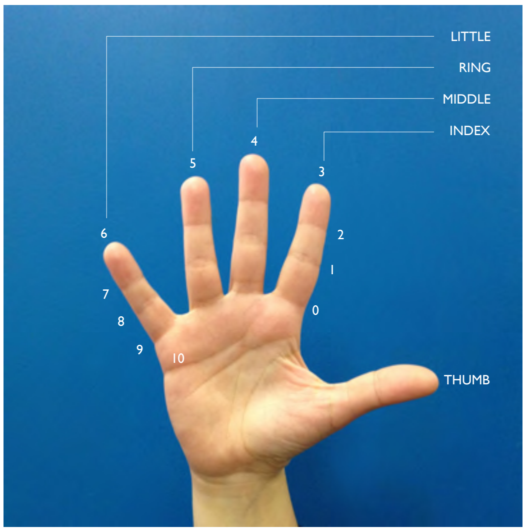


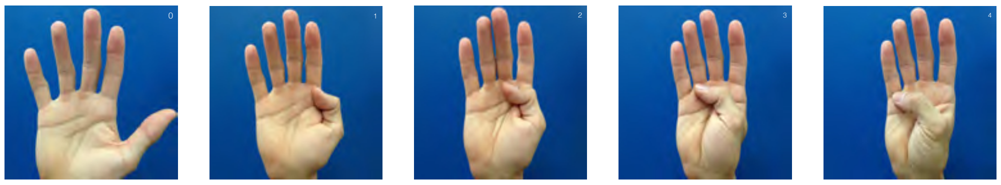
Figure 8.10: Kapandji's clinical evaluation of the thumb's opposition.

capabilities of the hand, by following testing routines that use anatomic landmarks to set combinations of fingers positions. These tests result in an index that scores the evolution of the hand as it recovers its functional mobility (Kapandji, 1987).

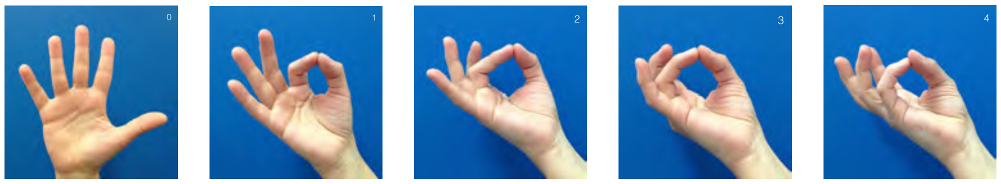
Since the opposition of the thumb is a critical component of skilled and precise manipulation, its global functional evaluation is one the most important routines of the Kapandji method (Kapandji, 1992). This exercise describes with the tip of the thumb a wide range-of-motion path that is defined in ten different stages, depicted in Figure 8.10. These stages are grouped in order to describe two grip patterns, namely terminolateral pinch or *key grip* (0-2) and tip-to-tip pinch (3-6); and lastly the extreme thumb's opposition (7-10).

As it provides a fast and reliable assessment of the functional capabilities of the hand, and particularly of the thumb, the Kapandji test has been recently used in the development process of the anthropomorphic robot hand DLR-Hand Arm System (Grebenstein et al., 2010; Chalon et al., 2010). By iteratively evaluating the prototypes with the Kapandji test and other common grasping tasks, the researchers were able to improve the kinematics of the hand until it fulfilled the aesthetics and functional requirements without any mathematical optimization of the parameters.

Motivated by the aforementioned findings, we propose an evaluation test to compare the Gifu Hand III and the Shadow Dexterous Hand, based on anthropomorphic functional features of the thumb. We have followed a similar approach as



(a) Test I: Thumb to metacarpophalangeal joint.



(b) Test II: Thumb to fingertip.

Figure 8.11: Clinical evaluation test of the thumb's mobility.

in (Grebenstein et al., 2010), by using a limited Kapandji test (Figure 8.11). The goal is to evaluate the location of the base joint of the thumb on the palm and the thumb's opposability, as important components of good grasping and manipulation skills. The evaluation is derived from the ability of the robot hand to go through target landmark locations set on the same hand. The proposed tests are the following:

Test I Contact the base of all fingers with the thumb's tip. Figure 8.11(a), stages 1 to 4.

Test II Contact the fingertip of all fingers with the tip of the thumb. Figure 8.11(b), stages 1 to 4.

These tests do not aim to be a benchmark measure of the robot hand anthropomorphism or dexterity. For this, it would be necessary a more thorough and broad set of routines that also include the most common grip patterns of the human hand. Instead, this tests merely provides insight on some of the task that can be performed with a certain robot hand.

8.3.1 Simulation Setup

Although these tests can be performed directly on the robot hand, they were implemented on a simulation environment to avoid the cumbersome task of manually controlling each joint for every target position; but more importantly, because it allows to easily adapt them to different robot hand platforms when the hardware device is not available. Furthermore, it would provide a simulation-supported tool to analyse the functionality of a robot hand during its design phase, numerically compare different designs based on the error measure obtained for each test, and finally adapt the kinematic parameters to improve its dexterity.

For the first test (I), the target positions of the thumb's tip were defined at the base of each finger approximately where the metacarpophalangeal joint is located in the human hand. The Cartesian rectangular coordinates of these landmark positions were computed through direct kinematics respect to a reference frame attached to the wrist. Accordingly, determining if any landmark in the test is within the reachable workspace of the hand inherently poses an inverse kinematics problem. To find the solution, this problem was formulated as the minimization of the position error between the known target point and the thumb's tip.

The second test (II) demands that the tip of the thumb to successfully meet the fingertip of the fingers. Contrary to the previous one, this test lacks of predefined target landmark positions for each stage of the path. Instead, it requires to find the intersection, if it exists, between the reachable workspace of the thumb and the corresponding workspace of each finger. That is, for each stage of the test, one must find one point in the task space of the hand that is reachable by both fingers. Since it basically involves solving a kinematics problem, a feasible solution would be to find (at least) one joint configuration that minimizes the distance between both fingertips at each stage. Noteworthy, the orientation and joint displacement constraints on the fingers have been ignored.

For comparison purposes, we have considered an alternative solution to the second test. In this case, the palm of the robot is used as reference to define an horizontal line equidistant from the frontal plane of the hand. The target point for each stage was thus defined at the intersection of the line and the *sagital* plane of each finger, with the exception of the fifth finger. Again, the kinematics problem was formulated as a minimization problem between the fingertips and the target

Table 8.2: δDE control parameters for the Kapandji test of hand mobility.

	$\mathbf{N_P}$	$\mathbf{F}$	$\mathbf{C_R}$	δ	β	σ	$\mathbf{g_{max}}$	ϵ_P
Gifu Hand III	100	0.5	1	6 %	10 %	0.2°	100	1mm
Shadow Dexterous Hand	900		0.9	10 %		0°		

position.

These minimization problems were solved using the δDE algorithm with the parameter setting listed in Table 8.2. The tests were run one thousand times on each robot hand, and the success threshold for all stages was set to one millimetre. Simulation results and data analysis are presented in the next section.

8.3.2 Simulation Results

The simulation results obtained for the two robot hands are reported in the Table 8.3. Results comprise the outcome of the tests and the performance of the memetic algorithm. The status refers to the state of completion of the stage according to the considered criteria (✓: success, ✗: failure). The position error (E_P) corresponds to the value obtained after a predefined maximum number of generations, whereas the convergence speed (No. Gen and Time) is related to the tolerance error (ϵ_P).

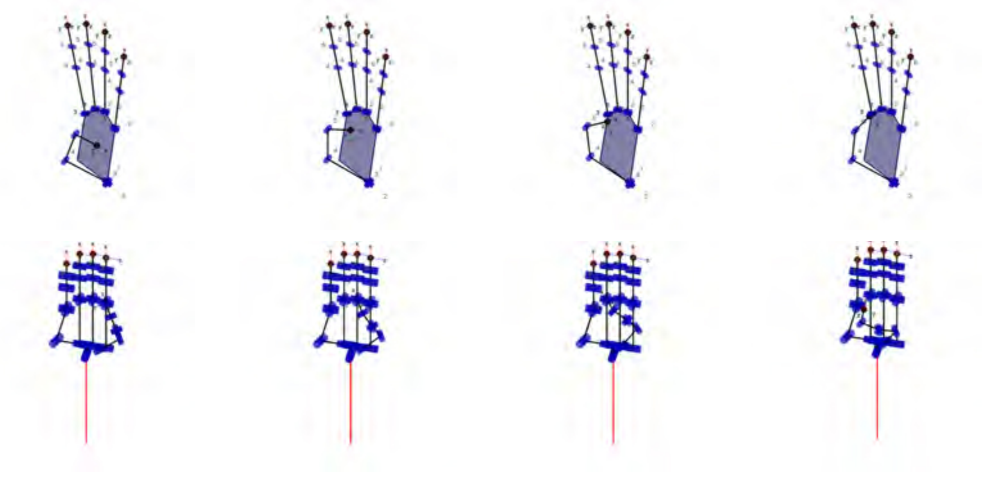
Test I evaluates the location of the base of the thumb on the palm for good grasping and manipulation performance. According to the kinematics analysis suggested in (Grebenstein et al., 2010), the thumb would fail to reach the base of all fingers, or any other position on the palm, if the intersection of the first axis of the thumb and the palm is located at the basis of one of the fingers. Although this is not the case for neither of the robot hands under study, none was able to successfully complete all the stages of the test (see Figure 8.12(a)).

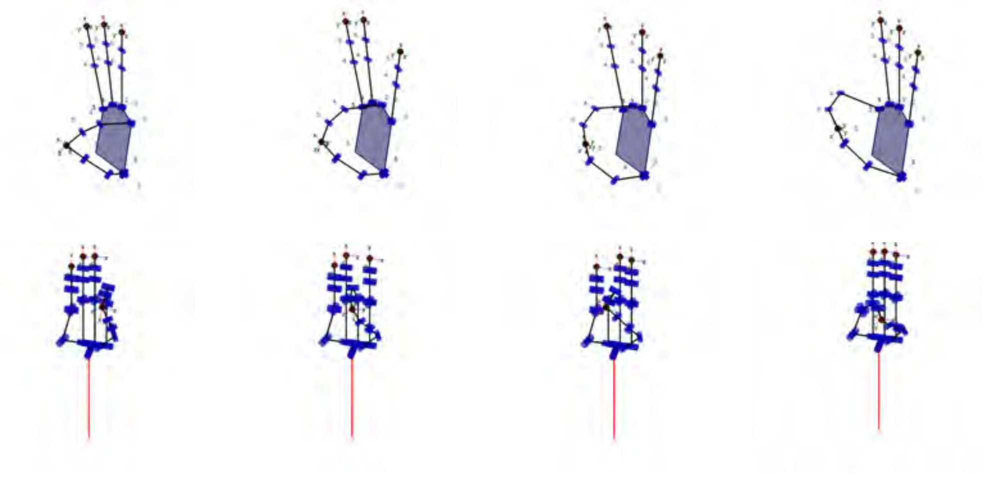
The Gifu Hand III failed to reach all the finger bases. The best results was obtained at the fourth stage of test (5th finger base) with an error of approximately 8.8mm. However, all the other stages demanded unfeasible (out-of-range) configurations of the first joint of the thumb (abduction/adduction motion). These results are partly due to the lack of longitudinal rotation of the thumb's base joint. This would allow the thumb to frontally oppose the fingers, and therefore, effectively reach their bases. Conversely, the Shadow Hand successfully reached the first three

Table 8.3: Simulation results of the Kapandji test of thumb's mobility applied to two anthropomorphic robot hands.

		Gifu Hand III				Shadow Dexterous Hand			
	Stage	Status	E_P (mm)	No. Gen.	Time (s)	Status	E_P (mm)	No. Gen.	Time (s)
Test I	1	✗	6.74E01±2.97E-01(0)	–	–	✓	5.95E-04±3.79E-03(100)	10.70±1.72	0.103±0.018
	2	✗	6.33E01±2.97E-07(0)	–	–	✓	8.20E-04±6.78E-03(100)	10.48±1.73	0.099±0.017
	3	✗	4.07E01±1.10E-06(0)	–	–	✓	4.10E-04±2.49E-03(100)	10.98±1.80	0.097±0.013
	4	✗	8.82E00±2.54E-06(0)	–	–	✗	8.64E00±1.55E-05(100)	–	–
Test II (1)	1	✓	6.64E-03±5.35E-03(100)	14.07±2.35	0.056±0.013	✓	2.37E-03±1.43E-02(100)	8.85±1.88	0.181±0.038
	2	✓	6.22E-03±4.80E-03(100)	14.18±2.30	0.055±0.013	✓	1.30E-03±8.22E-03(100)	10.22±1.82	0.206±0.039
	3	✓	5.55E-03±4.58E-03(100)	13.72±2.37	0.055±0.015	✓	8.81E-01±1.01E-04(100)	47.58±3.20	0.856±0.085
	4	✓	4.27E-03±3.62E-03(100)	13.18±2.45	0.050±0.012	✓	1.51E-03±8.57E-03(100)	13.82±1.95	0.259±0.034
Test II (2)	1	✓	1.18E-02±5.92E-03(100)	23.83±3.09	0.137±0.016	✓	1.26E-05±1.15E-05(100)	17.89±1.62	0.385±0.051
	2	✓	1.17E-02±6.01E-03(100)	23.38±2.51	0.134±0.014	✓	1.01E-05±9.19E-06(100)	17.07±1.62	0.371±0.052
	3	✓	1.12E-02±5.47E-03(100)	22.59±2.77	0.144±0.023	✗	9.94E00±2.37E-06(0)	–	–
	4	✓	1.26E-02±6.53E-03(100)	18.58±2.31	0.115±0.020	✓	6.60E-05±6.56E-05(100)	23.45±2.08	0.496±0.062

(1) Minimization of the distance between the thumb's tip and corresponding fingertip.
(2) Minimization of the position error of the thumb's tip and corresponding fingertip respect to a predefined target position.


(a) Test I. Thumb's tip to the base joint of fingers.


(b) Test II. Thumb's tip meets the fingertip of fingers.

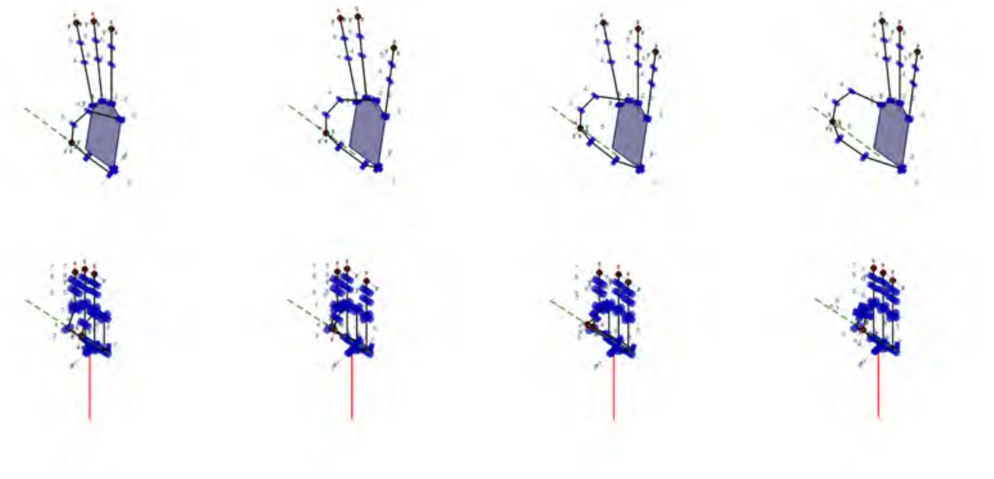
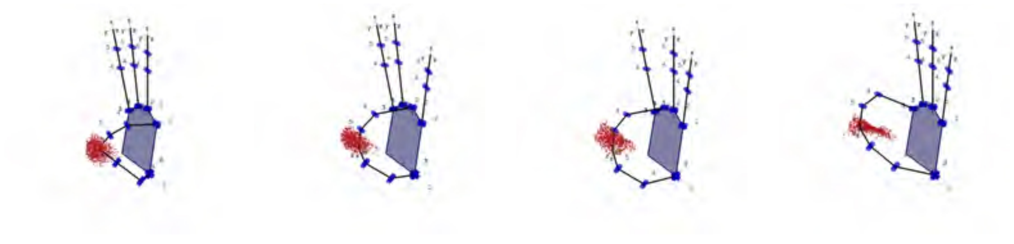

(c) Test II. Thumb tip and fingertip of fingers meet at a predefined target position.

Figure 8.12: Kapandji test of thumb's opposability anthropomorphic robot hands. The stages of each test are shown from left to right in (a),(b), and (c). Two anthropomorphic robot hands have been tested: Gifu Hand III (first, third, and fifth rows) and Shadow Hand (second, fourth, and sixth rows).


(a) Gifu Hand III.

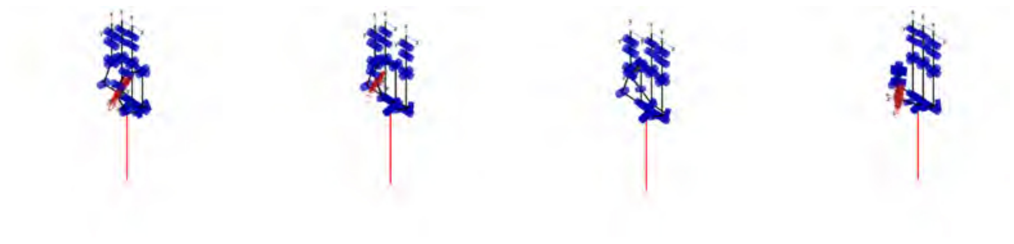

(b) Shadow Dexterous Hand.

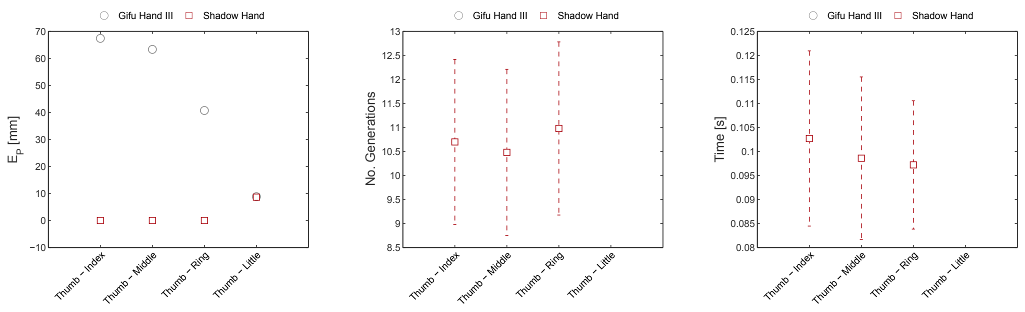
Figure 8.13: Multiple feasible solutions of the second test of the Kapandji method.

stages of the test while it failed the last one with an error of 8.6mm.

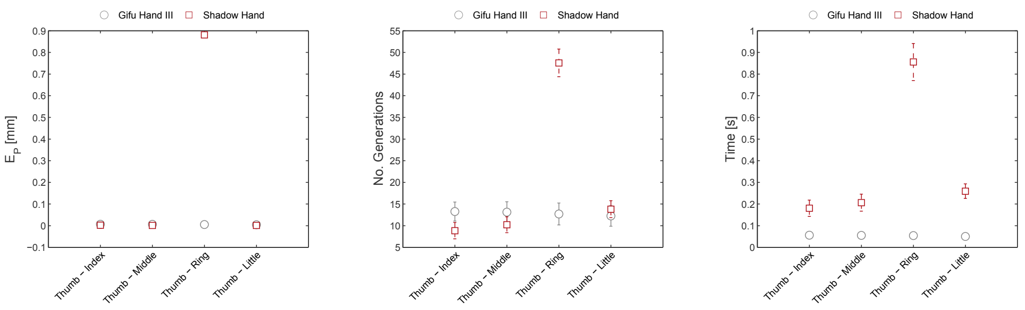
Test II is oriented to evaluate the opposability and fine manipulation capabilities of the robot hand (see Figure 8.12(b) and 8.12(c)). Every stage of the test was reproduced based on two different optimization criteria: (i) minimal distance between the tip of the thumb and the corresponding fingertip (Figure 8.12(b)), and (ii) minimal position error to a predefined target position within the workspace of both fingers (Figure 8.12(c)) . The latter aims at evaluating the positioning for fine manipulation, whereas the former takes advantage of the redundancy of the solution (see Figure 8.13).

Although the Gifu Hand III lacks of longitudinal rotation of the thumb for proper thumb's opposition, the long links of the thumb and fingers (phalanges) enlarge the reachable workspace of the hand. The resulting workspace is large enough to ensure that the Gifu Hand III is able to efficiently complete all the stages of the second test of the Kapandji method under the two considered criteria.

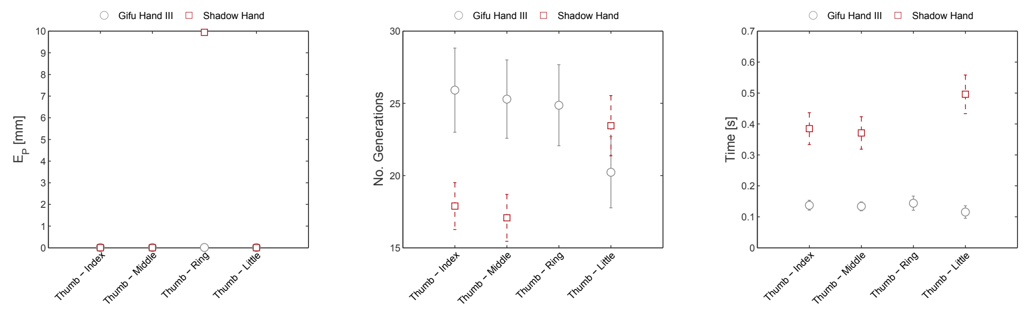
Similar good results were obtained for the Shadow Dexterous Hand. In contrast to the Gifu Hand III, the fingers of the Shadow hand preserve the dimension size ratios of an average adult human male hand. The shorter thumb provides grasping stability, but also a smaller reachable workspace. This is largely circumvented by



(a) Test I of the Kapandji method.



(b) Test II of the Kapandji method. Criteria: minimal distance between thumb's and fingers' tip.



(c) Test II of the Kapandji method. Criteria: minimal distace to predefined target position.

Figure 8.14: Simulation results of the Kapandji method on two anthropomorphic robot hands.

Table 8.4: Mobility index of the Kapandji method.

	Gifu Hand III					Shadow Dexterous Hand				
Test	Stage/Score				Subtotal	Stage/Score				Subtotal
	1	2	3	4		1	2	3	4	
Test I	0	0	0	0	0	1	2	3	0	6
Test II(1)	1	2	3	4	10	1	2	3	4	10
Test II(2)	1	2	3	4	10	1	2	0	4	7
Total					**20**/30					**23**/30

(1) Minimization of the distance between the thumb's tip and corresponding fingertip.
(2) Minimization of the position error of the thumb's tip and corresponding fingertip respect to a predefined target position.

the additional degree of mobility at the base of the thumb and little finger, that allows the thumb to position its tip closer to the palm and the fingertips of the other fingers. However, the simulation data show some disparities in the results obtained for the third stage of the test. A further examination of the results reveals that the minimal feasible distance between the thumb's tip and the ring's fingertip is of 0.8mm, which occurs when the first and second joints of the thumb reach their upper limit. Moreover, since this distance is lower than the maximum tolerance error the hand passes the test, but fails when a predefined position is set as target of both fingers. Figure 8.14 graphically compares the results of the Kapandji tests and the performance of the δDE algorithm on the two robot hands.

8.3.3 Evaluation

In order to monitor the patient's progress during the recovery process, each stage of the Kapandji tests is attributed with a score and evaluated accordingly. An index of hand mobility is thus derived by summing the scores of all the stages that are correctly performed by the patient (Kapandji, 1992; De Soras et al., 1994). Similarly, we have assessed the robot performance with a mobility index. To keep it simple, each stage has been scored with its own number in the routine, i.e., stage one scores one point. This is consistent with the fact that stages are presented in ascending order of difficulty. Table 8.4 summarizes the evaluation results obtained for the Gifu Hand III and Shadow Dexterous Hand.

8.3 Summary

The aim of this chapter was to provide further evidence of the performance of δDE algorithm as kinematics inversion method, this time considering a much larger feasible search space.

For this we have considered two robot hand applications. First, a simple path generation was formulated to examine the accuracy and convergence speed of δDE over a search space with sixteen and twenty independent joint variables, respectively. The second was focused on asserting the versatility of δDE on the evaluation of anthropomorphic robot hands mobility, based on standard clinical tests used on human hands. In both applications, δDE demonstrated its high flexibility in solving kinematics problems.

More importantly, the results show that the algorithm is a simple and fast method that allows to perform different manipulability tests during the design phase of anthropomorphic robot hand systems, as well as to evaluate and compare existing ones. The method provides a tool to improve or adapt the design to the functional specifications of the robot hand based on quantitative measures. This allows to compare the functionality of different designs and choose the one that meets the desired requirements.

Chapter 9

Conclusions

The inverse kinematics problem of robot manipulators states the functional mapping of a known end-effector pose into the joint configuration coordinates of the robot. This represents one of the fundamental problems of the kinematics analysis of robot manipulators, and except for a subset of kinematics geometries it cannot be analytically solved in closed-form. Therefore, a generic solution necessarily demands a numerical formulation of this problem.

In this thesis we have proposed a numerical solution for the kinematics inversion of generic robot manipulators, based on the memetic differential evolution scheme δDE. This approach transforms the functional mapping problem into a constrained non-linear optimization model in the configuration space of the robot. That is, it defines a direct search over the feasible configuration space of the robot to find the joint configuration vector that minimizes the position and orientation error of the end-effector (Chapter 5). The optimization approach for kinematics inversion has the following advantages: (i) generic, it admits robot manipulators with any kinematics geometry; (ii) simple, it only requires to define the forward kinematics equation of the robot to compute the end-effector pose error; (iii) flexible, it allows to combine different optimization criteria into a single objective function; (iv) direct, the solution is obtained in joint coordinates; (v) numerically stable, since it does not require the inverse Jacobian matrix for mapping between the Cartesian and the configuration spaces; (vi) and it directly handles the constraints on the search space.

Despite the many advantages of the optimization approach to kinematics inversion, a mayor concern that usually rises about numerical solutions, and in particular about population-based direct search methods, is the high computational cost associated to their low convergence speed. The proposed approach addresses this issue by combining a global search mechanism (`DE/rand/1/bin`) that efficiently explores the configuration space of the robot to find the solution, and an independent local improvement method (discarding) that exploits the most promising regions of the search space to increase the convergence speed. The balanced synthesis of both mechanisms within an evolutionary search scheme yields the memetic algorithm δDE.

Although previous works had already suggested a discarding mechanism to accelerate the convergence speed of the standard differential evolution, thus far its operating principle had been vaguely described. The main theoretical contributions of this thesis are related to the identification, formulation, and analysis of the components and control parameters of the memetic differential evolution algorithm δDE (Chapter 4).

On a further analysis about the operating principles of *discarding* we have identified two mechanisms: *migration* and *local search*. Migration provides a replacement policy to refresh the population with new information, whereas the local search implements the replacement strategy. The policy and the strategy of the discarding mechanism are controlled by three parameters: the discarding rate (δ), the elite pool size (β), and the local search length (σ). Their roles within the memetic search scheme framework have been identified, and their influence on the convergence behaviour of the search have been further investigated based on the empirical analysis of simulation experiments (Chapter 7). Although the analysis is only limited to the inverse kinematics problem, this is the first study that have been devoted to examine the parameter setting of the discarding mechanism.

Moreover, considerable testing have been presented to assess the performance of δDE as kinematics inversion method for redundant and non-redundant benchmark robot manipulators, as well as for anthropomorphic robot hands (Chapters 6, 7 and 8). The most important findings reported in this thesis can be summarized as:

- The discarding mechanism significantly improves the convergence speed of the standard DE direct search scheme. Its simple local improvement strategy effectively compensates the exploitation deficiencies of the standard DE search scheme, leading the population towards global minimizers in fewer generations. Nonetheless, as argued in Section 6.6 (Chapter 6), a remarkable reduction in the number of generations does not necessarily entail a lower computational cost, since a single iteration of δDE is actually more computationally expensive than one of the standard DE. An estimation of the average computational time of δDE showed that, although the margin of improvement is notably lower than the obtained in terms of number of generations, δDE still outperforms the standard DE, particularly on higher-dimensional configuration spaces.

- δDE provides high-accuracy solutions to the inverse kinematics problem of articulated robot systems. The local refinement strategy of discarding yield lower average error values in position and orientation without disrupting the overall global search of the `DE/rand/1/bin` scheme. This is highly advantageous for tasks that demands precise location of the end-effector with a competitive computational time.

- The proposed framework easily accommodates different requirements for the inverse kinematics problem with minor modifications of the objective function. Moreover, the design vector or candidate solutions can be defined to represent the joint configuration vector of a single robot manipulator, the joint configuration path for a finite set of nodes, or the joint configuration of all the fingers of a robot hand. It only requires to define, analytically or numerically, the forward kinematics model of the robot and the values of the control parameters. Such flexibility allows to perform kinematics analysis, design, calibration, and synthesis on a wide range of kinematics structures and tasks.

- The δDE algorithm increases to six the number of control parameters, classified according to their role in the search as diversification (N_P, F and C_R) and intensification (δ, β, and σ) parameters. Although this increases the complexity of the algorithm and its parameter setting, simulation results showed that a good trade-off between accuracy, efficiency, convergence speed, and computational cost is obtained by independently promoting exploration and exploitation through their corresponding parameters.

- The performance of δDE is particularly sensitive to the value of the discarding rate (δ). That is, a large migration rate increases the number of similar solutions in each generation, which leads to premature convergence. This has been observed as a decline in the success rate of the the algorithm as the value of the discarding rate (δ) increases.

In general, the proposed memetic approach provides a generic solution to the inverse kinematics problem of robot manipulators. By enhancing the exploitation features of the search with the discarding mechanism, we have been able to significantly improve the convergence speed while obtaining high-accuracy solutions. However, this approach also has some important limitations:

- Although δDE has been able to find the solution to the inverse kinematics of a 16 DOF anthropomorphic robot hand in less than a second, still it is not suitable to control real-time applications.

- In contrast to other numerical kinematics inversion methods, including direct search solutions (Tabandeh et al., 2006, 2010), this approach only converge to a single solution from the multiple possible solutions of the inverse kinematics problem.

- Moreover, the algorithm does not provide any information about the number of solutions of the inverse kinematics problem for any kinematic structure.

- According to the "No free lunch theorem" (Wolpert & Macready, 1997) the obtained simulation results supporting the superior performance of δDE over the standard DE scheme cannot be generalized to other optimization problems, and without further proof these are only constrained to this particular robot application.

9.1 Future Research

Being one of the fundamental problems in the field of robotics, the inverse kinematics problem of robot manipulators has drawn much attention from researcher over the decades. The result has been a sizeable group of numerical tools that aim to find a generic and efficient solution to this problem. We believe that by revisiting the inverse kinematics problem not only we have been able to explore a new approach to kinematics inversion, but also have laid the bases that hopefully might inspire others to find new applications of δDE in robotics or in other fields.

A promising niche of application is in the kinematics evaluation of complex robot systems with many degrees of freedom during their design phase. Preliminary results on the evaluation of anthropomorphic robot hands mobility showed that the method provides a useful tool to improve or adapt the design to the functional specifications of the robot hand based on quantitative measures. This is also useful in the kinematics calibration of robot manipulators.

A related but more challenging application is the kinematics synthesis of articulated robot systems (Tabandeh, 2009). Whether to determine the number and type of joints required to perform a specific task (type synthesis), or to compute the size of the articulated bodies of a predefined kinematics layout (dimensional synthesis), for instance, from biometric measures of the human hand.

Further research can also be focused on improving the understanding of the dynamic of the mechanisms that interplay in the direct search scheme of δDE. Particularly on the parameter setting that provides a suitable balance between global convergence, accuracy, and acceleration rates. In this regard, it would be interesting to investigate adaptation mechanisms of their values according to the needs of the search during different stages of the evolutionary process. For instance, reducing the discarding rate as the population gets closer to convergence would decrease the overall computational cost of the algorithm, and it would prevent premature convergence due to excessive exploitation. Alternatively, the local search length can be adaptively modified during the evolutionary process according to the distribution of the population over the search space.

Finally, alternative representations of the position and orientation error, as well as other optimization criteria, could be tested to examine their influence on the performance of the kinematics inversion method.

Bibliography

Aydin, Y., & Kucuk, S. (2006). Quaternion based inverse kinematics for industrial robot manipulators with Euler wrist. In *IEEE International Conference on Mechatronics, ICM'06*, (pp. 581–586).

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press.

Bäck, T., & Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, *1*(1), 1–23.

Baillieul, J. (1985). Kinematic programming alternatives for redundant manipulators. In *Proc. IEEE International Conference on Robotics and Automation, ICRA'85*, (pp. 722–728).

Bensalah, C., González-Quijano, J., Zhang, J., & Abderrahim, M. (2014). A new extended SDLS to deal with the JLA in the inverse kinematics of an anthropomorphic robotic hand. In *Advances in Intelligent Systems and Computing. ROBOT2013: First Iberian Robotics Conference*, vol. 253, (pp. 661–673).

Bingul, Z., Ertunc, H., & Oysu, C. (2005). Comparison of inverse kinematics solutions using neural network for 6R robot manipulator with offset. In *Congress on Computational Intelligence Methods and Applications*, (p. 5 pp.).

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, *10*(6), 646–657.

Brest, J., & Sepesy Maučec, M. (2008). Population size reduction for the differential evolution algorithm. *Applied Intelligence*, *29*, 228–247.

Buckley, K., Hopkins, S., & Turton, B. (1997). Solution of inverse kinematics problems of a highly kinematically redundant manipulator using genetic algorithms.

In *2nd Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA 97.*, (pp. 264–269).

Chalon, M., Grebenstein, M., Wimboeck, T., & Hirzinger, G. (2010). The thumb: Guidelines for a robotic design. In *International Conference on Intelligent Robots and Systems, IROS'10*, (pp. 5886–5893).

Chiaverini, S., Siciliano, B., & Egeland, O. (1994). Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. *IEEE Transactions on Control Systems Technology*, *2*(2), 123–134.

Chiddarwar, S. S., & Babu, N. R. (2010). Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach. *Engineering Applications of Artificial Intelligence*, *23*(7), 1083–1092.

Dainichi (2004). *Operation manual for Gifu Hand III – Hand mechanism*. Dainichi Co. Ltd., 1-33, Himegaoka, Kani, Gifu Prefecture 509-0249 Japan.

Das, S., Konar, A., & Chakraborty, U. K. (2005). Two improved differential evolution schemes for faster global search. In *Proc. ACM-SIGEVO GECCO*, (pp. 991–998).

Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press, Oxford, UK.

De Soras, X., Guinard, D., Moutet, F., & Gerard, P. (1994). A proposal for a functional evaluation record for the hand. *Annales de Chirurgie de la Main et Du Membre Superieur*, *13*(5), 297–307.

Denavit, J., & Hartenberg, R. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Applied Mechanics ASME*, (pp. 215–221).

Deo, A., & Walker, I. (1995). Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems*, *14*, 43–68.

Driscoll, J. (2000). Comparison of neural network architectures for the modeling of robot inverse kinematics. In *Proceedings of the IEEE Southeastcon*, (pp. 44–51).

Elgazzar, S. (1985). Efficient kinematic transformations for the PUMA560 robot. *IEEE Journal of Robotics and Automation*, *1*(3), 142–151.

Epitropakis, M., Plagianakos, V., & Vrahatis, M. (2009). Evolutionary adaptation of the differential evolution control parameters. In *IEEE Congress on Evolutionary Computation, CEC '09*, (pp. 1359 –1366).

Epitropakis, M., Tasoulis, D., Pavlidis, N., Plagianakos, V., & Vrahatis, M. (2011). Enhancing differential evolution utilizing proximity-based mutation operators. *IEEE Transactions on Evolutionary Computation*, *15*(1), 99 –119.

Fan, H.-Y., & Lampinen, J. (2003). A trigonometric mutation operation to differential evolution. *Journal of Global Optimization*, *27*, 105–129.

Fogel, D. (1992). *Evolving artificial intelligence*. Ph.D. thesis, University of California.

Fogel, D. (1994). An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, *5*(1), 3–14.

Fogel, L. (1962). Autonomous automata. *Industrial Research*, *4*, 14–19.

Fogel, L. (1964). *On the organization of intellect*. Ph.D. thesis, University of California.

Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study for differential evolution. In *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, (pp. 293–298).

Goldberg, D. E. (1989). Sizing populations for serial and parallel genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, (pp. 70–79). San Francisco, CA, USA.

Goldenberg, A., Benhabib, B., & Fenton, R. (1985). A complete generalized solution to the inverse kinematics of robot. *IEEE J. of Robotics and Automation*, *RA-1*(1), 14–20.

Gong, W., Cai, Z., & Ling, C. (2010). DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, *15*, 645–665.

González, C., & Blanco, D. (2013). A memetic differential evolution algorithm for the inverse kinematics problem of robot manipulators. *International Journal of Mechatronics and Automation*, *3*(2).

González, C., Blanco, D., & Moreno, L. (2009). Optimum robot manipulator path generation using differential evolution. In *IEEE Congress on Evolutionary Computation, CEC'09*, (pp. 3322–3329).

Grebenstein, M., Chalon, M., Hirzinger, G., & Siegwart, R. (2010). A method for hand kinematics designers: 7 billion perfect hands. In *1st International Conference on Applied Bionics and Biomechanics, ICABB'10*.

Guez, A., & Ahmad, Z. (1988). Solution to the inverse kinematics problem in robotics by neural networks. In *Proc. IEEE International Conference on Neural Networks*, vol. 2, (pp. 617–624).

Hasan, A., et al. (2006). An adaptive-learning algorithm to solve the inverse kinematics problem of a 6DOF serial robot manipulator. *Advances in Engineering Software*, *37*(7), 432–438.

Holland, J. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM*, *9*(3), 297–314.

Holland, J. (1975). *Adaptation in natural and artificial systems*. The University of Michigan Press.

Iacca, G., Neri, F., & Mininno, E. (2011). Opposition-based learning in compact differential evolution. In *Applications of Evolutionary Computation*, vol. 6624 of *Lecture Notes in Computer Science*, (pp. 264–273). Springer Berlin / Heidelberg.

Jia, D., Zheng, G., & Khan, M. K. (2011). An effective memetic differential evolution algorithm based on chaotic local search. *Information Sciences*, *181*(15), 3175 – 3187.

Kalra, P., Mahapatra, P., & Aggarwal, D. (2004). On the comparison of niching strategies for finding solution of multimodal robot inverse kinematics functions. In *IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, (pp. 5356–5361).

Kapandji, A. (1987). Proposal for a clinical score for flexion-extension of the long fingers. *Annales de Chirurgie de la Main et Du Membre Superieur*, *6*(4), 288–94.

Kapandji, A. I. (1992). Clinical evaluation of the thumb's opposition. *Journal of Hand Therapy*, *5*(2), 102 – 106.

Karlik, B., & Aydin, S. (2000). An improved approach to the solution of inverse kinematics problems for robot manipulators. *Engineering Applications of Artificial Intelligence*, *13*(2), 159–164.

Kim, S., & Kim, J.-H. (1996). Optimal path generation of a redundant manipulator with evolutionary programming. In *Proc. IEEE Int. Conf. on Industrial Electronics, Control, and Instrumentation, IECON'96*, vol. 3, (pp. 1909–1914).

Kucuk, S., & Bingul, Z. (2004). The inverse kinematics solutions of industrial robot manipulators. In *IEEE International Conference on Mechatronics, ICM'04.*, (pp. 274–279).

Kuroe, Y., Nakai, Y., & Mori, T. (1993). A new neural network approach to the inverse kinematics problem in robotics. In *Proc. Asia-Pacific Workshop on Advances in Motion Control*, (pp. 112–117).

Lampinen, J., & Zelinka, I. (2000). On stagnation of the differential evolution algorithm. In *Proceedings of 6th International Mendel Conference on Soft Computing*, (pp. 76–83).

Lenarcic, J. (1985). An efficient numerical approach for calculating the inverse kinematics for robot manipulators. *Robotica*, *3*(1), 21–26.

Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, *9*, 448–462.

Maciejewski, A. A., & Klein, C. A. (1988). Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *Journal of Robotic Systems*, *5*(6), 527–552.

Mallipeddi, R., & Suganthan, P. (2008). Empirical study on the effect of population size on differential evolution algorithm. In *IEEE Congress on Evolutionary Computation CEC'08*, (pp. 3663–3670).

Manocha, D., & Canny, J. F. (1994). Efficient inverse kinematics for general 6R manipulators. *IEEE Transactions on Robotics and Automation*, *10*, 648–657.

Martín, F. (2012). *Evolutionary-based Global Localization and Mapping of three-dimensional environments*. Ph.D. thesis, Universidad Carlos III de Madrid, Leganés, Spain.

Martín, F., Moreno, L., Garrido, S., & Blanco, D. (2011). High-accuracy global localization filter for three-dimensional environments. *Robotica*, (pp. 1–16).

Moed, M., Stewart, C., & Kelley, R. (1990). *Reducing the search time of a genetic algorithm using the immigration operator*. CIRSSE report. Center for Intelligent Robotic Systems for Space Exploration, Rensselaer Polytechnic Institute.

Monar, F. M., González, C., Moreno, L., & Blanco, D. (2010). Accelerated localization in noisy 3d environments using differential evolution. In *GEM*, (pp. 166–172). CSREA Press.

Moreno, L., Blanco, D., Muñoz, M. L., & Garrido, S. (2011). L1-L2-norm comparison in global localization of mobile robots. *Robotics and Autonomous Systems*, *59*(9), 597–610.

Morris, A., & Mansor, A. (1997). Finding the inverse kinematics of manipulator arm using artificial neural network with lookup table. *Robotica*, *15*(06), 617–625.

Morris, A. S., & Mansor, M. A. (1998). Manipulator inverse kinematics using an adaptive back-propagation algorithm and radial basis function with a lookup table. *Robotica*, *16*(04), 433–444.

Mouri, T., Kawasaki, H., Yoshikawa, K., Takai, J., & Ito, S. (2002). Anthropomorphic robot hand: Gifu Hand III. In *ICCAS'02*, (pp. 1288–1293).

Nakamura, Y., & Hanafusa, H. (1986). Inverse kinematic solution with singularity robustness for robot manipulator control. *ASME Journal of Dynamic Systems, Measurement, and Control*, *108*, 163–171.

Nearchou, A. (1998). Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. *Mechanism and machine*, *33*(3), 273–292.

Neri, F., Iacca, G., & Mininno, E. (2011). Disturbed exploitation compact differential evolution for limited memory optimization problems. *Information Sciences*, *181*(12), 2469 – 2487.

Neri, F., & Mininno, E. (2010). Memetic compact differential evolution for cartesian robot control. *IEEE Computational Intelligence Magazine*, *5*(2), 54–65.

Neri, F., & Tirronen, V. (2009). Scale factor local search in differential evolution. *Memetic Computing*, *1*, 153–171.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. Springer.

Noman, N., & Iba, H. (2008). Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, *12*(1), 107–125.

Parker, J., Khoogar, A., & Goldberg, D. (1989). Inverse kinematics of redundant robots using genetic algorithms. In *Proc. IEEE Conference on Robotics and Automation*, vol. 1, (pp. 271–276).

Paul, R. P., & Shimano, B. (1978). Kinematic control equations for simple manipulators. In *IEEE Conference on Decision and Control and* 17^{th} *Symposium on Adaptive Processes*, vol. 17, (pp. 1398–1406).

Phuoc, L., Martinet, P., Lee, S., & Kim, H. (2008). Damped least square based genetic algorithm with Gaussian distribution of damping factor for singularity-robust inverse kinematics. *Journal of Mechanical Science and Technology*, *22*, 1330–1338.

Price, K. (1999). *New ideas on optimization*, chap. An introduction to differential evolution, (pp. 79–106). McGraw-Hill.

Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential Evolution. A practical approach to global optimization*. Natural Computing. Springer-Verlag.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, *13*, 398–417.

Raghaven, M., & Roth, B. (1990). Kinematic analysis of the 6R manipulator of general geometry. In *The Fifth International Symposium on Robotics Research*, (pp. 263–269). Cambridge, MA, USA: MIT Press.

Rahnamayan, S., Tizhoosh, H., & Salama, M. (2008). Opposition-based differential evolution. In U. Chakraborty (Ed.) *Advances in differential evolution*, vol. 143 of *Studies in Computational Intelligence*, (pp. 155–171). Springer Berlin / Heidelberg.

Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. A. (2007). A novel population initialization method for accelerating evolutionary algorithms. *Computers and Mathematics with Applications*, *53*, 1605–1614.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Stuttgart, Germany: Frommann-Holzboog.

Rechenberg, I. (1994). *Evolutionsstrategies 9́4 werkstatt bionik und evolutionstechnik*. Frommann-Holzboog.

Ronkkonen, J., Kukkonen, S., & Price, K. (2005). Real-parameter optimization with differential evolution. In *IEEE Congress on Evolutionary Computation, CEC'05*, vol. 1, (pp. 506–513).

Rothlauf, F. (2006). *Representations for genetic and evolutionary algorithms*. Springer-Verlag, second ed.

Schwefel, H. (1975). *Evolutionsstrategie und numerische optimierung*. Ph.D. thesis, Technische Universität Berlin.

Schwefel, H. (1981). *Numerical optimization of computers models*. Chicherter, UK: John Wiley.

Sciavico, L., & Siciliano, B. (1996). *Modeling and control of robot manipulators*. McGraw-Hill.

Shadow (2009). *Shadow Dexterous Hand C6M – Technical Specification*. Shadow Robot Company LTD., 251 Liverpool Road, London. UK.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, *12*(6), 702–713.

Solis, F., & Wets, R. (1981). Minimization by random search techniques. *Mathematical Methods of Operations Research*, *6*(1), 19–30.

Somolinos, J. A., Feliu, V., & Sánchez, L. (2002). Design, dynamic modelling and experimental validation of a new three-degree-of-freedom flexible arm. *Mechatronics*, *12*(7), 919–948.

Storn, R., & Price, K. (1997). Differential Evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.

Tabandeh, S. (2009). *Development of novel task-based configuration optimization methodologies for modular and reconfigurable robots using multi-solution inverse kinematic algorithms*. Ph.D. thesis, University of Waterloo.

Tabandeh, S., Clark, C., & Melek, W. (2006). A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. In *Proc. IEEE Congress on Evolutionary Computation, CEC'06*, (pp. 1815–1822).

Tabandeh, S., Melek, W. W., & Clark, C. M. (2010). An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots. *Robotica*, *28*, 493–507.

Thangaraj, R., Pant, M., & Abraham, A. (2010). New mutation schemes for differential evolution algorithm and their application to the optimization of directional over-current relay settings. *Applied Mathematics and Computation*, (pp. 532–544).

Vasilyev, I., & Lyashin, A. (2010). Analytical solution to inverse kinematic problem for 6-DOF robot-manipulator. *Automation and Remote Control*, *71*, 2195–2199.

Wampler, C. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, *16*(1), 93–101.

Wampler, C., & Morgan, A. (1991). Solving the 6R inverse position problem using a generic-case solution methodology. *Mechanism and Machine Theory*, *26*(1), 91 – 106.

Wang, L., & Chen, C. (1991). A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, *7*(4), 489–499.

Whitney, D. (1969a). Optimum step size control for Newton-Raphson solution of nonlinear vector equations. *IEEE Transactions on Automatic Control*, *14*(5), 572–574.

Whitney, D. (1969b). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man Machine Systems*, *10*(2), 47–53.

Wolovich, W. A., & Elliott, H. (1984). A computational technique for inverse kinematics. In *The* 23^{rd} *IEEE Conference on Decision and Control*, vol. 23, (pp. 1359–1363).

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Yang, Y., Peng, G., Wang, Y., & Zhang, H. (2007). A new solution for inverse kinematics of 7-DOF manipulator based on genetic algorithm. In *IEEE International Conference on Automation and Logistics*, (pp. 1947–1951).

Yoshikawa, T. (1991). Translational and rotational manipulability of robotic manipulators. In *Proc. Int. Conf. on Industrial Electronics, Control and Instrumentation, IECON '91*, (pp. 1170–1175).

Zaharie, D. (2002). Critical values for the control parameters of differential evolution algorithm. In R. Matouek, & P. Omera (Eds.) *Proc. of Mendel 2002,* 8^{th} *International Conference on Soft Computing*. Brno, Czech Republic.

Zaharie, D. (2003). Control of population diversity and adaptation in differential evolution algorithms. In R. Matousek, & P. Osmera (Eds.) *Proc. of Mendel* 9^{th} *International Conference on Soft Computing*, (pp. 41–46). Brno, Czech Republic.

Zhang, J., & Sanderson, A. (2007). JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In *IEEE Congress on Evolutionary Computation, 2007. CEC'07*, (pp. 2251–2258).

Zhang, P.-Y., Lü, T.-S., & Song, L.-B. (2005). RBF networks-based inverse kinematics of 6R manipulator. *The International Journal of Advanced Manufacturing Technology*, *26*, 144–147.