Universidad Veracruzana
Centro de Investigación en Inteligencia Artificial

A Metaheuristic Based on the Center of Mass

THESIS

by

Jesús-Adolfo Mejía-de-Dios

supervised by
Dr. Efrén Mezura-Montes

Xalapa, Ver., México. August 2, 2018

Acknowledgments

First, I would like to thank my advisor, Dr. Efrén Mezura-Montes for supporting and giving me the complete freedom in choosing what I wanted to work on.

The development of this work would not have been possible without the constant love and support from my beloved Rocío S.G.

I would like to thank my family for the continuous support.

I want to say thank you to all my professors who helped me achieve this work.

Abstract

This work presents a novel population-based metaheuristic inspired by some principles from physics and mechanics, which is called Evolutionary Centers Algorithm (ECA). We utilize the center of mass definition for creating new directions to move the worst elements in the population, based on their objective function values, to better regions of the search space. First, we give some important definitions, properties and results about global optimization. Next, a preliminary version of the algorithm is exposed and an empirical study is given by using a recent competition benchmark. After that, our proposal is improved with adaptive mechanisms and we present a comparison against representative algorithms for global optimization. Finally, our approach is applied in three engineering optimization problems from mechatronics and computer vision with relevant results in terms of efficiency and accuracy.

Contents

List of Figures

List of Tables

Introduction

Nowadays, real-world problems in different areas (science, engineering, etc.) can be solved by transforming them to an *optimization* problem. An optimization problem can be defined by finding the set of all feasible solutions and a measure for evaluating their quality. There are many optimization algorithms derived from intense research activity. They are, however, still limited in their reach [90]. There exists several numerical methods for solving real-parametric optimization problems, for example Newton-like or quasi-Newton algorithms based on gradient information of objective function. That type of algorithms can not be applied on non-continuous or non-differentiable functions. Here, Metaheuristics take place, some of these algorithms are based on a population and assumptions about objective function are not required.

The design of new metaheuristics is an important topic due to real-world problems can be complex to solve with exact methods and it is well known that no algorithm can be universally more efficient than other algorithms. Thus, it is necessary design new methods to successfully solve more problems. Nevertheless, some authors do not agree novel metaheuristics, some of them insure that almost the novel proposal are the same process but with a different name [77]; that is a valid opinion but lacking in mathematical support, i.e., there are no results (theorems, prepositions, etc.) where the affirmation "*all (new) metaheuristics converges into a single process with same dynamic of search*" comes true. The dynamic of search is most important procedure in metaheuristics, then it is important to detail it in formal terms to complement the further empirical analysis in order to strengthen the proposal.

Hypothesis

It is possible to propose a new physic-inspired algorithm for real-parametric optimization problems such that it is simple (few parameters) but efficient (in terms of the objective function evaluations) and scalable.

Objectives

The general objective is to design a new metaheuristic for real-parametric single objective optimization with competitive results compared others in state-of-the-art on representative test functions for optimization.

Specific Objectives

- Make a literature review in order to identify important properties about optimization problems.
- Get inspired to design a new algorithm for solving optimization problems.
- Investigate a set of test functions from specialized literature.
- Implement a preliminary version of our proposal.
- Compare our approach with state-of-the-art algorithms.
- Implement a self-adaptive technique for the algorithm to improve results on high-dimensional problems.
- Solve real-world problems using our proposal.

Motivation

Metaheuristics have provided successful results when solving complex bound-constrained optimization problems [80]. However, most popular metaheuristics usually are those which design keeps simple and their number of parameters is low so as to facilitate the fine-tuning process when a particular problem is solved. We propose a new algorithm for real-parameter single-objective optimization. This algorithm will be simple (few parameters) but efficient and scalable. Our approach is based on the concept of center of mass, called Evolutionary Centers Algorithm (ECA).

To propose a successful metaheuristic for a given optimization problem, it is necessary supply a balance between the exploration (diversification) and the exploitation (intensification) in the search. The exploration process is focused on identifying regions of the search space with better solutions. Exploitation process intensifies the search in some promising areas by using the accumulated search experience. Different metaheuristics are proposed in the specialized literature, and the main differences among them is the different form they handle this balance [2, 13].

On the other hand, emerging algorithms, such as physics-based algorithms are gaining interest when solving optimization problems [11]. However, when dealing constrained search spaces they are not as preferred as other approaches.

Considerations must be made with regard to the search space, the types of objective functions, the properties of the design space, etc. In this dissertation, the following assumptions are made:

- The objective function is no-negative (not a restrictive assumption).
- Search space is continuous ($\mathbb{R}^D$ is considered).
- Only single-objective optimization is considered (at first).

Contents

The thesis consists of 5 chapters and two appendices. In the first Chapter 1, an introduction to the problem and the theory is given. Chapter 2 presents a novel optimization algorithm based on some principles from physic concepts called ECA. In Chapter 3, our approach is used to solve engineering design problems; a comparison against competitive metaheuristics is made. In Chapter 4, we describe the application of ECA to an image processing problem called Point-set Registration Problem. Finally, the conclusions and future work are given in Chapter 5. In Appendix A the

CEC17 benchmark is described. The statistical results obtained by the experimentation are reported in Appendix B.

Publications and Presentations

Here, we list the publications and presentations in different academic events derived from this work:

Publication

1. Jesús Adolfo-Mejía-De-Dios and Efrén Mezura-Montes, **A New Evolutionary Optimization Method Based on Center of Mass**, in Proceedings of the International Conference on Recent Trends in Operations Research and Statistics (RTORS), Springer proceedings (accepted), 2017.

Presentations

- Jesús Adolfo-Mejía-De-Dios and Efrén Mezura-Montes, **Evolución Cooperativa Metaheurística para Problemas de Optimización (poster)**, Escuela de Optimización, Simulación y Métodos Numéricos en Robótica. CIMAT, Guanajuato 2017.

- Jesús Adolfo-Mejía-De-Dios and Efrén Mezura-Montes, **Metaheurística para Optimización**, VII Foro Nacional de Divulgación Científica Y Tecnológica, FODICYT, Universidad Veracruzana 2017.

- Jesús Adolfo-Mejía-De-Dios and Efrén Mezura-Montes, **Affine Image Registration Transformation Estimation Using Metaheuristic for Optimization**, Congreso Sur Sureste de Matemáticas, Casa Matemática Oaxaca, 2017.

- Jesús Adolfo-Mejía-De-Dios and Efrén Mezura-Montes **Optimización Binivel y sus Implicaciones**, Coloquio de Matemáticas Aplicadas SUNEO, 2018.

Chapter 1

Optimization

Some theory of this chapter is based on [17, 71, 79, 90].

1.1 Global Optimization Problems

We start stating the *global optimization problem* definition.

Definition 1.1.1. Given a function $f : X \subset \mathbb{R}^n \to \mathbb{R}$ with $X \neq \emptyset$, for $\boldsymbol{x}^* \in X$ the value $f^* := f(\boldsymbol{x}^*) > -\infty$ is called a global minimum if and only if

$$\forall \boldsymbol{x} \in X, \;\; f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}).$$

Then, $\boldsymbol{x}^*$ is a global minimum point, f is called objective function and the set X is called the feasible region. The problem of determining a global minimum point is called the global optimization problem.

Without loss of generality, we will concentrate on minimization problems, since the identity for an upper bounded function:

$$\max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in X\} = -\min\{-f(\boldsymbol{x}) \mid \boldsymbol{x} \; \in X\}$$

holds.

If $\boldsymbol{x}^*$ is global minimum of f over X , often we write $f(\boldsymbol{x}^*) = \min_{\boldsymbol{x} \in X} f(x)$. Equivalently, optimization problems can be represented as finding the set:

$$X^* = \arg\min_{\boldsymbol{x} \in X} f(\boldsymbol{x}) = \{\boldsymbol{x}^* \in X \; : \; f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in X\}, \qquad (1.1)$$

where, X is a D-dimensional vector space of parameters, usually $X \subset \mathbb{R}^D$ is the domain for $\boldsymbol{x}$ representing constraints on allowable values for $\boldsymbol{x}$. Equation 1.1 may be read as: X^* is the set of values (arguments) $\boldsymbol{x} = \boldsymbol{x}^*$ that minimize $f(\boldsymbol{x})$ subject to X^*.

The solution set X^* in Equation 1.1 is non-empty, then it can be a unit set, a set with countable or uncountable number of elements. Next examples illustrate these type of solutions sets X^*.

Example 1.1.1. X^* is a unit set (contains unique solution). Suppose that

$$f(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2, \text{ with } \boldsymbol{x} = (x_1, \ldots, x_D) \in X \subset \mathbb{R}^D$$

The unique value that minimizes f is $\boldsymbol{x} = \mathbf{0} = (0, \ldots, 0)$. Thus, X^* is a unit set, i.e. $X^* = \{\mathbf{0}\}$.

Example 1.1.2. X^* has countable number of points. Let $f : \mathbb{R} \to \mathbb{R}$ be a scalar function, defined as $f(x) = \cos(x)$. If $X = [-2\pi,\ 2\pi]$, then $\cos(x) = -1$ at the points $X^* = \{-2\pi, 0, 2\pi\}$ which is a countable set with a finite number of elements. On the other hand, if $X = \mathbb{R}$ then $X^* = \{2\pi n\ :\ n \in \mathbb{Z}\}$ a countable set with an infinite number of elements.

Example 1.1.3. X^* has uncountable number of points. Suppose, $D \geq 2$ and

$$f(\boldsymbol{x}) = \left(-1 + \sum_{i=1}^{D} x_i^2\right)^2 \text{ and } X = \mathbb{R}^D.$$

This objective function is minimized when $\sum_{i=1}^{D} x_i^2 = 1$, which is a D-dimensional sphere having radius 1. Hence, X^* is an uncountable set.

An optimization problem is solved only when a global minimum is found. However, global minimum are, in general, difficult to find. Therefore, in practice, we often have to find at least a local minimum. On the other hand, to formalize the notion of a local minimum a distance measure is needed for an arbitrary vector space.

Definition 1.1.2. Let V be a vector space. A metric (distance) on V is a mapping $d : V \times V \to \mathbb{R}_0^+$, such that for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$ the following conditions are satisfied:

$$\begin{array}{rcl} d(\boldsymbol{x}, \boldsymbol{y}) & \geq & 0 \\ d(\boldsymbol{x}, \boldsymbol{y}) & = & 0 \text{ if and only if } \boldsymbol{x} = \boldsymbol{y} \\ d(\boldsymbol{x}, \boldsymbol{y}) & = & d(\boldsymbol{y}, \boldsymbol{x}) \\ d(\boldsymbol{x}, z) & \leq & d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}), \end{array}$$

where $\mathbb{R}_0^+ = [0,\ \infty)$. Usually, is convenient use norm instead metrics in vector spaces.

Definition 1.1.3. Let V a vector space on $\mathbb{R}$. A mapping $\|\cdot\| : V \to \mathbb{R}_0^+$ is called norm if for $\boldsymbol{x}, \boldsymbol{y} \in V$, $\alpha \in \mathbb{R}$:

$$\begin{array}{rcl} \|\boldsymbol{x}\| & = & 0, \text{ iff } \boldsymbol{x} = 0 \\ \|\alpha \boldsymbol{x}\| & = & |\alpha| \|\boldsymbol{x}\| \\ \|\boldsymbol{x} + \boldsymbol{y}\| & \leq & \|\boldsymbol{x}\| + \|\boldsymbol{y}\|. \end{array}$$

It is well known than any normed space is also a metric space by setting

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|.$$

The inversion of this implication does not hold. Usually, when $V = \mathbb{R}^n$ is considered, the Euclidean norm is presupposed, i.e., if $\boldsymbol{x} = (x_1,\ x_2,\ \ldots,\ x_n) \in \mathbb{R}^n$:

$$\|\boldsymbol{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

After these definitions, the meaning of a local minimum is defined as:

Definition 1.1.4. (**Local minimum**) For $\hat{\boldsymbol{x}} \in X$ the value $\hat{f} := f(\hat{\boldsymbol{x}})$ is called a local minimum if and only if exists $\varepsilon > 0$ such that for all $\boldsymbol{x} \in X$

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\| < \varepsilon \text{ implies } \hat{f} \leq f(\boldsymbol{x})$$

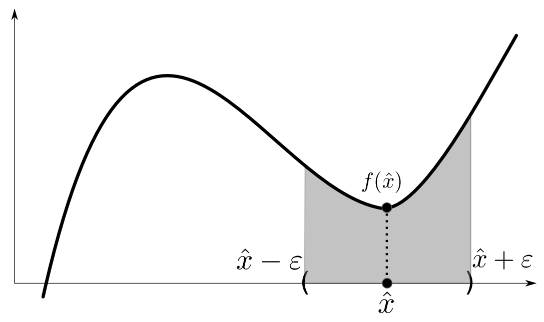


Figure 1.1: Local minimum definition representation.

Note that any global minimum is also a local one. Figure 1.1 presents a representation for definition of local minimum.

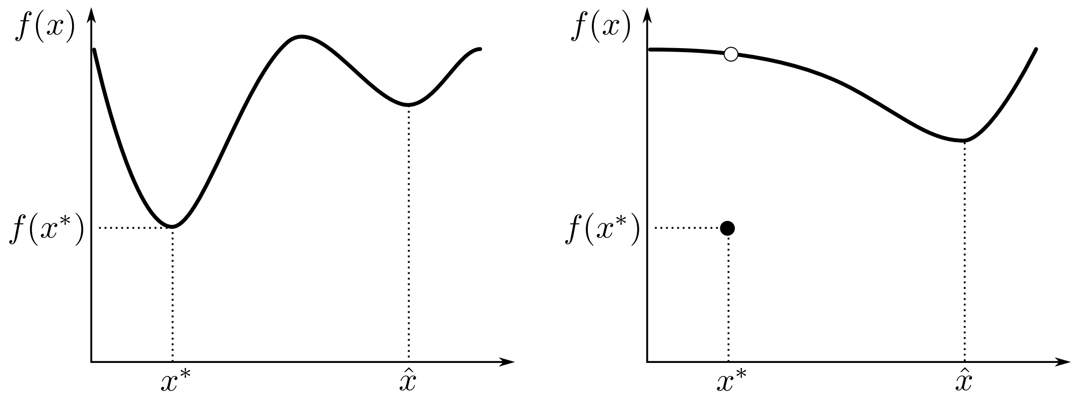
In practice, there exists a problem with Definition 1.1.4 because limited exactness of floating-point representation of real numbers by computers. Therefore, the global optimization problem can be considered as solved if a member of the level set

$$L_{f^*+\varepsilon} = \{\boldsymbol{x} \in M | f(\boldsymbol{x}) \le f(\boldsymbol{x}^*) + \varepsilon\}.$$

Unfortunately, [83] probed that global optimization problem with continuous objective function on a compact feasible region within a finite number of steps is unsolvable. On this way, proposing new methods for global optimization problems is necessary.

For a case where the problem dimension $D = 1$, Figure 1.2 illustrates two problems with distinct local and global minimum. This example illustrates that, in general, obtaining a global solution requires knowledge about the possible existence of such point.

Definition 1.1.5. (Unimodal and Multi-modal Functions) Let $X \subset \mathbb{R}$ a connected set, i.e., it cannot be divided into two disjoint nonempty closed sets. An objective function f on X is called unimodal if it has exactly one local minimum, otherwise it is called multi-modal.



(a) Easy Problem (b) Hard Problem

Figure 1.2: Easy problem and hard problem for global optimization. In (a), a global algorithm will easily avoid $\hat{\boldsymbol{x}}$ and find $\boldsymbol{x}^*$. In (b) a global algorithm will only find $\hat{\boldsymbol{x}}$ since it is effectively unlikely to find $\boldsymbol{x}^*$, since there is nothing near $\boldsymbol{x}^*$ to indicate the presence of a minimum.

For example, the sphere model is a continuous convex unimodal function. We present the mathematical formulation:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} (x_i - c_i)^2,$$

where $\boldsymbol{x},\ \boldsymbol{c} \in \mathbb{R}^n$ with $\boldsymbol{x} = (x_1,\ \dots,\ x_n)$ and $\boldsymbol{c} = (c_1,\ \dots,\ c_n)$. Note that $\boldsymbol{x}^* = \boldsymbol{c}$, $f^* = 0$ for $n \in \mathbb{N}$ and $\boldsymbol{x},\ \boldsymbol{c} \in [a,\ b]^n$.

On the other hand, Ackley's function is a continuous multi-modal non-convex function obtained adding cosine waves to sphere function. General formulation is:

$$f(\boldsymbol{x}) = \alpha n + \sum_{i=1}^{n} [x_i^2 - \alpha \cos(2\pi x_i)],$$

where $\alpha = 10$. It has global optimum $\boldsymbol{x}^* = 0$ and $f^* = 0$.

1.1.1 Constrained Optimization

Some real-world problems have constraints. This kind of problems are called Constrained Optimization Problems (COPs). Without loss of generality, a COP can be defined as to:

Definition 1.1.6. (**Constrained Optimization Problem**) A general constrained optimization problem can be written as follows:
Minimize:

$$f(\boldsymbol{x}),\ \boldsymbol{x} \in S \subseteq \mathbb{R}^D \tag{1.2}$$

subject to:

$$g_i(\boldsymbol{x}) \leq 0, \qquad i = 1, \dots, p \tag{1.3}$$

$$h_j(\boldsymbol{x}) = 0, \qquad j = p+1, \dots, m \tag{1.4}$$

where $S = \prod_{k=1}^{D} [x_{k,\min},\ x_{k,\max}]$ i.e. $x_k \in [x_{k,\min},\ x_{k,\max}]$ for $k = 1, 2, \dots, D$. The problem is subject to p inequality constraints and $m - p$ equality constraints. If $\boldsymbol{x}$ satisfies $g_i(\boldsymbol{x}) \leq 0$, for $i = 1, \dots, p$ and $|h_j(\boldsymbol{x})| \leq \varepsilon$, for $j = p+1, \dots, m$ with $\varepsilon > 0$ a small value; then $\boldsymbol{x}$ is regarded feasible.

Many unconstrained optimization algorithms can be transformed to an constrained one, usually by using Deb's rules or penalty method [24]. The reader is referred to [21, 71].

Next section gives some important facts related to the existence of optimal values.

1.1.2 Conditions for Local Minima

This Section presents some conditions for guaranteeing local minima for unconstrained optimization [21].

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous and derivable function. The first-order derivate of f, denoted Df is:

$$Df = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n},\right].$$

The gradient ∇f is defined as $\nabla f = (Df)^T$. The second derivate (Hessian) of f, is given by:

$$D^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_1} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_n} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Given an optimization problem with constraints set $X \in \mathbb{R}^n$, such that exists $\hat{\boldsymbol{x}} \in X$, the next definition gives a notion of feasible directions.

Definition 1.1.7. A non-zero vector $\boldsymbol{v} \in \mathbb{R}^n$ is a feasible direction at $\boldsymbol{x} \in X$ if there exists $\eta_{\max} > 0$ such that $\boldsymbol{x} + \eta \boldsymbol{v} \in X$ for all $\eta \in [0,\ \eta_{max}]$.

The directional derivate of f in the direction $\boldsymbol{v}$ is the real-valued function defined as:

$$\frac{\partial f}{\partial \boldsymbol{v}}(\boldsymbol{x}) = \lim_{\eta \to 0} \frac{f(\boldsymbol{x} + \eta \boldsymbol{v}) - f(\boldsymbol{x})}{\eta}.$$

To compute the directional derivate above, suppose that $\boldsymbol{x}$ and $\boldsymbol{v}$ are given. Then $f(x + \eta \boldsymbol{v})$ is function of η and

$$\frac{\partial f}{\partial \boldsymbol{v}}(\boldsymbol{x}) = \frac{d}{d\eta} f(\boldsymbol{x} + \eta \boldsymbol{v}) \bigg|_{\eta=0}.$$

By using the chain rule:

$$\frac{\partial f}{\partial \boldsymbol{v}}(\boldsymbol{x}) = \frac{d}{d\eta} f(\boldsymbol{x} + \eta \boldsymbol{v}) \bigg|_{\eta=0} = \nabla f(\boldsymbol{x})^T \boldsymbol{v} = \boldsymbol{v} \nabla f(\boldsymbol{x})^T.$$

Note 1.1.1. If $\boldsymbol{v} \in \mathbb{R}^n$ such that $\|\boldsymbol{v}\| = 1$, then $\boldsymbol{v} \nabla f(\boldsymbol{x})^T$ is the rate of increase of f at the point $\boldsymbol{x}$ in the direction $\boldsymbol{v}$.

Now, we can state the following theorem, which is a necessary for giving conditions to guarantee the existence of local minimum.

Theorem 1.1.1. *Let $X \subset \mathbb{R}^n$ be a set and f a continuous and derivable function on X. If $\hat{\boldsymbol{x}} \in X$ is a local minimum of f, then for any feasible direction $\boldsymbol{v}$ at $\hat{\boldsymbol{x}}$, we have*

$$\boldsymbol{v^T} \nabla f(\hat{\boldsymbol{x}}) = \frac{\partial f}{\partial \boldsymbol{v}} \geq 0.$$

The proof of this result can be found in [71].

In other words, if $\hat{\boldsymbol{x}}$ is a local minimum, then the rate of increase of f at $\hat{\boldsymbol{x}}$ in any feasible direction $\boldsymbol{v} \in X$ is nonnegative. The special case, when $\hat{\boldsymbol{x}} \in X - \partial X$, where ∂X is the border of X. In this case, the next corollary establishes that any direction is feasible.

Corollary 1.1.2. *Let $X \subset \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ a continuous and derivable function on X, if $\hat{\boldsymbol{x}}$ is a local minimum of f over X and if $\hat{\boldsymbol{x}}$ is an interior point of X, then*

$$\nabla f(\hat{\boldsymbol{x}}) = 0.$$

Proof. Suppose that f has a local minimum at $\hat{\boldsymbol{x}}$ which is an interior point of X. Since $\hat{\boldsymbol{x}}$ is an interior point of X, the set of feasible directions at $\hat{\boldsymbol{x}}$ is the whole of $\mathbb{R}^n$. Thus, for any $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{v}^T \nabla f(\hat{\boldsymbol{x}}) \geq 0$ and $-\boldsymbol{v}^T \nabla f(\hat{\boldsymbol{x}}) \geq 0$. Hence, $\boldsymbol{v}^T \nabla f(\hat{\boldsymbol{x}}) = 0$ for all $\boldsymbol{v} \in \mathbb{R}^n$, which implies that $\nabla f(\hat{\boldsymbol{x}}) = 0$. $\square$

Now, a second-order necessary condition can be derived for a local minimum in the feasible set.

Theorem 1.1.3. *Let $X \subset \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ a with second derivate on X, $\hat{\boldsymbol{x}}$ a local minimum of f over X, and v a feasible direction at $\hat{\boldsymbol{x}}$. If $\boldsymbol{v}^T \nabla f(\hat{\boldsymbol{x}}) = 0$, then:*

$$\boldsymbol{v}^{\boldsymbol{T}} D^2 f(\hat{\boldsymbol{x}}) \geq 0,$$

where $D^2 f$ is the Hessian of f.

An immediate consequence is given by the following Corollary.

Corollary 1.1.4. *Let $\hat{\boldsymbol{x}}$ be an interior point of $X \subset \mathbb{R}^n$. If $\hat{\boldsymbol{x}}$ is a local minimum of $f : X \to \mathbb{R}$, then*

$$\nabla f(\hat{\boldsymbol{x}}) = 0,$$

and $D^2 f(\hat{\boldsymbol{x}})$ is a positive semidefinite matrix ($D^2 f(\hat{\boldsymbol{x}}) \geq 0$), i.e. for all $\boldsymbol{v} \in \mathbb{R}^n$,

$$\boldsymbol{v}^{\boldsymbol{T}} D^2 f(\hat{\boldsymbol{x}}) \boldsymbol{v} \geq 0.$$

Proof. If $\hat{\boldsymbol{x}}$ is an interior point, then all directions are feasible. The result then follows from Corollary 1.1.2 and Theorem 1.1.3. $\square$

Next theorem give us sufficient conditions that imply that $\hat{\boldsymbol{x}}$ is a local minimum and grantee the existence.

Theorem 1.1.5. *Let f be a function with second derivate, defined on a region in which $\hat{\boldsymbol{x}}$ is an interior point. Suppose that*

1. $\nabla f(\hat{\boldsymbol{x}}) = 0.$
2. $D^2 f(\hat{\boldsymbol{x}}) = 0.$

Then, $\hat{\boldsymbol{x}}$ is a strict local minimum of f.

In summary, a theoretical basis for the solution of nonlinear unconstrained problems was presented. Also, some conditions for optimality were given. However, in real-world problems some assumptions cannot be archived and we need to propose methods for general objective functions. This work focuses on problems where f is sufficiently complex such that it is not possible to obtain the solution of Equation 1.1 by analytical methods.

The next section presents some theorems about algorithms for solving optimization problems.

1.1.3 No Free Lunch Theorems

Here, some *No Free Lunch* (NFL) theorems are given which state that no algorithm can be universally more efficient than other algorithms [89]. These theorems gives an interesting interpretation of what it means for an algorithm to be very appropriate to an optimization problem.

Optimization algorithms on digital computers are restricted on finite sets X and Y which are some 32 or 64 bit representation of real numbers. An objective function is defined as $f : X \to Y$ and the space of all possible objective functions

$$\mathcal{F} = X^Y = \{f : X \to Y\}.$$

Suppose $|X|$ and $|Y|$ are the size of X and Y, respectively. Note, $\mathcal{F}$ is of size $|X|^{|Y|}$ a finite but large number. Objective functions that depends explicitly on time are considered in this part.

Definition 1.1.8. (Sample) A sample is a time-ordered of m distinct visited solutions defined by the following:

$$d_m = \{(d_x(i),\ d_y(i)\ :\ i = 1, 2, \ldots, m\},$$

where $d_x(i) \in X$ and $d_y(i) \in Y$ are ith successive element.

The space of all samples of size m is $D_m \subset X \times Y$, that is, $d_m \in D_m$. Now, we can define an optimization algorithm in term of samples.

Definition 1.1.9. (**Optimization Algorithm**) An optimization algorithm $\mathcal{A}$ is an injective function from visited samples to single new solution in X. Formally, for $m \in \mathbb{N}$

$$\mathcal{A} : D_m \to X$$

Popular examples of optimization algorithms are evolutionary algorithms, simulated annealing or other called metaheuristics (See Section 1.2). On the other hand, this definition only considers deterministic optimization algorithm. Since, all algorithm implemented on a computer are deterministic [89]. Hence, the Definition 1.1.9 is not restrictive.

Now, the first NFL theorem in terms of probability is given. If $f \in \mathcal{F}$, then the distribution $P(f) = P(f(x_1), \ldots, P(x_{|X|}))$.

Theorem 1.1.6. *For any pair of algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$:*

$$\sum_f P(d_y|f, \mathcal{A}_1) = \sum_f P(d_y|f, \mathcal{A}_2).$$

A proof of this result can be found in [89]. This theorem establishes that an algorithm obtain a good performance on a set of problems is necessarily offset by its performance on the remaining optimization problems. Hence, all algorithms have the same f-averaged performance on all optimization problems defined over $\mathcal{F}$.

The best way to evaluate the performance of an algorithm in time-dependent objective functions is not trivial. Authors regard two ways based on manipulations of the sample definition. In first way, the values of Y in $d_y(j)$ related to a particular $d_x(j)$ is achieved by the objective function value that was obtained when $d_x(j)$ was sampled.

In contrast, for second way, a sample d_y obtained by the Y values from the current objective function for each $\boldsymbol{x} \in d_x$. That is, if $d_x = \{d_x(1), \dots, d_x(m)\}$, then in the first way:

$$d_x = \{f_1(d_x(1)), \dots, T_{m-1}(f_{m-1})(d_x(m))\}$$

and in scheme 2

$$D_y = \{f_m(d_x(1)), \dots, f_m(d_x(m))\}$$

where $f_m = T_{m-1}f_{m_1}$. is the final objective function value.

Theorem 1.1.7. *For all d_y, D_y, $m > 1$ algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, and initial objective functions f_1*

$$\sum_T P(d_y \ : \ f_1, T, \mathcal{A}_1) = \sum_T P(d_y \ : \ f_1, T, \mathcal{A}_2)$$

and

$$\sum_T P(d_y \ : \ f_1, T, \mathcal{A}_1) = \sum_T P(d_y \ : \ f_1, T, \mathcal{A}_2)$$

Thus, in particular, if an algorithm outperforms another for set of dynamic objective function $\mathcal{F}_1$, then the reciprocal must be true on the complement set of $\mathcal{F}_1$.

In summary, the NFL theorems state that if an optimization algorithm has a good average performance for a kind of problems then it must be outperformed on average by other algorithm over the remaining problems.

Moreover, there are several numerical methods for solving real-parametric optimization problems, for example Newton-like or quasi-Newton algorithms based on gradient information of objective function. That type of algorithms can not be applied on non-continuous or non-differentiable functions. Here, Metaheuristics take place, some of these algorithms are based on populations and assumptions about objective function are not important.

1.2 Metaheuristics

A metaheuristic is an algorithm created to solve approximately a different kind of difficult optimization problems without having to adapt deeply to each problem. In fact, the Greek prefix "meta", present in the name, indicates that these algorithms are "high-level" procedure or heuristic designed to generate a heuristic. Metaheuristics generally are applied to problems for which there is no specific satisfactory algorithm to solve them. They have been successfully used to solve complex problems in science, engineering or industry; for example: in areas such finance to production, management or economics [13].

Metaheuristics usually are inspired on nature, some principles from physics, biology or ethology. These algorithms can be stochastic and do not make strong assumptions about the objective function [2, 9].

To propose a successful metaheuristic for a given optimization problem, it is necessary supply a balance between the exploration (diversification) and the exploitation (intensification). Exploration process is focused for identifying parts of the search space with high quality solutions. Exploitation process intensifies the search in some promising areas by using the accumulated search experience.

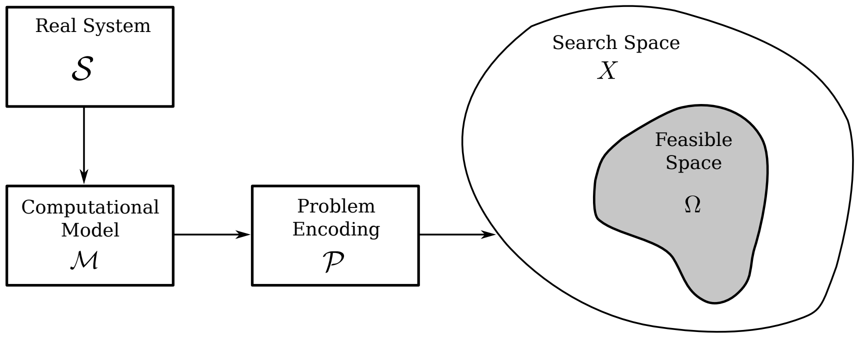


Figure 1.3: Diagram of the process to define an optimization problem from real system.

Distinct metaheuristics are proposed, and the main differences among them is the particular way in which they handle this balance [2, 13]. Figure 1.3 illustrates the sequence for solving real-world optimization problem through optimization algorithms.

As mentioned, there are a wide variety of metaheuristics and a number of properties useful to classify them.

- **Single-solution Based Metaheuristics** This kind of algorithms are also called trajectory methods. Unlike population-based metaheuristics, they start with a single initial solution and move through the search space while describes a trajectory. Some of them can be seen as intelligent extensions of local search algorithms. Representative algorithms of this type are, simulated annealing method [49], tabu search [36], greedy randomized adaptive search procedure (GRASP) [29], variable neighborhood search [61], guided local search [85], iterated local search [55], and their variants.

- **Population-based Metaheuristic** Population-based metaheuristics employ a set of solutions (population) instead using a single solution. The most representative population-based methods are related to Swarm Intelligence (SI), Evolutionary Computation (EC) and emerging algorithms like those physics-inspired. EC algorithms are inspired by biological evolution, where new population individuals generated by using recombination and mutation operators. In SI, the idea is to emulate "intelligent" behavior with simple rules to generate new positions (solutions) for the agents. Physics-inspired algorithms adopt the laws of physics; they are successful used to solve complex optimization problems. For details, the reader is referred to [13].

Next Sections describe evolutionary algorithms and physics-inspired which are important to describe our approach.

1.2.1 Evolutionary Algorithms

Definition 1.2.1. (General Evolutionary Algorithm) An evolutionary algorithm (EA) is defined as an 8-tuple:

$$EA = (I, \Phi,\ \Omega,\ \Psi,\ s,\ l,\ \mu,\ \lambda),$$

where $I = A_x \times A_s$ is the space of individuals, and $A_x,\ A_s$ denotes arbitrary sets. $\Phi : I \to \mathbb{R}$ denotes a fitness function assigning real values to individuals.

$$\Omega = \{\omega_{\Theta_1}, \dots, \omega_{\Theta_x} | \omega_{\Theta_i} : I^\lambda \to I^\lambda\} \cup \{\omega_{\Theta_0} : I^\mu \to I^\lambda\}$$

is a set of probabilistic genetic operators ω_{Θ_i}, each one controlled by specific parameters summarized in the sets $\Omega_i \subset \mathbb{R}$.

$$s_{\theta_s} : I^\lambda \cup I^{\mu+\lambda} \ \to I^\mu$$

denotes the selection operator, which may change the number of individuals from λ or $\lambda + \mu$ to μ, where $\mu, \lambda \in \mathbb{N}$.

An additional set Θ_s of parameters may be used by the selection operator. μ is the number of parents individuals, while λ denotes the number of *offspring* individuals. Finally $l : I^\mu \to \{true, false\}$ is a determination criterion for the EA, and the generation $\Psi : I^\mu \to I^\mu$ describes the complete process of transforming a population P into a subsequent one by applying genetic operators and selection.

$$\begin{aligned} \Psi &= s \circ \omega_{\Theta_{i_1}} \circ \dots \circ \omega_{\Theta_{i_j}} \circ \omega_{\Theta_0} \\ \Psi(P) &= s_{\Theta_s}(Q \cup w_{\Theta_{i_1}}(\dots(\omega_{\Theta_{i_j}}(\omega_{\Theta_0}(P)))\dots)). \end{aligned}$$

Here $\{i_1, \dots, i_j\} \subset \{1, \dots, z\}$ and $Q \in \{\emptyset, P\}$

Definition 1.2.2. Given an Evolutionary Algorithm with generation transition $\Psi : I^\mu \to I^\mu$ and an initial population $P(0) \in I^\mu$, the sequence $P(0), P(1), P(2), \dots$ is called a population sequence or evolution of $P(0)$ if and only if for all $t \geq 0$

$$P(t+1) = \Psi(P(t)).$$

Usually, population $P(0)$ is initialized at random but there are works about initializing the population using *a priori* information [4, 70].

Definition 1.2.3. (Panmictic) A genetic operator $\omega_\Theta : I^p \to I^q$ is called panmictic iff exists $\omega'_\Theta : I^p \to I$ such that

$$\omega_\Theta(a_1, \dots, a_p) = \underbrace{(\omega'_\Theta(a_1, \dots, a_p), \dots, \omega'_\Theta(a_1, \dots, a_p))}_{q}$$

There are also others genetic operators for crossing, i.e., asexual and sexual crossover. Algorithm 1 gives the outline of an Evolutionary Algorithm.

This kind of algorithms are inspired by the Darwinian principles of nature's capability to evolve living beings well adapted to their environment [13]. Evolutionary algorithms are divided in the groups: genetic algorithms [60], evolution strategies [58], evolutionary programming [16, 17], and genetic programming [5].

Algorithm 1 Outline of an Evolutionary Algorithm

```
1: procedure EA
2:     Initialize $P(0) := \{\boldsymbol{a}_1(0), \ldots, \boldsymbol{a}_\mu(0)\} \in I^\mu$
3:     Evaluate $P(0) :\ \{\Phi(\boldsymbol{a}_1(0)), \ldots, \Phi(\boldsymbol{a}_\mu(0))\}$
4:     $t = 1$
5:     while $l(P(t)) \neq$ true do
6:         Recombine: $P'(t) = r_{\Theta_r}(P(t))$
7:         Mutate: $P''(t) = m_{\Theta_m}(P'(t))$
8:         Evaluate: $P''(t) :\ \{\Phi(\boldsymbol{a}''_1(t)), \ldots, \Phi(\boldsymbol{a}''_\lambda(t))\}$
9:         Select: $P(t+1) = s_{\Theta_s}(P''(t) \cup Q)$
10:        $t = t + 1$
11:    end while
12: end procedure
```

1.2.2 Physics-Inspired Algorithms

As mentioned, Physics-inspired algorithms can be used to solve complex optimization problems. This kind of algorithms inherit properties from its respective phenomena inspiration. Besides, the parameters considered can affect the convergence and exploration-exploitation balance [11].

The main areas covered by these algorithms are quantum theory, the laws of motion, electrostatics, electromagnetism, Newton's gravitational law.

Biswas et al. [11] have categorized those physics-inspired metaheuristics by their metaphor. This classification is presented in next sections to provide a point of view and how we can start with our proposal. Also, a brief description of the most important algorithms is given.

Newton's gravitational law

Some algorithms are based on the theory of gravitational field and particle kinematics. That theory implies that heavier particles will have more force of attraction as compared to smaller ones. Using this principle, the solution of an optimization problem is represented by the particle with the most force of attraction.

Some algorithms reported are:

- Central Force Optimization (CFO) [35].
- Gravitational Search Algorithm (GSA) [73].
- Immune Gravitation Inspired Optimization Algorithm (IGOA) [95].
- Gravitational Interaction Optimization (GIO) [34].
- Artificial physics optimization (APO) [92]
- Vector model of artificial physics optimization (VM-APO) [91]
- Binary gravitational search algorithm (BGSA) [72]
- Multiobjective gravitational search algorithm (MOGSA) [38]
- PSO gravitational search algorithm (PSOGSA) [42]
- Improved gravitational search algorithm (IGSA) [54]
- Chaotic Gravitational Search Algorithm (CGSA) [59]
- Water Cycle Algorithm (WCA) [14]
- Extended central force optimization (ECFO) [26]

CFO

This algorithms uses an analogous Newton's gravitation law defined as follows:

$$F = G\frac{m_1 m_2}{r^p}$$

where m_1, m_2 are the masses of particles, F and r are the force and the distance between them, respectively. G represents the gravitational constant.

CFO updates its positions $\boldsymbol{x}_i$ and velocity $\boldsymbol{v}_i$ of particles using the following formulation:

$$\begin{aligned} \boldsymbol{v}_i(t+1) &= w\boldsymbol{v}_i(t) + \frac{\lambda \boldsymbol{F}_i}{m_i} \\ \boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i + \boldsymbol{v}_i(t+1) \end{aligned}$$

where $w \in (0,\ 1)$ is a user-defined weight, $\lambda \in [0,\ 1]$ uniformly at random and F the total force which contains the bias for giving direction to find the best solutions.

GSA

It is inspired by law of motion and Newton's law of universal gravitation. Also, the gravitational constant G depends of the time and it provides a parameter to handle convergence to the algorithm. Authors of GSA expressed G is defined as follows:

$$G(t) = G(t_0) \times \left(\frac{t_0}{t}\right)^{\beta},\ \beta < 1.$$

In contrast with CFO, GSA computes the mass $M_i(t)$ of any agent $\boldsymbol{x}_i \in \mathbb{R}^D$ in terms of objective function at time t,

$$M_i(t) = \frac{m_i(t)}{\sum_j m_j(t)} \ \text{ with } \ m_i(t) = \frac{f(\boldsymbol{x}_i(t)) - f(\boldsymbol{x}_{\text{worst}}(t))}{f(\boldsymbol{x}_{\text{best}}(t)) - f(\boldsymbol{x}_{\text{worst}}(t))},$$

where f is the objective function, $\boldsymbol{x}_{\text{worst}}$ is the agent with the worst objective function value at time t; $f(\boldsymbol{x}_{\text{best}})$ the best objective function value found at time t.

The acceleration $\boldsymbol{a}_i(t)$ of any agent i at time t is given by:

$$\boldsymbol{a}_i(t) = \frac{F_i}{M_i},$$

where F_i is the total force which contains the bias obtained from the objective function and the distance between $\boldsymbol{x}_i$ and the remaining agents.

The next position is computed by the following expressions:

$$\begin{aligned}\boldsymbol{v}_i(t+1) &= r\boldsymbol{v}_i(t) + \boldsymbol{a}_i(t),\\ \boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1).\end{aligned}$$

where $r \in [0, 1]$ is a random distributed variable and $\boldsymbol{v}_i$ is the "velocity" which provides a new direction to move the agent into promising regions.

With the above formulation, agents can converge towards high-quality agents with cumulative attraction. GSA can converge slowly and easily fall into local optimum solution. IGOA algorithm improves these issues of GSA using vaccination and memory antibody replacement.

GIO

GIO and the algorithm called charged system search (CSS) are similar to GSA. GIO is inspired from Newton's gravitation law while CSS is based on electrostatic dynamics law which is analogous to Newton's gravitation law. The reader is referred to [11].

Quantum mechanics

Here, the main inspiration is given by quantum mechanics. Some reported algorithms are:

- Quantum-inspired bacterial swarming optimization (QBSO) [19]
- Quantum-inspired genetic algorithm (QGA) [65]
- Quantum-inspired evolutionary algorithm (QEA) [37]

- Quantum-inspired immune clonal algorithm (QICA) [52]
- Quantum genetic optimization (QGO) [56]
- Quantum-behaved particle swarm optimization (QPSO) [81]
- Versatile quantum-inspired evolutionary algorithm (vQEA) [69]
- Binary Quantum-inspired evolutionary algorithm (BQEA) [37]
- Continuous quantum ant colony optimization (CQACO) [51]
- Reduced quantum genetic algorithm (RQGA) [76]
- Quantum Swarm Evolutionary Algorithm (QSE) [87]
- Improved quantum evolutionary algorithm (IQEA) [94]

Quantum-inspired Genetic Algorithm

QGA [65] uses the idea of parallel universe in a genetic algorithm [60] to simulate quantum computing. Depending on this parallel universe interpretation, there is a population contained in each universe. All populations are governed by the same rules, but a population of a universe can be affected by other universe.

Quantum-inspired Evolutionary Algorithm

This algorithm is based on quantum bit and “superposition of states”. QEA [65] was originally inspired by quantum computing and takes important concepts of quantum mechanics. It has been widely applied [11].

Quantum Swarm Evolutionary Algorithm

QSE [87] takes the concepts from both PSO [48] and QEA [65]. Similar to PSO algorithm, in QSE the swarms are described using Q-bits. The main difference with QEA is that QSE uses a representation of Q-bit and changes the probabilistic parameters. α and β are replaced with angular parameters $\sin\theta$ and $\cos\theta$, where θ is the quantum angle.

Here, $\boldsymbol{x}$ and $\boldsymbol{c}$ are current location vector and current best or the center, respectively. L is called imagination parameter or creativity of particle. A new location vector is generated as follows:

$$x(t) = c \pm \frac{L}{2} \log\left(\frac{1}{R}\right),$$

where, $R \in [0, 1]$ is a random distributed variable. The parameter L is computed using the following formula:

$$L = 2\alpha|c - x(t)|.$$

Here, α is the creative coefficient and acts as main ingredient for convergence towards the optima.

Universe theory

Algorithms reported are:

- Big Bang-big Crunch algorithm (BB-BC) [28]
- Galaxy-based search algorithm (GbSA) [93]
- Unified big bang-chaotic big crunch algorithm (UBB-CBC) [28]

Big Bang-big Crunch

Big bang-big crunch (BB-BC) algorithm is based on the expansion phenomenon of Big Bang and shrinking phenomenon of Big Crunch [28].

BB-BC algorithm implements two stages: The first one is called “Big Bang phase” and “Big Crunch phase” is the second one. During the first stage, a new set of solutions is generated my using the center of mass. During Big Crunch phase, the center of mass simulates a gravitational attractor like a black hole. Here, the **population is shrunk to the unique center of mass** generated using the current population distribution.

BB-BC algorithm does not fulfill the balance exploration-exploitation, i.e this algorithm can move the new solutions into a local optimum. For example, when a candidate with the best objective function value converges to an optima at the very beginning of the algorithm, then all remaining solutions follow that the best solution and get trapped into local optima. This happens because BB-BC algorithm initializes a not uniformly distributed population.

When Chaotic Big Crunch phase is performed, the next position of each solution is updated using the following formula:

$$\boldsymbol{x}_i = \boldsymbol{x}_c \pm \frac{\alpha(t)(\boldsymbol{x}_{\text{best}} - \boldsymbol{x}_{\text{worst}})}{t}$$

where $\alpha_{t+1} = cf(\alpha(t))$, $0 < \alpha(t) < 1$, here $cf(x)$ is a chaotic function which provides a stepsize to shrink or stretch the population from center of mass $\boldsymbol{x}_c$ at time t.

Electromagnetism

Electromagnetism-like Heuristic

EM algorithm is inspired by the superposition principle of electromagnetism [10], which states that "*the force exerted on a point via other points is inversely proportional to the distance between the points and directly proportional to the product of their charges*". Here, points in search space are considered as particles and are governed by that principle.

EM algorithm supplies a good balance between exploration and exploitation since its mechanism are useful. Exploration and convergence of EM are controlled by a stochastic parameter. Exploitation is controlled by a vector whose components denote the allowed feasible values i.e. limits the movements of particles into the search space delimited by the upper and lower bounds.

Glass demagnetization

Hysteretic Optimization

HO algorithm is based on the demagnetization process of a magnetic piece [67]. A magnetic piece is transformed to a very stable state (low-energy) called "ground state", when it is demagnetized by an oscillating magnetic field with some properties. After demagnetization, the system is shakedup many times to obtain better results. HO is inspired on these two phases of magnetic piece for getting a stable state by repeating demagnetization followed by a number of shakeups.

The exploration is performed by the demagnetization process. The exploitation process is performed when a number of shakeup operations are made.

Electrostatics

Charged System Search

CSS algorithm is inspired from Coulomb's law, Gauss's law and superposition principle from electrostatics [47]. Here, a solution is a charged particle.

The CSS algorithm has good exploring and exploiting capabilities of the search space. Exploitation of charged particle is guarantee by the resulting electric force of any particle. The repulsiveness and attractiveness of resulting force of any charged particle with the new concept of a parameter is very effective for exploitation. However, whether charged particle is going to explore or exploit the search space depends on two parameters which handle the exploitation-exploration balance. CSS algorithm can ensure convergence towards better solutions with its exploration process.

CSS provides a high exploration at the beginning of the optimization process, thus the algorithm does not suffer from premature convergence. However, since good solution attracts others, if the initial set of charged particles are not uniformly distributed over search space, then the algorithm can converge into a local optima.

Other Inspirations

- Ray Optimization (RO) [46]
- Optics inspired optimization (OIO) [45]
- Ions Motion Optimization (IMO) [41]
- Galactic Swarm Optimization (GSO) [63]
- Electromagnetic field optimization (EFO) [1]

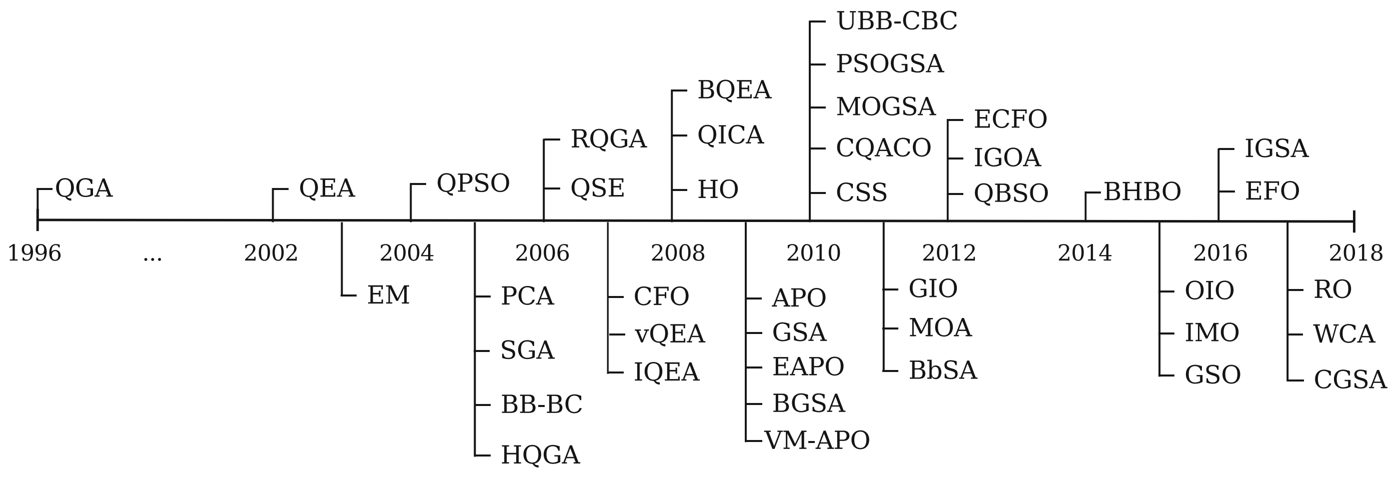


Figure 1.4: Physics-Inspired algorithms through time.

In summary, this kind of algorithms have been successful used for solving real-world problems. Although, they are not as popular as the algorithms inspired from Darwin's natural evolution; physics-inspired algorithms can show good performance. Figure 1.4 shows the evolution of physics-inspired algorithms. In the following section Evolutionary Centers Algorithm is presented within the general framework introduced so far.

Chapter 2

Evolutionary Centers Algorithm

2.1 Introduction

EAs have provided successful results when solving complex bound-constrained optimization problems [80]. However, most popular EAs usually are those which design keeps simple and their number of parameters is low so as to facilitate the fine-tuning process when a particular problem is solved.

Motivated by the above mentioned, we propose a physics-inspired algorithm based on the center of mass concept on a D-dimensional space for real-parameter single-objective optimization. The general idea is to promote the creation of an irregular body using K mass points in the current population, then the center of mass is calculated to get a new direction for the next population.

There are different algorithms based on biological or physical metaphors with different characteristics. Some of them use the current population distribution to generate new solutions, i.e., swarm intelligence algorithms such as particle swarm optimization (PSO) [48], and the artificial bee colony (ABC) [44]. There are also algorithms inspired by physical phenomena such as Newton's Law of Universal Gravitation (CFO) [11, 35]. The relationship among those algorithms is their mathematical formulation for generating solutions through an iterative process:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{v}_{i+1} \tag{2.1}$$

where each algorithm updates $\boldsymbol{v}_{i+1}$ as follows:

- PSO:

$$\boldsymbol{v}_{i+1} = \omega \boldsymbol{v}_i + c_1 r_{1,i}(\boldsymbol{x}_{pbest,i} - \boldsymbol{x}_i) + c_2 r_{2,i}(\boldsymbol{x}_{gbest,i} - \boldsymbol{x}_i),$$

  where ω is a inertia weight used for balancing the global search and local search, $\boldsymbol{x}_{pbest,i}$ and $\boldsymbol{x}_{gbest,i}$ are the best position reached by solution i so far and the best solution in the population, respectively; c_1 and c_2 are two positive constants, $r_{1,i},\ r_{2,i}$ are random numbers with uniform distribution in the range [0, 1].

- ABC:

$$\boldsymbol{v}_{i+1} = \phi_i(\boldsymbol{x}_i - \boldsymbol{x}_r),$$

  where $\boldsymbol{x}_i$ is the current solution, $\boldsymbol{x}_r$ is a randomly chosen solution, ϕ_i is a randomly produced number with uniform distribution in the interval $[-1,\ 1]$.

- CFO:

$$\boldsymbol{v}_{i+1} = \omega \boldsymbol{v}_i + \lambda \boldsymbol{F}_i / m_i,$$

  where $\boldsymbol{v}_i$ is the current solution, λ is a uniformly distributed random number in [0, 1], $\omega \in (0,\ 1)$ is user-defined weight, $m_i,\ F_i$ are mass and force functions, respectively, both defined by the authors.

In the three previous cases, the $\boldsymbol{v}$ value depends of the population distribution at current generation i.

The following Section 2.2 describes a preliminary version of our algorithm and how it relates to what has been described above. Also, presents results of the preliminary version.

2.2 Evolutionary Centers Algorithm: Preliminary Version

In this section, a preliminary version of Evolutionary Centers Algorithm (ECAp) is detailed. Also, experiments are presented.

2.2.1 Motivation

The center of mass is a geometric property of any object. Intuitively, it is the weighted location of an object. We can fully describe the movement of any object through space in terms of the rotation of the object about its center of mass if it is free to rotate and the translation of the center of mass of the object from one place to another. This is the motivation for using the center of mass concept, we translate the population to places where the mass of the entire population is maximum.

We present ECAp details. First, we introduce the center of mass in physics terms [50, 75].

Definition 2.2.1. The center of mass is the unique point $\boldsymbol{c}$ at the center of a distribution of mass $U = \{\boldsymbol{u}_1,\ \boldsymbol{u}_2, \ldots, \boldsymbol{u}_K\}$ in a space that has the property that the weighted sum of position vectors relative to this point is zero. That is:

$$\sum_{i=1}^{K} m(\boldsymbol{u}_i)(\boldsymbol{u}_i - \boldsymbol{c}) = 0, \quad \text{implies} \quad \boldsymbol{c} = \frac{1}{M}\sum_{i=1}^{K} m(\boldsymbol{u}_i)\boldsymbol{u}_i, \tag{2.2}$$

where $m(\boldsymbol{u}_i)$ is the mass of $\boldsymbol{u}_i$ and M is the sum of the masses of vectors in U. Here, m is a non-negative function.

Note 2.2.1. Similar as in Statistics, the center of mass is the mean location of a distribution of mass in space.

The concept of *center of mass* is, by far, not new. It was introduced by the ancient Greek physicist, mathematician, and engineer Archimedes of Syracuse. Archimedes worked with some assumptions about gravity in a uniform field, so as to get the mathematical properties of what we now call the center of mass [50].

For this work, the following proposition is required for ensuring stability and keep ECAp solutions into the convex space.

Proposition 2.2.1. *If $\boldsymbol{c}$ is the center of mass of a system of particles U, then for all $\boldsymbol{u} \in U$:*

$$d(\boldsymbol{c},\ \boldsymbol{u}) \leq \mathit{diam}(U).$$

Here, $\mathit{diam}(U) := \sup\{d(\boldsymbol{u},\ \boldsymbol{v}) \mid \boldsymbol{u},\ \boldsymbol{v} \in U\}$.

In others words, the center of mass of U is never out of the minimum convex set that contains U. We are assuming Euclidean distance and $U \subset \mathbb{R}^D$ [86].

In this work, the objective function of the optimization problem represents the mass of each solution in the population, i.e., we set $f = m$. Without loss of generality, we assume that we want to maximize the non-negative function f.

2.2.2 Algorithm Description

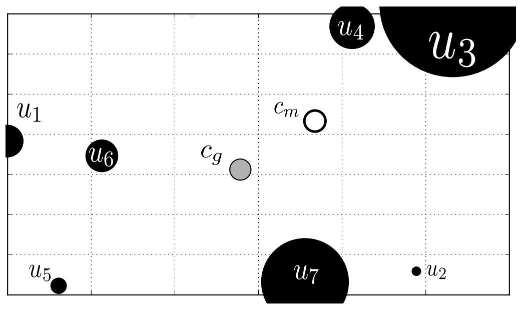


Figure 2.1: c_m is center of mass, c_g is geometric center of black points. Black point radius is its mass. Note the bias given by the weighted sum.

For each solution $\boldsymbol{x}_i$ in the population $P = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ of N solutions, we select a subset $U \subset P$ with K solutions; then, from U we obtain the center of mass $\boldsymbol{c}$. After that, based on a randomly chosen solution $\boldsymbol{u}_r \in U$, and the already generated center of mass $\boldsymbol{c}$, we generate a direction to locate a new solution $\boldsymbol{h}_i$. We suggest using the following strategy:

$$\boldsymbol{h}_i = \boldsymbol{x}_i + \eta_i(\boldsymbol{c}_i - \boldsymbol{u}_r), \tag{2.3}$$

where

$$\boldsymbol{c}_i = \frac{1}{W}\sum_{\boldsymbol{u}\in U} f(\boldsymbol{u}) \cdot \boldsymbol{u}, \quad W = \sum_{\boldsymbol{u}\in U} f(\boldsymbol{u}). \tag{2.4}$$

Note 2.2.2. If f is constant, then the center of mass of U is the geometric center of U. That is, assume that $f(\boldsymbol{x}) = \alpha$ for every $\boldsymbol{x} \in \mathbb{R}^D$, with α a positive constant. The center of mass is:

$$\boldsymbol{c}_i = \frac{1}{K\alpha}\sum_{\boldsymbol{u}\in U} \alpha \cdot \boldsymbol{u} = \frac{1}{K}\sum_{\boldsymbol{u}\in U} \boldsymbol{u}. \tag{2.5}$$

Thus, for a constant mass function, we have the center of mass converging to the geometric center. In real world problems, functions can be flat in some regions, then this algorithm may find some difficulties when dealing with such issue.

Note 2.2.3. The bias is given by Equation (2.4) because for a solution with the highest mass, the position of the center of mass is nearest to its position, see Figure 2.1.

Algorithm 2 ECAp pseudocode (ECA: preliminary version)

```
 1: procedure ECAp($K = 7,\ \eta_{\max} = 2$)
 2:     $N \leftarrow 2K * D$
 3:     Generate and evaluate start population $P$ with $N$ elements
 4:     while the end criterion is not achieved do
 5:         $A = \emptyset$
 6:         for each $\boldsymbol{x}$ in $P$ do
 7:             Generate $U \subset P$ such that card$(U) = K$
 8:             Calculate $\boldsymbol{c}$ using $U$ with (2.4)
 9:             $\eta \leftarrow$ rand$(0,\ \eta_{\max})$
10:             $\boldsymbol{h} \leftarrow \boldsymbol{x} + \eta * (\boldsymbol{c} - \boldsymbol{u})$ where $\boldsymbol{u} \in U$ random
11:             if $f(\boldsymbol{x}) < f(\boldsymbol{h})$ then
12:                 Append $\boldsymbol{h}$ in $A$
13:             end if
14:         end for
15:         $P \leftarrow$ best elements in $P \cup A$
16:     end while
17:     Report best solution in $P$
18: end procedure
```

Note that ECAp has only two parameters: the number of neighbors K and the stepsize $\eta_{\max}$. For large K values, ECAp could converge faster, we suggest $K = 7$ and $\eta_{\max} = 2$, values obtained experimentally. Figure 2.2 shows a representation of ECAp solution update.

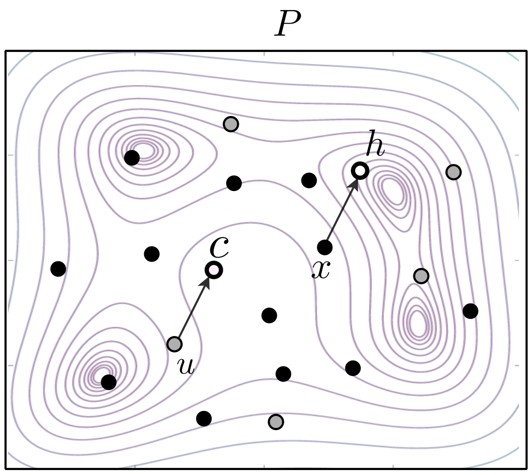


Figure 2.2: Schematic diagram representing a generation of ECAp. Gray points represent elements in U.

2.2.3 Experiments

Algorithm 2 details the procedure for the implementation of ECAp. Such algorithm was coded in C language using a PC with quad-core 2.4 GHz CPU and 8 GB of RAM and it was tested in thirty functions of the CEC 2017 competition on real-parameter single-objective optimization [3].

The algorithm developed in this this section was tested by solving the CEC17 benchmark. For this experimentation, $D = 10$ was considered. Here, the optimal values for the test functions are known (see Appendix A where test functions are shown). There is also a maximum number of evaluations equal to $10,000D$. The parameters in all experiments were: $K = 7$, η_i is a uniform random number between in (0, $\eta_{\max}$] with $\eta_{\max} = 2$. The size of the population was $N = 2K * D$. Those values were obtained experimentally such that provided good performance in a preliminary empirical study.

ECAp was compared with jSO [15] (an efficient solution for the CEC17 benchmark) and SQP [12, 66] (representative algorithm for nonlinear optimization).

Table 2.1: Representative functions from CEC 2017 benchmark (all the test functions are detailed in the Appendix A). This set of functions are shifted and rotated. The search range is $[-100,\ 100]^D$.

Function	Formula
Bent Cigar Function	$f_1(\boldsymbol{x}) = x_1 + 10^6 \sum_{i=2}^{D} x_i^2$
Sum of Different Power Function	$f_2(\boldsymbol{x}) = \sum_{i=1}^{D} \lvert x_i\rvert^{i+1}$
Zakharov Function	$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4$
Rosenbrock's Function	$f_5(\boldsymbol{x}) = 10D + \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i))$
Rastrigin Function	$f_5(\boldsymbol{x}) = 10D + \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i))$
High Conditioned Elliptic Function	$f_{11}(\boldsymbol{x}) = \sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}} x_i^2$
Discus Function	$f_{12}(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$
Griewank's Function	$f_{15}(\boldsymbol{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

2.2.4 Results

The statistical results obtained by ECAp are reported in Table 2.2. Wilcoxon rank-sum test ($\alpha = 0.05$) was computed [25]. In Table 2.3 "+" means that ECAp outperformed jSO/SQP in the function in the corresponding row, "-" means that jSO/SQP outperformed ECAp, and "$\approx$" means that no significant-difference was observed between algorithms. Note that SQP algorithm was outperformed by ECAp in 29 of 30 functions.

It is worth noting that ECAp obtained results close the optimum while reporting low standard deviation values. Therefore, ECAp behavior can be considered as robust and suitable to deal with different type of search spaces. Furthermore, we compared ECAp against a nonlinear optimization algorithm (SQP) [12, 66], and the most competitive algorithm in the CEC 2017 competition on real-parameter single-objective optimization (jSO), which is an adaptive algorithm based on differential evolution. From Table 2.3, we can see that, ECAp, based on the 95%-confidence Wilcoxon Rank-Sum test, was able to outperform jSO in five test functions, it reached similar results in seven test problems, and finally ECAp was outperformed by jSO in eighteen functions.

ECAp was then competitive in twelve test problems. Moreover, ECAp is more simple to implement than jSO and requires less mechanisms to operate. Note that ECAp outperformed SQP most of the time.

Figure 2.3 shows ECAp convergence graphs. Those plots show that ECAp is able to converge fast in most cases, which can be suitable for computationally-expensive real-world optimization problems.

2.2.5 Conclusions of ECAp

A new meta-heuristic optimization algorithm, denoted as Evolutionary Centers Algorithm (preliminary version), inspired by the center of mass of a system of particles was proposed. The results showed the capability of ECAp to consistently reach the vicinity of the global optima in different types of search spaces. ECAp also provided a competitive, but still not better, performance against the winner of the CEC 2017 competition on real-parameter single-objective optimization. ECAp is a simple algorithm which requires the fine-tuning of just two parameters, besides the population size.

Next section presents some improvements of ECAp. Also a release version is given.

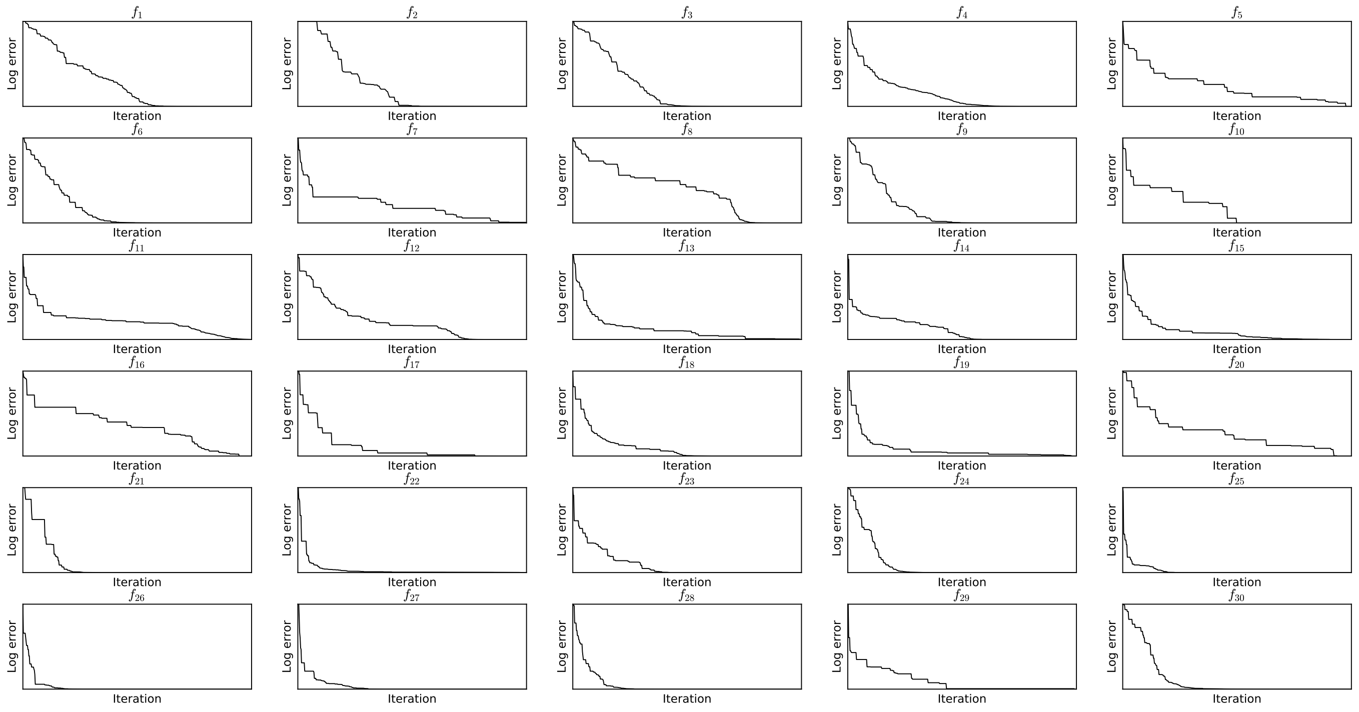


Figure 2.3: Convergence graphs at the run located in the median of 51 independentruns. Log scale is used for visualization purposes.

f	Best	Worst	Median	Mean	Std.
f_1	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
f_2	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
f_3	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
f_4	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
f_5	9.94967E–01	1.54772E+01	7.95967E+00	7.93134E+00	3.77745E+00
f_6	4.74662E–07	1.87951E–03	1.87537E–05	7.68970E–05	2.64095E–04
f_7	1.11988E+01	2.87323E+01	1.82470E+01	1.79819E+01	4.13487E+00
f_8	0.00000E+00	1.39919E+01	3.97988E+00	5.20411E+00	3.44675E+00
f_9	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
f_{10}	2.48759E+01	1.08470E+03	7.38475E+02	7.15419E+02	1.72816E+02
f_{11}	0.00000E+00	6.57982E+00	9.94986E–01	1.40699E+00	1.58245E+00
f_{12}	0.00000E+00	2.55756E+02	1.14822E+02	7.23057E+01	6.74642E+01
f_{13}	0.00000E+00	1.36689E+01	2.44396E+00	3.57464E+00	3.36532E+00
f_{14}	0.00000E+00	9.86504E+00	9.94959E–01	1.62895E+00	2.14527E+00
f_{15}	4.70850E–03	3.29460E+00	1.13397E+00	1.03424E+00	7.22193E–01
f_{16}	3.90059E–01	2.42227E+01	2.14109E+00	3.42342E+00	3.81635E+00
f_{17}	6.04248E+00	4.77547E+01	3.68447E+01	3.65033E+01	6.29621E+00
f_{18}	1.91438E–02	2.54001E+00	4.26493E–01	5.91709E–01	5.31956E–01
f_{19}	3.15884E–02	1.56140E+00	2.90525E–01	5.22549E–01	4.57284E–01
f_{20}	1.30976E+00	4.53821E+01	2.69242E+01	2.45399E+01	9.82172E+00
f_{21}	1.00000E+02	2.04138E+02	1.00000E+02	1.02042E+02	1.44386E+01
f_{22}	0.00000E+00	1.01678E+02	1.15631E+01	4.87450E+01	4.90438E+01
f_{23}	3.43302E–08	3.20754E+02	3.09754E+02	3.03630E+02	4.32143E+01
f_{24}	1.98982E–07	3.31138E+02	1.00000E+02	1.11616E+02	5.65212E+01
f_{25}	3.97743E+02	4.43546E+02	3.98009E+02	3.99730E+02	8.83947E+00
f_{26}	3.00000E+02	3.00000E+02	3.00000E+02	3.00000E+02	0.00000E+00
f_{27}	3.88861E+02	3.97791E+02	3.93436E+02	3.92839E+02	1.83721E+00
f_{28}	3.00000E+02	3.00000E+02	3.00000E+02	3.00000E+02	0.00000E+00
f_{29}	2.31919E+02	2.87909E+02	2.57749E+02	2.57882E+02	9.95923E+00
f_{30}	3.94649E+02	4.08051E+02	3.95237E+02	3.98513E+02	5.55498E+00

Table 2.2: ECAp results of 51 independent runs on CEC17 problems for $D = 10$.

f	ECAp		jSO	SQP	
f_1	0.00000E+00	$\approx$	0.00000E+00	3.220300E−04	+
f_2	0.00000E+00	$\approx$	0.00000E+00	6.065700E+20	+
f_3	0.00000E+00	$\approx$	0.00000E+00	0.000000E+00	$\approx$
f_4	0.00000E+00	$\approx$	0.00000E+00	6.253000E−01	+
f_5	7.93134E+00	–	1.67777E+00	2.685479E+02	+
f_6	7.68970E–05	–	0.00000E+00	9.300540E+01	+
f_7	1.79819E+01	–	1.20817E+01	5.279684E+02	+
f_8	5.20411E+00	–	1.91188E+00	1.641353E+02	+
f_9	0.00000E+00	$\approx$	0.00000E+00	4.765000E+03	+
f_{10}	7.15419E+02	–	3.83851E+01	1.582000E+03	+
f_{11}	1.40699E+00	–	0.00000E+00	8.694410E+01	+
f_{12}	7.23057E+01	–	3.55067E–01	5.889026E+02	+
f_{13}	3.57464E+00	$\approx$	2.68638E+00	2.617198E+02	+
f_{14}	1.62895E+00	–	1.36563E–01	1.054084E+02	+
f_{15}	1.03424E+00	–	3.00324E–01	9.494020E+01	+
f_{16}	3.42342E+00	–	5.49544E–01	6.348346E+02	+
f_{17}	3.65033E+01	–	5.25569E–01	8.089248E+02	+
f_{18}	5.91709E–01	–	2.17729E–01	1.050214E+02	+
f_{19}	5.22549E–01	–	7.72037E–03	8.497336E+02	+
f_{20}	2.45399E+01	–	3.36657E–01	5.496386E+02	+
f_{21}	1.02042E+02	+	1.42465E+02	3.480549E+02	+
f_{22}	4.87450E+01	+	1.00000E+02	1.470100E+03	+
f_{23}	3.03630E+02	–	3.01261E+02	6.704390E+02	+
f_{24}	1.11616E+02	+	2.96919E+02	4.196612E+02	+
f_{25}	3.99730E+02	+	4.12195E+02	4.360669E+02	+
f_{26}	3.00000E+02	$\approx$	3.00000E+02	1.591800E+03	+
f_{27}	3.92839E+02	–	3.89468E+02	4.350638E+02	+
f_{28}	3.00000E+02	+	3.40596E+02	4.037759E+02	+
f_{29}	2.57882E+02	–	2.34365E+02	2.005400E+03	+
f_{30}	3.98513E+02	–	3.94521E+02	3.957100E+08	+
Mean	116.9022		99.03207	2.021900E+19	

Table 2.3: Comparison of error results at mean between ECAp, jSO and SQP in $D = 10$ CEC 2017 test problems.

2.3 Evolutionary Centers Algorithm: Release Version

As mentioned in the previous Section, ECAp is based on the center of mass definition, which is adopted for creating new directions and generate a bias in the population, and such bias is based on the objective function values of the solutions in the population.

For ECAp, the objective function value of a given solution represents its mass, then each solution in the population has a mass value associated, i.e., we set $f(\boldsymbol{x}) = m(\boldsymbol{x})$ for all $\mathbb{R}^D$. Without loss of generality, it is assumed the maximization of a non-negative function f. This approach (ECA) uses this name since the population changes over time and therefore each center of mass will evolve over time. As described, ECAp is improved. Thus, next sections detail some improvements for ECAp.

2.3.1 Variation Operator

For each solution $\boldsymbol{x}_i$ in the population $P = \{\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \ldots,\ \boldsymbol{x}_N\}$ of N solutions, we generate $U_i \subset P$ with K **different** solutions such that

$$\bigcup_{i=1}^{N} U_i = P.$$

Next, from U_i the center of mass $\boldsymbol{c}_i$ is computed by using the Equation 2.2. After that, **the worst element** $\boldsymbol{u}_{\text{worst}}$ in U_i is selected according to the following rule:

$$\boldsymbol{u}_{\text{worst}} \in \arg\min\{f(\boldsymbol{u})\ :\ \boldsymbol{u} \in U_i\}.$$

Now, we are able to generate a direction to locate a new solution $\boldsymbol{y}_i$ using the already generated center of mass $\boldsymbol{c}_i$, $\boldsymbol{x}_i$ and $\boldsymbol{u}_{\text{worst}}$.

$$\boldsymbol{y}_i = \boldsymbol{x}_i + \eta_i(\boldsymbol{c}_i - \boldsymbol{u}_{\text{worst}}).$$

It is necessary to improve the exploration process to avoid premature convergence. A binomial crossover is considered (see Section 2.3.2). As mentioned in Section 2.2, ECA provides a fast convergence. Thus we can take advantage of it to exploit the vicinity near the best solution found using the last objective function evaluations. That is, generating a new solution with Equation 2.6:

$$\boldsymbol{y}_i = \begin{cases} \boldsymbol{x}_i + \eta_i(\boldsymbol{c}_i - \boldsymbol{u}_{\text{worst}}) & \text{if } P_{\text{evals}} < P_{\text{exploit}} \\ \boldsymbol{x}_i + \eta_i(\boldsymbol{x}_{\text{best}} - \boldsymbol{c}_i) & \text{otherwise} \end{cases} \tag{2.6}$$

where, P_{evals} is the evaluation ratio (current number of function evaluation / max. number of function evaluations), P_{exploit} is the last percentage of evaluations used for an exploitation process and $\boldsymbol{x}_{\text{best}}$ is the best element ever found. Finally, if $\boldsymbol{h}_i$ is better than $\boldsymbol{x}_i$, then the worst element in P is replaced by $\boldsymbol{h}$.

2.3.2 Binomial Crossover

Let us start the discussion by defining the binomial crossover.

Definition 2.3.1. (**Binomial Crossover**) The binomial crossover is defined for $\boldsymbol{x},\ \boldsymbol{y} \in \mathbb{R}^D$ as follows:

$$\boldsymbol{h} = \begin{cases} x_j & r < P_{\text{bin}} \\ y_j & \text{otherwise} \end{cases} \tag{2.7}$$

where $P_{\text{bin}} \in [0,\ 1]$ and r is a real random number uniform in the range [0, 1] and $j = 1, \ldots, D$.

To improve ECAp using this concept, suppose $\boldsymbol{y}_i$ is generated using Equation 2.6, the new solution can be $\boldsymbol{h}_i$ with coordinates:

$$h_{i,j} = \begin{cases} y_{i,j} & \text{if rand()} < P_{\text{bin}} \\ u_{\text{best},j} & \text{otherwise} \end{cases} \tag{2.8}$$

for $j = 1, 2, \ldots, D$.

Why binomial crossover? Suppose ECA is solving an optimization problem without binomial crossover for optimizing a function $f : \mathbb{R}^D \to \mathbb{R}$ and the entire population is contained in a plain $V \subset \mathbb{R}^D$ such that $\mathbf{0} \in V$. This plain V is a vector subspace of $\mathbb{R}^D$, i.e. any linear combination of vectors is in V when vectors are in V.

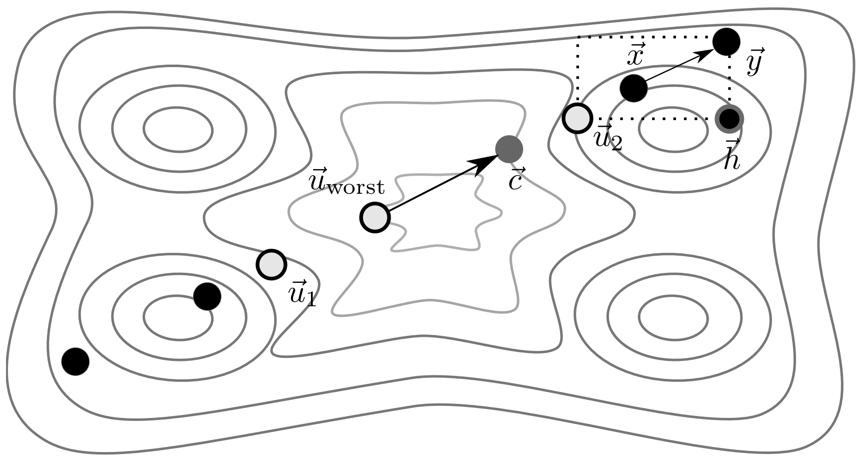


Figure 2.4: Binomial crossover representations.

Now, let $T : V^3 \to V$ be a linear transformation defined as

$$T(\boldsymbol{x},\ \boldsymbol{c},\ \boldsymbol{u}) = \boldsymbol{x} + \eta(\boldsymbol{c} - \boldsymbol{u}) \text{ where } \boldsymbol{x},\ \boldsymbol{c},\ \boldsymbol{u} \in V.$$

Assume, $\boldsymbol{x}_{\text{best}} = \boldsymbol{x}*$ such that $\boldsymbol{x}_{\text{best}}$ is not in V. Here, ECA can not reach $\boldsymbol{x}_{\text{best}}$ because ECA only generate solution in V (see Figure 2.4). Although, ECA without crossover can not generate solutions outside V, using binomial crossover with a low probability allows ECA to scape from a planar configuration.

The next section describes a self-adaptive crossover, since binomial crossover can be useful but it is necessary to establish a suitable value for P_{bin}.

2.3.3 Self-adaptive crossover

This section presents a self-adaptive strategy to handle the binomial crossover for ECA.

Definition 2.3.2. (**Generalized Binomial Crossover**) The generalized binomial crossover is defined for $\boldsymbol{x},\ \boldsymbol{y} \in \mathbb{R}^D$ as follows: for $j = 1, 2, \dots, D$, a new solutions $\boldsymbol{h}$ (generated via generalized binomial crossover) has coordinates:

$$h_j = \begin{cases} x_j & r < P_{\text{bin}}^j \\ y_j & \text{otherwise} \end{cases} \tag{2.9}$$

where each $P_{\text{bin}}^j \in [0,\ 1]$ and r is a real random number uniform distributed in the range [0, 1].

Note that, in generalized binomial crossover the vector variables can change with different probability. That can be profitable when some vector variables are independent each other.

The self-adaptation of parameter P_{bin}^j is based on its previous value. Let us describe this mechanism when ECA is performed.

1. Initialize population P.
2. Initialize $P_{\text{bin}}^j \in [0, 1]$ uniformly at random, for $j = 1, \dots, D$.
3. For each $\boldsymbol{x}_i \in P$, generate $\boldsymbol{y}_i$ using Equation 2.6.
4. Compute $\boldsymbol{h}_i$ using $\boldsymbol{x}_i,\ \boldsymbol{y}_i$ and Equation 2.9.
5. Calculate the frequency for all coordinate j where the generalized binomial crossover was applied ($r < P_{\text{bin}}^j$) and $\boldsymbol{h}_i$ is not better that $\boldsymbol{x}_i$.
6. Estimate the empirical probabilities P_e^j relative the frequency
7. For $j = 1, \dots, D$, update P_{bin}^j using the Equation 2.10.

$$P_{\text{bin}}^j = \begin{cases} P_{\text{bin}}^j & P_e^j < 0.3 \\ z & \text{otherwise} \end{cases}, \quad \text{with} \quad z \sim N(\mu,\ \sigma^2). \tag{2.10}$$

where $\mu = P_{\text{bin}}^{\text{best}}$, $\sigma = 0.3$ and $P_{\text{bin}}^{\text{best}} = \min_j P_e^j$. Algorithm 2 describes ECA.

Optionally, a linear reduction of the population (deleting the worst elements) can be applied. The initial population size is $N(0) = 2K * D$ and the final population size $N(T) = 2 * K$ (for successfully generating the center of mass). Thus, the population size over time is:

$$\begin{aligned} N(t) &= 2KD - \frac{(2KD - 2K)t}{T} \\ &= 2K\left(D - \frac{(D-1)t}{T}\right), \end{aligned} \tag{2.11}$$

where $t = 0, 1, 2, \dots, T$ and T is the maximum number of iterations.

Algorithm 3 is the general pseudocode of ECA.

Algorithm 3 ECA pseudocode

```
1:  procedure ECA($K = 7,\ \eta_{\max} = 2,\ P_{\text{exploit}} = 0.95$)
2:      $N \leftarrow 2K * D$
3:      Generate and evaluate initial population $P$ with $N$ elements
4:      Initialize $P_{\text{bin}}^1,\ \dots, P_{\text{bin}}^D$
5:      while the end criterion is not achieved do
6:          for each $\boldsymbol{x}$ in $P$ do
7:              Generate a subset $U \subset P$ such that card$(U) = K$
8:              Calculate $\boldsymbol{c}$ using $U$ with (2.4)
9:              $\eta \leftarrow$ rand$(0,\ \eta_{\max})$
10:             Calculate $\boldsymbol{h}$ using Eq. (2.8)
11:             if $f(\boldsymbol{x}) < f(\boldsymbol{h})$ then
12:                 Replace worst element in $P$ with $\boldsymbol{h}$
13:             end if
14:         end for
15:         Update all $P_{\text{bin}}^j$.
16:         Resize $P$ if necessary.
17:     end while
18:     Report best solution in $P$
19: end procedure
```

2.4 Empirical Study

Algorithm 3 was coded in Julia language using a PC with quad-core 2.4 GHz CPU and 8 GB of RAM. We tested ECA on the CEC 2017 benchmark [3]. The dimensions considered in this experimentations are $D = 10,\ 30$. The optimal values are known for all test functions. The maximum number of objective function evaluations is $10,000D$.

The parameters in ECA with the best performance in a preliminary empirical study were $K = 7$, $\eta_{\max} = 2$ and $P_{\text{exploit}} = 0.95$. The statistical results are detailed in Tables 2.6 and 2.7 for each dimension. In those tables, error values $|f(\boldsymbol{x}) - f^*|$ are presented. The error values between the best objective function values found in each run out of 31 independent runs. The best, median, mean, worst, and standard deviation of the error values are given in each respective column in the tables.

After that, we compare the performance of ECA against three recent representative algorithms: jSO (evolutionary algorithm), CMA-ES (evolution strategy) and CGSA (physics-inspired algorithm). Summarized results of the statistical test are in Tables 2.4 and 2.5 for $D = 10$ and $D = 30$, respectively. 95%-confidence Wilcoxon Rank-Sum test was used.

	jSO	CMA-ES	CGSA
"+"	8	25	27
"≈"	11	5	1
"−"	11	0	2

Table 2.4: Results of comparison for $D = 10$.

	jSO	CMA-ES	CGSA
"+"	17	29	28
"≈"	3	1	1
"−"	10	0	1

Table 2.5: Results of comparison for $D = 30$.

f_n	Best	Median	Mean	Worst	std
1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
3	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
4	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
5	1.9899e+00	3.9798e+00	4.5255e+00	9.9496e+00	2.1159e+00
6	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
7	1.0729e+01	1.3257e+01	1.4060e+01	2.0938e+01	2.6503e+00
8	0.0000e+00	2.9849e+00	3.7873e+00	1.0945e+01	2.2457e+00
9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
10	3.1227e-01	1.3368e+02	1.7893e+02	7.0640e+02	1.8006e+02
11	0.0000e+00	0.0000e+00	1.6048e-01	9.9496e-01	3.7199e-01
12	0.0000e+00	1.1381e+01	5.9787e+01	1.3486e+02	6.1880e+01
13	0.0000e+00	4.8371e+00	3.6322e+00	1.0705e+01	2.7258e+00
14	0.0000e+00	9.9496e-01	1.0529e+00	3.9798e+00	1.0521e+00
15	3.9249e-05	4.2573e-01	3.1989e+01	9.7743e+02	1.7547e+02
16	1.9432e-02	2.4585e-01	3.6355e-01	1.1034e+00	2.5660e-01
17	1.2174e-01	1.9518e+00	8.2085e+00	3.4509e+01	1.1037e+01
18	1.7212e-06	3.8349e-01	4.2395e-01	1.4341e+00	3.5770e-01
19	0.0000e+00	1.9729e-02	7.3706e-02	1.0341e+00	1.8309e-01
20	0.0000e+00	6.2435e-01	4.6841e+00	1.0797e+02	1.9402e+01
21	1.0000e+02	1.0000e+02	1.4780e+02	2.0984e+02	5.3562e+01
22	1.6350e+01	1.0000e+02	9.7378e+01	1.0064e+02	1.5039e+01
23	3.0000e+02	3.0585e+02	3.0549e+02	3.1300e+02	3.4501e+00
24	0.0000e+00	3.3334e+02	3.0839e+02	3.4176e+02	8.1875e+01
25	3.9774e+02	3.9958e+02	4.1873e+02	4.4585e+02	2.3100e+01
26	3.0000e+02	3.0000e+02	3.0000e+02	3.0000e+02	0.0000e+00
27	3.8692e+02	3.9382e+02	3.9314e+02	3.9815e+02	2.4019e+00
28	3.0000e+02	3.0000e+02	3.2927e+02	6.1182e+02	9.0998e+01
29	2.3127e+02	2.3790e+02	2.3973e+02	2.5269e+02	5.8143e+00
30	3.9450e+02	3.9475e+02	2.6762e+04	8.1758e+05	1.4677e+05

Table 2.6: Results of 31 independent runs of ECA on CEC17 problems for $D = 10$.

f_n	Best	Median	Mean	Worst	std
1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
2	0.0000e+00	0.0000e+00	7.4324e-06	2.2130e-04	3.9719e-05
3	5.9014e-04	2.8640e-01	4.6026e+00	3.4171e+01	8.4360e+00
4	1.1017e-03	4.0578e+00	3.0135e+01	6.7897e+01	3.0162e+01
5	8.9546e+00	2.2884e+01	2.3815e+01	4.2783e+01	6.9833e+00
6	0.0000e+00	0.0000e+00	3.2192e-06	5.8124e-05	1.0734e-05
7	3.8549e+01	4.8880e+01	4.8074e+01	6.5107e+01	5.6393e+00
8	8.9546e+00	1.9899e+01	2.0445e+01	3.5818e+01	6.0294e+00
9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
10	1.4760e+03	2.7036e+03	2.6634e+03	3.8467e+03	6.6384e+02
11	2.9849e+00	9.9496e+00	1.1469e+01	6.7939e+01	1.1195e+01
12	5.1373e+02	4.5552e+03	4.7011e+03	9.3850e+03	2.3913e+03
13	1.9619e+01	4.9515e+01	2.1511e+02	2.8108e+03	5.3228e+02
14	5.1385e+00	2.9391e+01	2.6437e+01	4.2577e+01	9.5370e+00
15	1.3379e+00	6.2463e+00	6.7772e+00	1.5993e+01	3.6803e+00
16	1.3627e+01	2.4183e+02	2.3558e+02	5.0178e+02	1.3612e+02
17	1.1459e+01	3.8681e+01	3.9928e+01	7.2637e+01	1.4051e+01
18	2.3018e+01	2.9299e+01	3.1869e+01	6.0968e+01	8.5271e+00
19	3.3277e+00	7.2680e+00	7.4862e+00	1.1759e+01	2.2081e+00
20	4.1907e+00	3.6551e+01	4.7707e+01	1.8685e+02	4.1628e+01
21	2.0793e+02	2.2214e+02	2.2103e+02	2.4113e+02	7.0141e+00
22	1.0000e+02	1.0000e+02	1.0000e+02	1.0000e+02	1.5498e-13
23	3.5356e+02	3.6843e+02	3.6905e+02	3.9151e+02	9.1451e+00
24	4.2665e+02	4.3813e+02	4.4120e+02	4.5736e+02	8.5273e+00
25	3.8670e+02	3.8690e+02	3.8690e+02	3.8714e+02	1.0768e-01
26	9.6620e+02	1.1359e+03	1.1387e+03	1.3212e+03	8.5575e+01
27	4.9484e+02	5.0613e+02	5.0600e+02	5.1677e+02	5.0949e+00
28	3.0000e+02	3.0000e+02	3.1666e+02	4.0329e+02	3.8616e+01
29	4.1191e+02	4.4819e+02	4.4993e+02	5.1759e+02	2.1281e+01
30	2.0491e+03	2.2882e+03	2.4074e+03	3.7279e+03	4.0395e+02

Table 2.7: Results of 31 independent runs of ECA on CEC17 problems for $D = 30$.

f_n	ECA	jSO		CMA-ES		CGSA	
1	0.0000e+00	0.0000e+00	$\approx$	2.4348e+08	$\approx$	1.3181e+07	$+$
2	0.0000e+00	0.0000e+00	$\approx$	1.0527e+02	$\approx$	1.9381e-05	$+$
3	0.0000e+00	0.0000e+00	$\approx$	0.0000e+00	$\approx$	1.1253e-06	$+$
4	0.0000e+00	2.7712e-01	$+$	0.0000e+00	$\approx$	5.7446e+01	$+$
5	4.5255e+00	8.8980e+00	$+$	2.3393e+01	$+$	2.6222e+01	$+$
6	0.0000e+00	0.0000e+00	$\approx$	6.1227e+00	$+$	5.1515e-01	$+$
7	1.4060e+01	2.0989e+01	$+$	2.1957e+01	$+$	1.6590e+01	$+$
8	3.7873e+00	8.9524e+00	$+$	1.1394e+01	$+$	1.5791e+01	$+$
9	0.0000e+00	0.0000e+00	$\approx$	6.8353e+00	$+$	1.2937e-07	$+$
10	1.7893e+02	6.6419e+02	$+$	1.1114e+03	$+$	1.3222e+03	$+$
11	1.6048e-01	9.8559e-05	$\approx$	5.5887e+01	$+$	3.5320e+01	$+$
12	5.9787e+01	2.4972e-01	$-$	2.0442e+06	$+$	7.4507e+03	$+$
13	3.6322e+00	4.6841e-01	$-$	3.5502e+02	$+$	7.4119e+03	$+$
14	1.0529e+00	6.4191e-02	$-$	1.2461e+02	$+$	3.6500e+03	$+$
15	3.1989e+01	1.7697e-01	$-$	1.1296e+02	$+$	2.5960e+03	$+$
16	3.6355e-01	7.2302e-01	$+$	2.3617e+02	$+$	4.5268e+02	$+$
17	8.2085e+00	1.1581e+01	$+$	6.9682e+01	$+$	1.6157e+02	$+$
18	4.2395e-01	2.6222e-01	$-$	2.1084e+05	$+$	1.9343e+05	$+$
19	7.3706e-02	1.0853e-02	$-$	9.5611e+02	$+$	3.7784e+03	$+$
20	4.6841e+00	2.9050e-01	$-$	1.2307e+02	$+$	2.2855e+02	$+$
21	1.4780e+02	1.1406e+02	$-$	1.7544e+02	$+$	2.2379e+02	$+$
22	9.7378e+01	1.0000e+02	$\approx$	1.0031e+02	$+$	1.1825e+02	$+$
23	3.0549e+02	3.0044e+02	$-$	3.7762e+02	$+$	4.0944e+02	$+$
24	3.0839e+02	1.6125e+02	$-$	3.1685e+02	$\approx$	2.3491e+02	$\approx$
25	4.1873e+02	3.9929e+02	$-$	4.3569e+02	$+$	4.3724e+02	$+$
26	3.0000e+02	3.0000e+02	$\approx$	3.4781e+02	$+$	2.3549e+02	$-$
27	3.9314e+02	3.9382e+02	$\approx$	4.2220e+02	$+$	4.9873e+02	$+$
28	3.2927e+02	3.0000e+02	$\approx$	5.4430e+02	$+$	6.3503e+02	$+$
29	2.3973e+02	2.4557e+02	$+$	3.6578e+02	$+$	4.0740e+02	$+$
30	2.6762e+04	3.9671e+02	$\approx$	7.1739e+05	$+$	6.4058e+03	$-$

Table 2.8: Comparison of results between ECA, jSO, CMA-ES and CGSA in $D = 10$ CEC 2017 test problems.

f_n	ECA	jSO		CMA-ES		CGSA	
1	0.0000e+00	0.0000e+00	$\approx$	4.7213e+09	$\approx$	2.4301e+03	$+$
2	7.4324e-06	0.0000e+00	$-$	1.7176e+39	$+$	2.3449e+42	$+$
3	4.6026e+00	0.0000e+00	$-$	3.7971e+03	$+$	1.5727e-04	$-$
4	3.0135e+01	5.8509e+01	$+$	3.8031e+03	$+$	4.0259e+02	$+$
5	2.3815e+01	1.1035e+02	$+$	2.2655e+02	$+$	1.5178e+02	$+$
6	3.2192e-06	0.0000e+00	$-$	5.5748e+01	$+$	1.3647e+01	$+$
7	4.8074e+01	1.5046e+02	$+$	9.0464e+01	$+$	5.5260e+01	$+$
8	2.0445e+01	1.1282e+02	$+$	1.1130e+02	$+$	1.0948e+02	$+$
9	0.0000e+00	0.0000e+00	$\approx$	1.8762e+03	$+$	3.4286e-06	$+$
10	2.6634e+03	5.9442e+03	$+$	4.9945e+03	$+$	3.2854e+03	$+$
11	1.1469e+01	1.7845e+01	$+$	2.6729e+03	$+$	1.6805e+02	$+$
12	4.7011e+03	4.6925e+02	$-$	2.6107e+09	$+$	1.7286e+07	$+$
13	2.1511e+02	4.6606e+01	$\approx$	7.9236e+08	$+$	1.4608e+04	$+$
14	2.6437e+01	4.2698e+01	$+$	1.8592e+05	$+$	2.7151e+03	$+$
15	6.7772e+00	1.3490e+01	$+$	4.6134e+06	$+$	1.4326e+03	$+$
16	2.3558e+02	7.7689e+02	$+$	2.1963e+03	$+$	1.7109e+03	$+$
17	3.9928e+01	1.4474e+02	$+$	8.6677e+02	$+$	9.9596e+02	$+$
18	3.1869e+01	2.4420e+01	$-$	3.9128e+06	$+$	3.8069e+04	$+$
19	7.4862e+00	1.5831e+01	$+$	9.6085e+06	$+$	3.4305e+03	$+$
20	4.7707e+01	1.4568e+02	$+$	5.6785e+02	$+$	9.4309e+02	$+$
21	2.2103e+02	3.0305e+02	$+$	5.3913e+02	$+$	3.2436e+02	$+$
22	1.0000e+02	1.0000e+02	$-$	6.5648e+02	$+$	8.2481e+02	$+$
23	3.6905e+02	4.3656e+02	$+$	8.9144e+02	$+$	8.2813e+02	$+$
24	4.4120e+02	4.9628e+02	$+$	9.4080e+02	$+$	6.3770e+02	$+$
25	3.8690e+02	3.8672e+02	$-$	7.5352e+02	$+$	3.8961e+02	$+$
26	1.1387e+03	1.4247e+03	$+$	5.9769e+03	$+$	1.0957e+03	$\approx$
27	5.0600e+02	4.8565e+02	$-$	1.0240e+03	$+$	8.1208e+02	$+$
28	3.1666e+02	3.0000e+02	$-$	1.3230e+03	$+$	3.2042e+02	$+$
29	4.4993e+02	6.2118e+02	$+$	1.5870e+03	$+$	1.2617e+03	$+$
30	2.4074e+03	2.0486e+03	$-$	1.2426e+08	$+$	7.1514e+03	$+$

Table 2.9: Comparison of results between ECA, jSO, CMA-ES and CGSA in $D = 30$ CEC 2017 test problems.

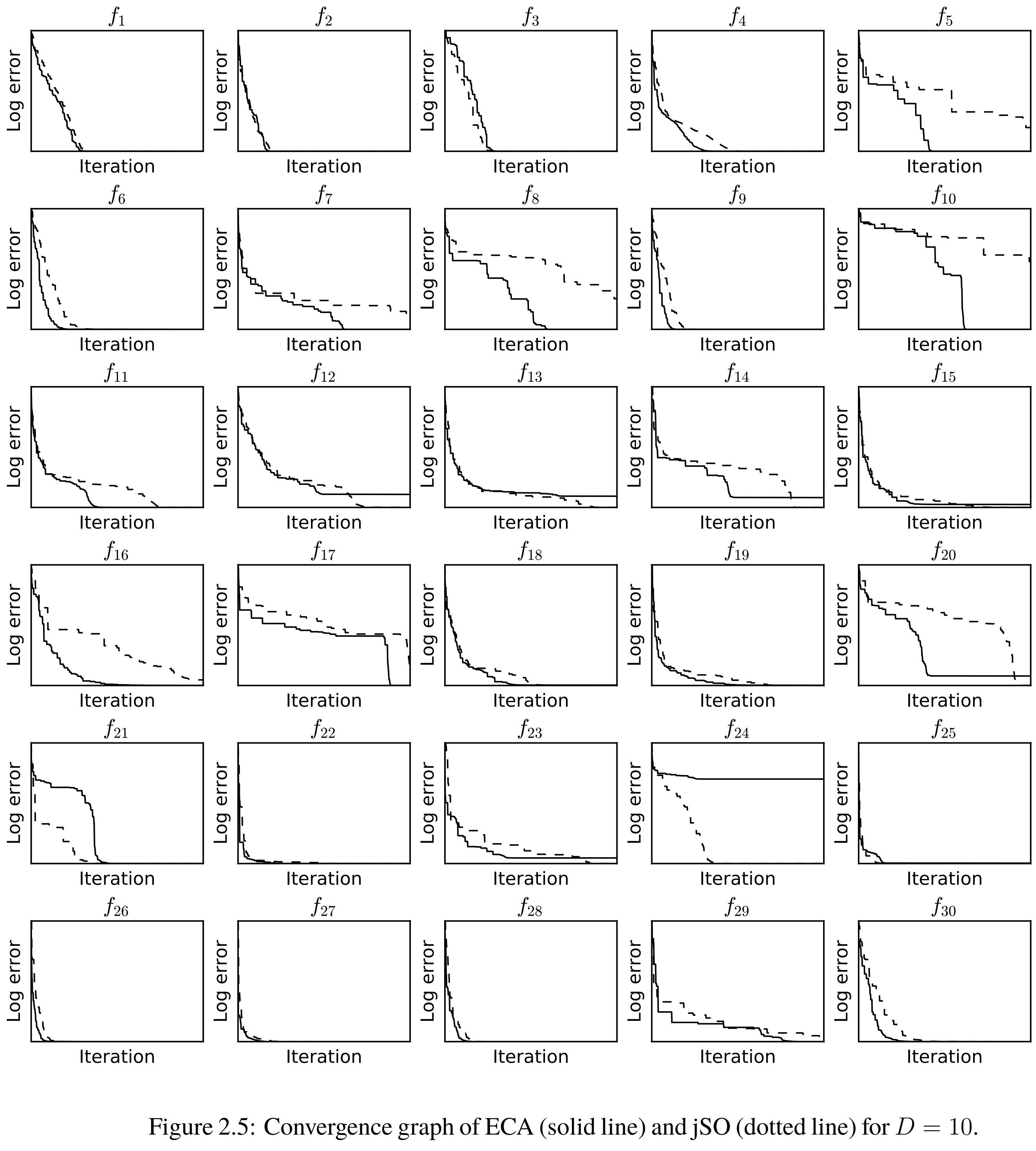


Figure 2.5: Convergence graph of ECA (solid line) and jSO (dotted line) for $D = 10$.

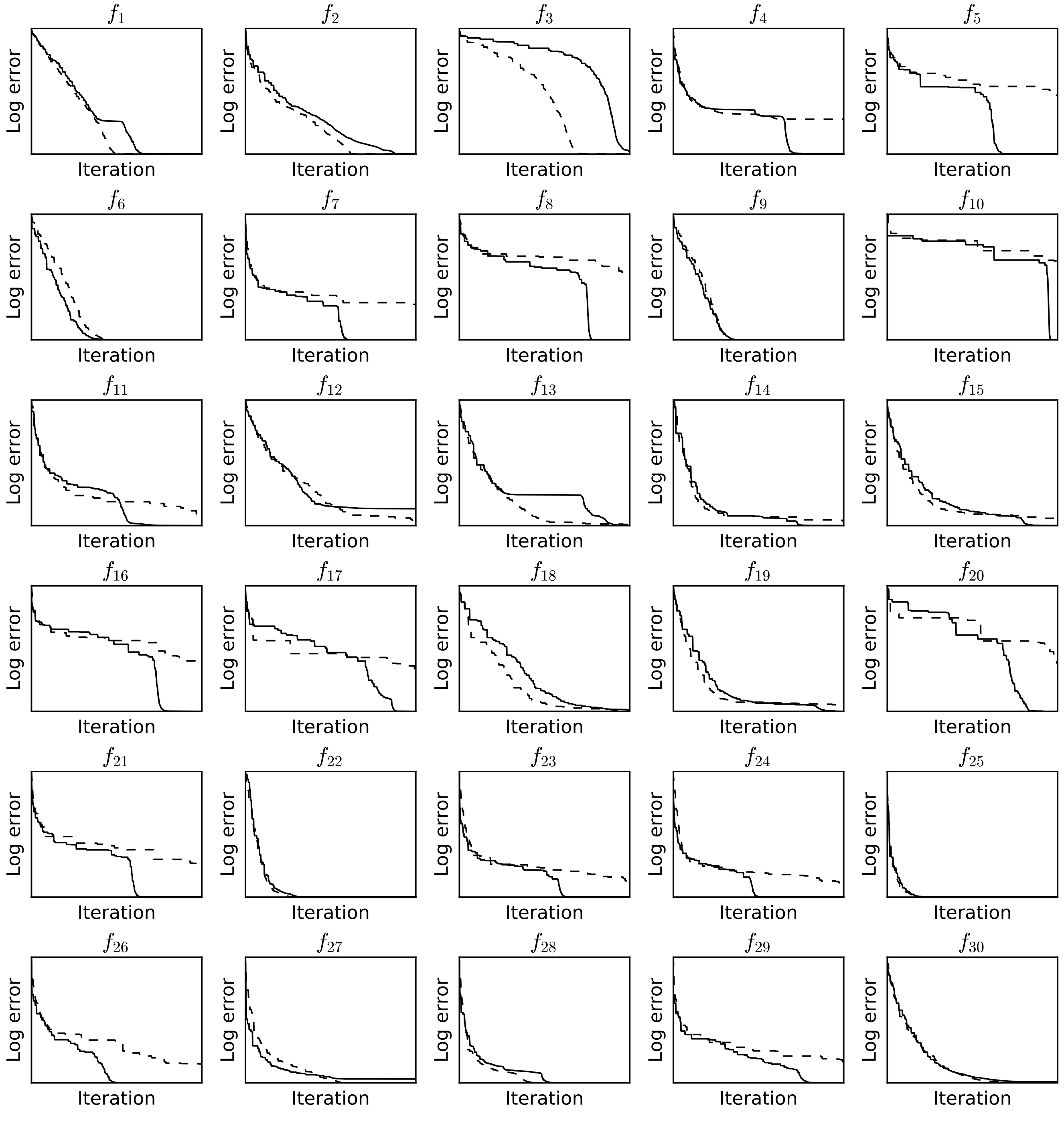


Figure 2.6: Convergence graph of ECA (solid line) and jSO (dotted line) $D = 30$.

2.4.1 Discussion

According to the results of Tables 2.8 and 2.9, ECA is able to provide very competitive results. For $D = 10$, our approach outperforms all others in f_4, f_5, f_7, f_8, f_{10}, f_{16}, f_{17} and f_{29}. For $D = 30$, ECA outperforms all others 17 test functions. It may be noted that the unimodal functions $f_1, \dots, f_3$ are suitable for benchmarking exploitation when $D = 10,\ 30$. Therefore, these results show good performance of ECA in terms of exploration and exploitation the optimum value. This is due to the proposed exploration and exploitation operators previously discussed.

Moreover, ECA is able to provide very competitive results most of the time, since our approach outperforms CMA-ES and GSA. Moreover, ECA shows very competitive results compared to jSO and outperforms it occasionally when $D = 10$ and jSO was outperformed most of time when $D = 30$. These results verify the performance of ECA in solving various benchmark functions compared to well-known metaheuristics.

Figures 2.5 and 2.6 show ECA and jSO convergence graphs. Those plots show that ECA is able to converge fast in most cases, which can be suitable for computationally-expensive real-world optimization problems.

2.4.2 Conclusion

This Chapter detailed a novel physics-inspired metaheuristic based on center of mass. Thirty test functions of a recent benchmark were employed in order to analyze the performance of the proposed algorithm in terms of precision, objective function evaluations and convergence.

Moreover, the results showed that ECA was able to provide highly competitive results compared to representative metaheuristics such as jSO, CMA-ES and CGSA. First, the results on the unimodal functions showed a good balance exploration-exploitation of the ECA algorithm, since the exploration ability of ECA was confirmed by the results.

Chapter 3

Optimal Synthesis of Mechanisms with ECA

Metaheuristics have been successfully used for solving real-word problems [33]. This kind of algorithms are highly competitive against Mathematical Programming when problems are highly non-linear and/or non-differentiable functions, large number of variables, among others [40]. Most recent and important Metaheuristics can be divided in three groups: Swarm Programming, Evolutionary Algorithms (EAs) and Physics-inspired Algorithms [11, 33, 43, 79].

In this Chapter, numerical constrained optimization problems (COP) are considered as defined in Definition 1.1.6.

Section 3.1 explains the general synthesis of a four-bar mechanism and three representative study cases on this topic. Section 3.2 presents the coupled dynamics of the four-bar mechanism with DC motor. Finally, the conclusions are given in Section 3.3.

3.1 Synthesis of Four-Bar Mechanisms

This section is based on the work of Hernández-Ocaña et al [39]. Four-bar mechanisms are formed by a references bar, input bar (crank), a coupler and a out bar (rocker). The simplicity of four-bar mechanisms (FBM) is convenient for a large

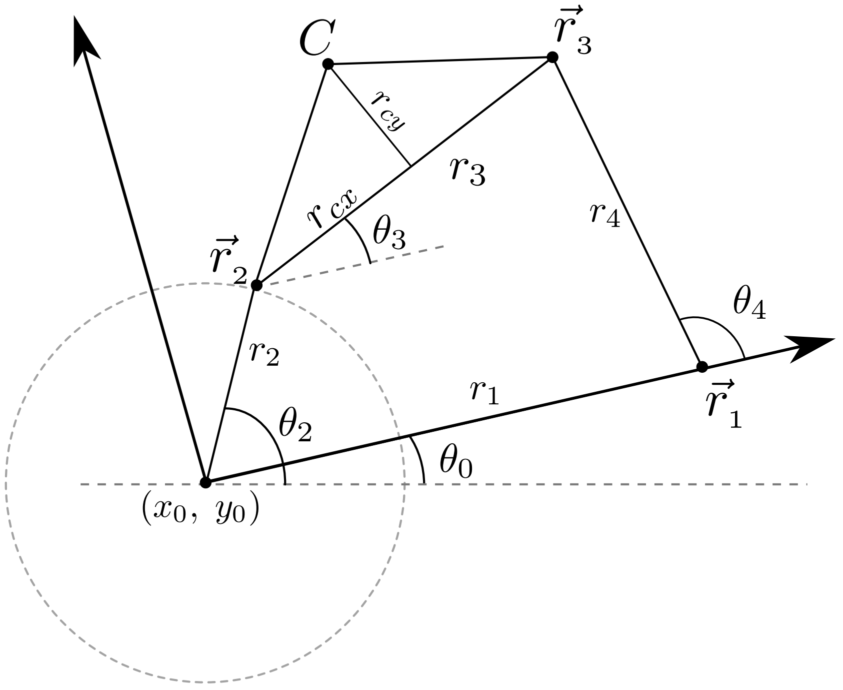


Figure 3.1: Four-Bar mechanism diagram.

number of industrial applications [64, 57, 84]. In fact, the kinematics of this kind of planar mechanisms has been extensively studied. A good idea it is transform from planar coordinates to polar coordinates, i.e.

$$\boldsymbol{r}_k = r_k(\cos(\theta_k) + i\sin(\theta_k)) \text{ for } k = 1, 2, 3, 4. \tag{3.1}$$

3.1.1 Kinematics of The Mechanism

To analyze the mechanism position, the closed loop equation can be established as follows:

$$\boldsymbol{r}_1 + \boldsymbol{r}_4 = \boldsymbol{r}_2 + \boldsymbol{r}_3. \tag{3.2}$$

From Equations (3.1) and (3.2) we obtain the following system:

$$\begin{cases} r_1 \cos\theta_1 + r_4 \cos\theta_4 = r_2 \cos\theta_2 + r_3 \cos\theta_3 \\ r_1 \sin\theta_1 + r_4 \sin\theta_4 = r_2 \sin\theta_2 + r_3 \sin\theta_3 \end{cases} \quad \text{implies} \tag{3.3}$$

Expressing the equation system (3.3) in terms of $\boldsymbol{r}_4$

$$\begin{cases} r_4 \cos\theta_4 = r_2 \cos\theta_2 + r_3 \cos\theta_3 - r_1 \cos\theta_1 \\ r_4 \sin\theta_4 = r_2 \sin\theta_2 + r_3 \sin\theta_3 - r_1 \sin\theta_1 \end{cases} \tag{3.4}$$

$$a \cos\theta_3 + b \sin\theta_3 + c = 0, \tag{3.5}$$

where

$$\begin{aligned} a &= 2r_3(r_2 \cos\theta_2 - r_1 \cos\theta_1), \\ b &= 2r_3(r_2 \sin\theta_2 - r_1 \sin\theta_1), \\ c &= r_1^2 + r_2^2 + r_3^2 - r_4^2 - 2r_1 r_2 \cos(\theta_1 - \theta_2). \end{aligned}$$

Then, the angle θ_3 can be calculated as a function with parameters a, b, c and θ_3, and $\cos\theta_3$ in terms of $\tan(\theta_3/2)$:

$$\sin\theta_3 = \frac{2\tan(\theta_3/2)}{1 + \tan^2(\theta_3/2)} \tag{3.6}$$

$$\sin\theta_3 = \frac{-\tan^2(\theta_3/2)}{1 + \tan^2(\theta_3/2)} \tag{3.7}$$

A second-order lineal equation is obtained by substitution on

$$[c - a]\tan^2(\theta_3/2) + 2b\tan(\theta_3/2) + a + c = 0 \tag{3.8}$$

From the solution of (3.8), the angular position θ_3 is given by:

$$\theta_3 = 2\arctan\left[\frac{-B \pm \sqrt{b^2 - 4ac}}{c - a}\right]. \tag{3.9}$$

Analogously, we can get θ_4.

3.1.2 Kinematics of the Coupler

Since the point of interest in the coupler is $C(C_{xr}, C_{yr})$, to determine its position in the reference system OX_rY_r it has to be established that

$$C_{xr} = r_2 \cos\theta_2 + r_{cx} \cos\theta_3 - r_{cy} \sin\theta_3, \tag{3.10}$$

$$C_{yr} = r_2 \sin\theta_2 + r_{cx} \sin\theta_3 - r_{cy} \cos\theta_3, \tag{3.11}$$

In the global coordinate system, this point is expressed as:

$$\begin{bmatrix} C_x \\ C_y \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & -\cos\theta_0 \end{bmatrix} \begin{bmatrix} C_{xr} \\ C_{yr} \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}. \tag{3.12}$$

Equations (3.9) and (3.12) and the expressions from the kinematics of the mechanism are sufficient to calculate the position of C along the trajectory.

3.1.3 Design Objective Function

In order to find the best mechanism such that the trajectory corresponds to n precision points, we quantify the error using quadratic error. Let $M = \{C_d^i \in \mathbb{R}^2 \mid i = 1, 2 \ldots, n\}$ be a set of precision points. Then, given the parameters of a mechanism, each point of the coupler can be expressed as a function of the input bar position

$$C^i = [C_x(\theta_2^i),\ C_y(\theta_2^i)]^T$$

Now, it is desired to minimize the error between each precision point C_d^i and the calculated point C^i, respectively. Thus, the error is calculated using the expression:

$$\text{error} = \sum_{i=1}^{n} \left[(C_{xd}^i - C_x^i)^2 + (C_{yd}^i - C_y^i)^2 \right].$$

3.1.4 Design Constraints

It is important to add some constraints to guarantee the movement criteria and restrict the size and shape of the mechanism. Here is where the Grashof's Law takes place:

Grashof's Law establishes that for a planar four-bar linkage, the sum of the shortest and the largest bars cannot be larger than the sum of the remaining bars, if a continual relative rotation between two elements is desired [64].

Hence, applying the Grashof's Law to our problem, we obtain

$$r_1 + r_2 \leq r_3 + r_4.$$

Now, we need a Crank-rocker mechanism. Thus, for ensure the solution method, the following constraints were established:

$$r_2 < r_3, \; r_3 < r_4, \; \text{ and } \; r_4 < r_1.$$

On the other hand, the angle sequence $\theta_2^1, \, \ldots, \, \theta_2^n,$ must satisfy $\theta_2^1 < \theta_2^2 < \cdots < \theta_2^n,$ because we are assuming successive precision points.

3.1.5 Experiments Design

Experiments details are detailed here. The vector of design variables is

$$\boldsymbol{p} = (r_1, \, r_2, \, r_3, \, r_4, \, r_{cx}, \, r_{cy}, \, x_0, \, y_0, \, \theta_0, \, \theta_2^1, \, \theta_2^2, \, \ldots, \, \theta_2^n) \in S \subset \mathbb{R}^{9+n}, \quad (3.13)$$

where the search space is $S = [0, \, 60]^4 \times [-60, \, 60]^4 \times [0, \, 2\pi]^{n+1}$, i.e. each boundary is:

$$\begin{aligned} r_1, \, r_2, \, r_3, \, r_4 &\in [0, \, 60] \\ r_{cx}, \, r_{cy}, \, x_0, \, y_0 &\in [-60, 60] \\ \theta_0, \, \theta_2^1, \, \ldots, \, \theta_2^n &\in [0, 2\pi]. \end{aligned}$$

Here, $r_1, \, \ldots, \, r_4$ correspond to the lengths of the bars. $(r_{cx}, \, r_{cy})$ is the position of the coupler. $O_2(x_0, \, y_0)$ is the coordinate position, θ_0 is the orientation angle of the system with respect to the horizontal. Finally, $\theta_2^1, \, \ldots, \, \theta_2^n$ are angle values of the input bar r_2.

Based on the mentioned above, constructing a four-bar crank-rocker mechanism is equivalent to solve the following constrained optimization problem:

$$\min f(\boldsymbol{p}) = \sum_{i=1}^{n} \left[(C_{xd}^{i} - C_{x}^{i})^2 + (C_{yd}^{i} - C_{y}^{i})^2 \right]. \tag{3.14}$$

subject to:

$$g_1(\boldsymbol{p}) = r_1 + r_2 - r_3 - r_4 \tag{3.15}$$

$$g_j(\boldsymbol{p}) = r_j - r_{j+1} \leq 0, \text{ for } j = 2, 3 \tag{3.16}$$

$$g_4(\boldsymbol{p}) = r_4 - r_1 \tag{3.17}$$

$$g_{4+j}(\boldsymbol{p}) = \theta_2^{j} - \theta_2^{j+1} \leq 0, \text{ for } j = 1, \ldots, n-1. \tag{3.18}$$

The next section presents three study cases and how they are solved using the Algorithm 3.

3.1.6 Study Cases

Here, three study cases are presented. Each case consists of n precision points which we need obtain a four-bar mechanism such that the coupler passes through them. Figure 3.2 shows the point distribution for each case.

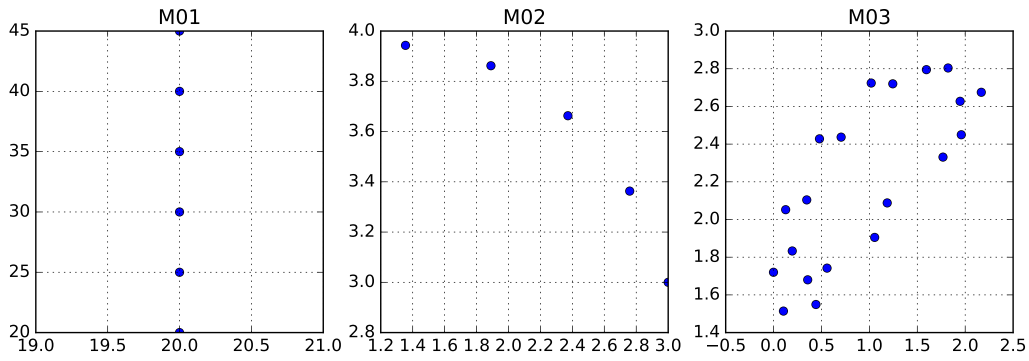


Figure 3.2: Distribution of the precision points. Three study cases are considered.

M01. It is the design of a four-bar mechanism that follows a linear vertical path defined by a sequence of six precision points, without a previously established synchronization. The set of precision points is defined as

$$M_1 = \{(20, 20),\ (20, 25),\ (20, 30),\ (20, 35),\ (20, 40),\ (20, 45)\}$$

The vector of design variables (see Eq. 3.13) becomes $\boldsymbol{p} \in \mathbb{R}^{15}$, since $|M| = 6$.

M02. The second study case is the design of a four-bar mechanism that follows a trajectory generated by precision points. Here, the values of angles are known. That is,

$$M_2 = \{(3, 3),\ (2.759,\ 3.363),\ (2.372,\ 3.663),\ (1.890,\ 3.862),\ (1.355, 3.943)\}$$

at respective angles of crank:

$$\theta_2^i = \frac{2\pi}{12}, \frac{3\pi}{12}, \ldots, \frac{6\pi}{12}.$$

Also, $x_0, y_0, \theta_0 = 0$. Hence, the design variables is $\boldsymbol{p} \in \mathbb{R}^6$. This case has only four constraints $g_1(\boldsymbol{p}), \ldots, g_4(\boldsymbol{p})$ in the COP (3.14)-(3.17).

M03. This case considers a sequence of ten pairs of precision points: The vector of design variables is $\boldsymbol{p} \in \mathbb{R}^{19}$. Here, the coupler must adjust a trajectory to each par simultaneously.

Pair	M_3^1	M_3^2
1	(1.768, 2.3311)	(1.9592, 2.44973)
2	(1.947, 2.6271)	(2.168, 2.675)
3	(1.595, 2.7951)	(1.821, 2.804)
4	(1.019, 2.7241)	(1.244, 2.720)
5	(0.479, 2.4281)	(0.705, 2.437)
6	(0.126, 2.0521)	(0.346, 2.104)
7	(-0.001, 1.720)	(0.195, 1.833)
8	(0.103, 1.514)	(0.356, 1.680)
9	(0.442, 1.549)	(0.558, 1.742)
10	(1.055, 1.905)	(1.186, 2.088)

Table 3.1: Pairs of precision points for case study 3.

In this case, It is necessary to modify the objective function, then:

$$f(\boldsymbol{p}) = \sum_{i=1}^{n} \left[(C_{xd}^{i} - C_{x}^{i}) + C_{yd}^{i} - C_{y}^{i}) \right] + \sum_{i=1}^{n} \left[(C_{2xd}^{i} - C_{x}^{i}) + C_{2yd}^{i} - C_{y}^{i}) \right], \tag{3.19}$$

with $(C_{xd}^{i},\ C_{yd}^{i}) \in M_3^1$ and $(C_{2xd}^{i},\ C_{2yd}^{i}) \in M_3^2$. Thus, the optimization problem for this cased is given by (3.19) and (3.15)–(3.18).

The next section presents the results and discussion about how ECA solve those study cases.

3.1.7 Results and Discussion

Algorithm 3 was used for solving the three four bar-mechanism synthesis design problems detailed above. ECA was coded in Julia 0.6 [8] and executed on a PC with a Core i5 and 8GB of RAM, and 64 bits Linux Mint 18 operating system. Since those problems are complex to solve, the set of parameters for ECA for each case (and run) are chosen randomly:

- $K \in \{3, 4, \ldots 7\}$
- $N = K * D$
- $\eta_{\text{max}} \in [2,\ 4]$
- $P_{\text{exploit}} = [0.90, 0.91]$

The constraint-handling approach used in work is based on three feasibility rules, also called Deb's rules [24]. Additional information is presented in Table 3.2.

The statistical results obtained from 31 independent runs for each problem and algorithm are reported in Table 3.3. Furthermore, we compared ECA against a variant of differential evolution (DE) [80] which is a competitive evolutionary algorithm designed for solving optimal four-bar synthesis [39] and some recent representative algorithms: an adaptive version of DE called jSO [15], an evolutionary strategy with auxiliary evolution path (CMA-ES) [53] and a physics-inspired algorithm based on

	Max. Evals.	Constraints
M01	300000	10
M02	60000	4
M03	350000	14

Table 3.2: Parameters of Algorithm 3. First column is the number of study case. Note the high number of constraints.

	Alg.	**Evals.**	**Best**	**Median**	**Mean**	**Std.**	**FR**
M01	ECA	300000	**0.000000E+00**	3.455233E–04	4.449034E–04	7.345703E–04	100
	DE	750000	7.573065E–29	1.328421E–05	2.601458E–03	7.588249E–03	100
	jSO	300000	1.009917E–05	4.035744E–04	6.499463E–04	6.215423E–04	100
	CMA-ES	300000	1.045442E–03	2.266048E+00	9.512426E+02	2.332255E+03	100
	CGSA	300000	1.407237E+03	2.879480E+03	4.652438E+03	3.420529E+03	16
M02	ECA	60000	**2.628079E–03**	2.628079E–03	2.628079E–03	4.408496E–19	100
	DE	100000	**2.628079E–03**	2.628079E–03	2.628079E–03	6.736261E–18	100
	jSO	60000	**2.628079E–03**	2.628079E–03	2.628079E–03	1.010175e-17	100
	CMA-ES	60000	**2.628079E–03**	2.628079E–03	2.180029E+00	1.212253E+01	100
	CGSA	60000	4.400252E+00	9.183622E+01	6.436224E+02	1.072564E+03	97
M03	ECA	380000	**2.749687E–01**	2.783940E–01	3.755563E–01	2.502182E–01	100
	DE	500000	**2.749687E–01**	2.773593E–01	7.610914E–01	1.116940E+00	100
	jSO	380000	2.749911E–01	3.094869E–01	3.802881E–01	2.314864E–01	100
	CMA-ES	380000	2.864445E+00	9.541319E+01	8.168649E+03	2.395725E+04	84
	CGSA	380000	–	–	–	–	0

Table 3.3: Each algorithm solved 31 times each problem, and their statistical results are presented here. A result in **boldface** indicates the best values found.

the law of gravity and mass interactions (CGSA) [59]. For those recent algorithms, Deb rules were used as constraint handling method.

The feasibility rate (FR) is defined as *(# of feasible runs) / Total runs*, where a feasible run is a run where at least one feasible solution was found. Here, "algorithm $\mathcal{A}_1$ outperforms algorithm $\mathcal{A}_2$" means $\text{FR}_{\mathcal{A}_1} > \text{FR}_{\mathcal{A}_2}$ or if $\text{FR}_{\mathcal{A}_1} = \text{FR}_{\mathcal{A}_2} = 100$, then a 95%-confidence Wilcoxon Rank-Sum test was computed.

	DE	jSO	CMA-ES	CGSA
M01	$\approx$	+	+	+
M02	$\approx$	$\approx$	$\approx$	+
M03	$\approx$	$\approx$	+	+

Table 3.4: A comparison between ECA, DE, JSO, CMA-ES and CGSA. "+" means that ECA outperformed DE, JSO, CMA-ES and CGSA in the problem in the corresponding row and "$\approx$" means that no significant-difference was observed between algorithms, all based on the 95% confidence Wilcoxon Rank-Sum test.

Comparing ECA and DE, ECA reached similar results in the three problems but with less evaluations of objective function. Also, ECA outperformed jSO, CMA-ES and CGSA most of the time. As we can see in Figure 3.4, ECA was able to converge fast in the three cases. Figure 3.3 shows the best mechanism found by this approach (numerical values in Table 3.5). Thus, ECA is capable to solve this kind of problems with less objective function evaluations.

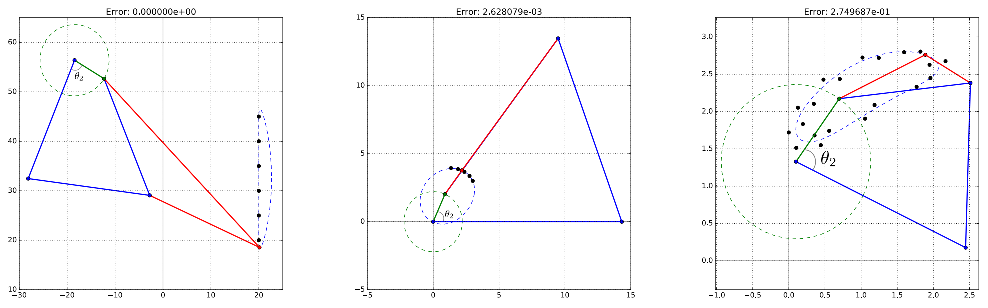


Figure 3.3: Mechanisms provided by ECA. From a mechanical point of view, these designs are good, since the trajectory of M01 and M02 pass over the precision points.

On the other hand, to solve real-word problems, it is necessary to find the optimal value of the objective function, here ECA obtained the best value known in all problems. Section 3.2 presents a low-dimensional problem but with a computational expensive objective function. Thus, a fast convergence with good results are required.

	r_1	r_2	r_3	r_4	r_{cx}	r_{cy}	θ_0	x_0	y_0
M01	25.8069	7.20684	25.4333	25.5819	43.7612	17.3124	4.32806	-18.4544	56.3919
M02	14.3145	2.21117	14.3145	14.3145	2.17436	0.022209	0.0	0.0	0.0
M03	2.61459	1.0348	1.82689	2.20789	1.25092	0.447341	5.8268	0.0991696	1.3288

Table 3.5: Best feasible solutions found by ECA.

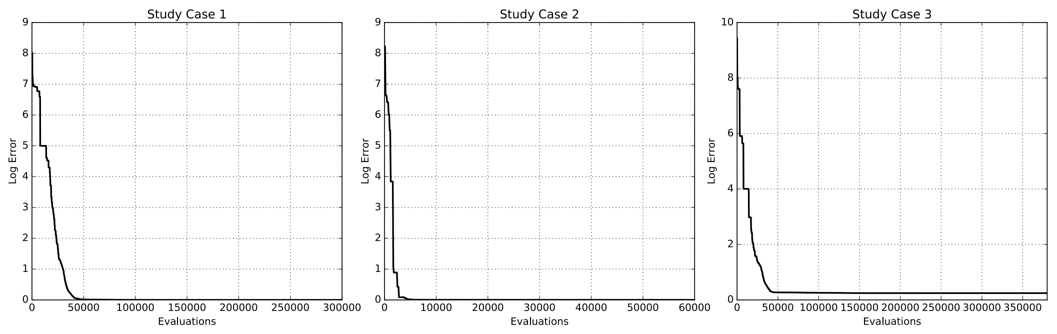


Figure 3.4: Convergence at median. ECA provides fast convergence in the three study cases.

3.2 Control of a Four-Bar Mechanism with Spring and Damping Forces

In the preview section, the angular velocity of the actuator is constant is the main assumption in this problem. That assumption is not always achieved, if an electric motor moves the crank. For example, the center of mass of FBM can move when the crank rotates. Also, the angular velocity of the crank is not constant when the inertia of the FBM yields an external load to the motor. Thus, we need a control system that guarantees an efficient and uniform adjustment of the angular velocity [18].

3.2.1 Dynamic Model

One degree of freedom (DoF) in the crank is considered in the FBM with spring and damping forces (FBM-SDF) (see Figure 3.5). Here, a DC motor is used to drive this DoF.

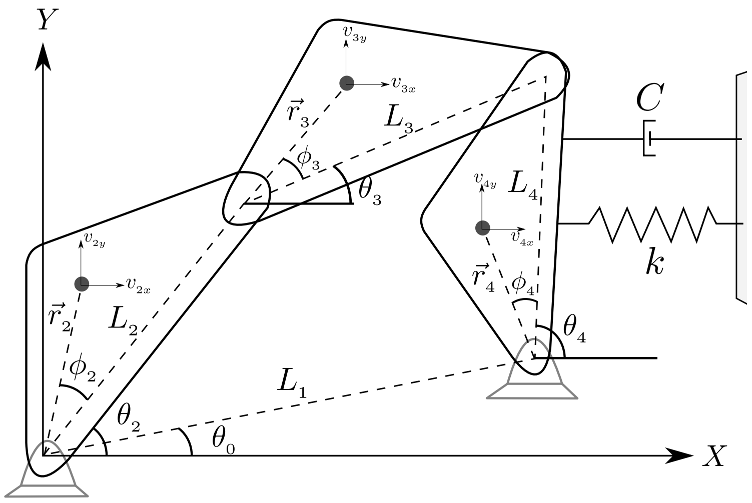


Figure 3.5: Representation of a four-Bar mechanism with spring and damping forces. Here, the $i-$th link is represented by its mass m_i, inertia J_i, length L_i, center of mass length r_i and center of mass angle ϕ_i. The angle of the $i-$th link associated to the base reference (X, Y) is set as θ_i; C and k are the the damping coefficient of the damper and constant of the spring, respectively.

Angular velocity $\dot{\theta}_i$, $i = 2, 3, 4$ and the linear velocity v_{ix}, v_{iy} of the center of mass of the $i - th$ link with respect to the inertial frame are:

$$\dot{\theta} = \gamma_i \dot{\theta}_2 \tag{3.20}$$
$$v_{ix} = \alpha_i \dot{\theta}_2 \tag{3.21}$$
$$v_{iy} = \beta_i \dot{\theta}_2, \tag{3.22}$$

where

$$\alpha_2 = -r_2 \sin(\theta_2 + \phi_2) \tag{3.23}$$

$$\alpha_3 = -L_2 \sin\theta_2 - r_3\gamma_3 \sin(\theta_3 + \phi_3) \tag{3.24}$$

$$\alpha_4 = -r_4\gamma_4 \sin(\theta_4 + \phi_4) \tag{3.25}$$

$$\beta_2 = r_2 \cos(\theta_2 + \phi_2) \tag{3.26}$$

$$\beta_3 = L_2 \cos\theta_2 - r_2\gamma_3 \cos(\theta_2 + \phi_3) \tag{3.27}$$

$$\beta_4 = -r_4\gamma_4 \cos(\theta_4 + \phi_4) \tag{3.28}$$

$$\gamma_2 = 1 \tag{3.29}$$

$$\gamma_3 = \frac{L_2 \sin(\theta_4 - \theta_2)}{L_3 \sin(\theta_3 - \theta_4)} \tag{3.30}$$

$$\gamma_4 = \frac{L_2 \sin(\theta_3 - \theta_2)}{L_3 \sin(\theta_3 - \theta_4)}. \tag{3.31}$$

Let $\tilde{L}$ be the Lagrangian function, where P and K is potential and the kinematic energy, respectively:

$$\tilde{L} = K - P,$$

where

$$K = \sum_{i=2}^{4} \left(\frac{1}{2} m_i (v_{ix}^2 + v_{iy}^2) + \frac{1}{2} J_i \dot{\theta}_i^2 \right) = \frac{1}{2} A(\theta_2) \dot{\theta}_2^2 \tag{3.32}$$

$$P = \frac{1}{2} k (\theta_4 - \theta_{4,0})^2 \tag{3.33}$$

$$A(\theta_2) = \sum_{i=2}^{4} (m_i(\alpha_i^2 + \beta_i^2) + \gamma_i^2 J_i). \tag{3.34}$$

The coupled dynamics of the DC motor with the FBM-SDF is given by combining the dynamic equation of the DC motor:

$$T_b = nK_f i(t) - n^2 B\dot{\theta}_2 - n^2 J\ddot{\theta}_2 \tag{3.35}$$

$$L\frac{di(t)}{dt} + Ri(t) = V_{in}(t) - nk_b\dot{\theta}_2 \tag{3.36}$$

and

$$T = A(\theta_2)\ddot{\theta}_2 + \frac{1}{2}\frac{dA(\theta_2)}{d\theta_2}\dot{\theta}_2^2 + k\gamma_4(\theta_4 - \theta_{4,0}) + C\gamma_4^2\dot{\theta}_2$$

where

$$A(\theta_2) = C_0 + C_1\gamma_3^2 + C_2\gamma_4^2 + C_3\gamma_3\cos(\theta_2 - \theta_3 - \phi_3) \tag{3.37}$$

$$\frac{dA(\theta_2)}{d\theta_2} = 2C_1\gamma_3\frac{d\gamma_3}{d\theta_2} - 2C_2\gamma_4\frac{d\gamma_4}{d\theta_2}. \tag{3.38}$$

C_i, $\frac{d\gamma_3}{d\theta_2}$ and $\frac{d\gamma_4}{d\theta_2}$ are detailed in [18]. Assume that $\boldsymbol{x} = [x_1, x_2, x_3]^T = [\theta_2, \dot{\theta}_2, i]^T$ and the input vector $u = V_{in}$, the state space representation for the coupled dynamics of the DC motor in the FBM-SDF is given by:

$$\dot{x} = f(\boldsymbol{x}, u(t), t) \tag{3.39}$$

$$= \begin{bmatrix} x_2 \\ A_0(x_2^2 + A_2x_2 + nK_fx_3 + A_3] \\ \frac{1}{L}(u(t) - nK_bx_2 - Rx_3) \end{bmatrix} \tag{3.40}$$

where

$$A_0 = \frac{1}{A(x_1) + n^2J_1} \tag{3.41}$$

$$A_1 = -\frac{1}{2}\frac{d}{A(x_1)}dx_1 \tag{3.42}$$

$$A_2 = -(C\gamma_2^4 + n^2B) \tag{3.43}$$

$$A_3 = -k\gamma_4(\theta_4 - \theta_{4,0}). \tag{3.44}$$

3.2.2 Optimization Problem Definition

In 2013, Calva-Yáñez et al [18] based its strategy on the work of Tao and Sadler [82]. The proposed control strategy is used in this work as well. The controller is stated as follow:

$$u(t) = K_pe(t)\int_0^t \dot{\theta}_2^d dt + K_I\int_0^t e(t)dt + K_D\dot{e}(t), \tag{3.45}$$

where K_p, K_I, K_D is the proportional, integral and derivate gains, respectively. Here, $\dot{e}(t) = -\ddot{\theta}_2$ and $e(t) = \dot{\theta}_2^d - \dot{\theta}_2$, with $\dot{\theta}_2^d$ the constant desired speed.

The correct election of $K_p,\ K_d,\ K_I$ is important, since a wrong selection of the PID gains, the input velocity of the crank can be considerably affected. Thus, the design variable vector is $\boldsymbol{p} = [K_p, K_d, K_I]^T \in \mathbb{R}^3$, Note that the gains of the modified PID controller are included in $\boldsymbol{p}$.

The dynamic optimization problem from a Mechanical point of view, consists in finding the optimum design variables $\boldsymbol{p} \in \mathbb{R}^3$ such that:

$$\min_{\boldsymbol{p}\in\mathbb{R}^2} F(\boldsymbol{p}) \tag{3.46}$$

subject to:

$$\dot{x} = f(\boldsymbol{x}, u(\boldsymbol{p},\ t), t) \tag{3.47}$$

$$u(t) = K_p e(t) \int_0^t \dot{\theta}_2^d dt + K_I \int_0^t e(t)dt + K_D \dot{e}(t) \tag{3.48}$$

$$\boldsymbol{x}(0) = x_0 \tag{3.49}$$

$$g_1(\boldsymbol{x}) \leq 0 \tag{3.50}$$

$$g_2(\boldsymbol{x}) \leq 0 \tag{3.51}$$

$$p_{i,\min} \leq p \leq p_{i,\min}, \tag{3.52}$$

with objective function defined for $t_0 = \min\{t \in [0, t_f] \mid x_2(t) = \dot{\theta}_2^{\mathrm{d}}\}$:

$$F(\boldsymbol{p}) = \left| \max_{t\in[t_0,\ t_f]} x_2(t) - \min_{t\in[t_0,\ t_f]} x_2(t) \right|, \tag{3.53}$$

which may *measure* the change of the input speed of the crank. If $F \to 0$, then $x_2(t) = \dot{\theta}_2(t)$ goes constant in $[t_0,\ t_f]$. The first constraint is the solutions of the differential equation given by the dynamic model of the FBM-SDF where x_0 is the initial condition. This constraint provides the dynamic behavior of the system in the optimization problem. It is necessary establish the rise time t_r of the angular velocity of the crank $\dot{\theta}_2(t) < 0.1$ and the overshoot does not exceed of 1.7% of the desired angular velocity. Thus

$$g_1(t_r) = t_r \leq 0.1s \tag{3.54}$$

$$g_2(t_r) = \dot{\theta}_2(t_r) \leq \dot{\theta}_2^d + 0.017\dot{\theta}_2^d. \tag{3.55}$$

The next section presents the performance of Algorithm 3 solving the constrained optimization problem (3.46) – (3.52).

3.2.3 Experiments

To solve the dynamic optimization problem (3.46) – (3.52), the close loop system (3.48) is solved using the Runge-Kutta method [27] with initial condition $x_0 = [0,0,0]^T$ with desired velocity selected as $\theta_2^d = 30$ rad/s and the kinematic and dynamic parameters of the coupled dynamics detailed in [18]. The bounds of the design variable vector is defined as $\boldsymbol{p} \in [0.1,\ 50]^3 \subset \mathbb{R}^3$.

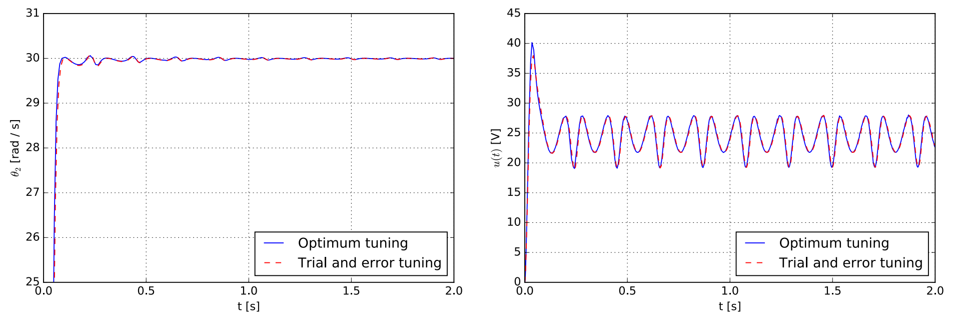
3.2.4 Results

Using the same experiment settings of Section 3.1.7, we use Algorithm 3 for solving the dynamic optimization problem (3.46) - (3.52). Here, the parameters were randomly chosen as described before. ECA is compared against the CHDE algorithm which is successfully applications for tunning the PID controller [18] and jSO, CMA-ES, CGSA described in Section 3.1.7.

	Evals.	**Best**	**Median**	**Mean**	**Std.**	**FR**	
ECA	10000	**2.055877E–01**	2.340816E–01	2.292241E–01	9.315045E–03	100	
CHDE	20000	2.120037E–01	**2.232873E–01**	**2.206308E–01**	8.093537E–03	100	$\approx$
jSO	10000	2.137288E–01	2.310883E–01	2.272267E–01	8.210808E–03	100	$\approx$
CMA-ES	10000	2.261747E–01	2.364060E–01	2.454021E–01	1.590420E–02	100	$\approx$
CGSA	10000	3.334081E–01	3.908357E–01	4.291667E–01	1.304116E–01	19	$+$

Table 3.6: Results of 31 independent runs of ECA, CHDE, jSO, CMA-ES and CGSA on the dynamic optimization problem (3.46) – (3.52). A result in **boldface** indicates the best values found for each correspondent column.

Statistical results obtained from 31 independent runs by ECA, CHDE, jSO, CMA-ES and CGSA are presented in Table 3.6. jSO, DE and CMA-ES do not show significant differences against ECA, based on the 95%-confidence Wilcoxon Rank-Sum test. Also, CGSA showed a poor performance. Hence, ECA obtained good results for this kind of problems. Besides, Algorithm 3 requires less evaluations of objective function to get good results compared with competitive state-of-art evolutionary algorithms.



(A) (B)

Figure 3.6: (A) Shows the angular crank velocity with time (seconds) in [0.05, 2] with both approaches. (B) shows the behavior of the control signal obtained by the optimum design variable vector and with the experimental tuning.

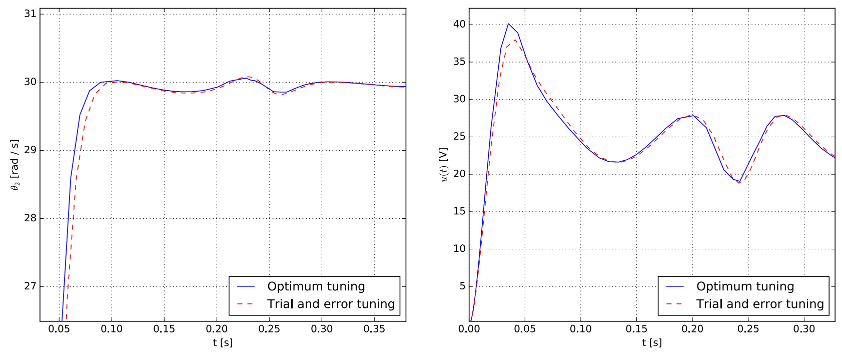


Figure 3.7: Zoom of each respective graphic in Figure 3.6. The rise time of the angular velocity of optimization approach is less than the experimental approach.

The Figure 3.6 (A) shows that the control signal of the manual approach has a lower overshoot than the second approach to reach the reference value of 30 rad/s.

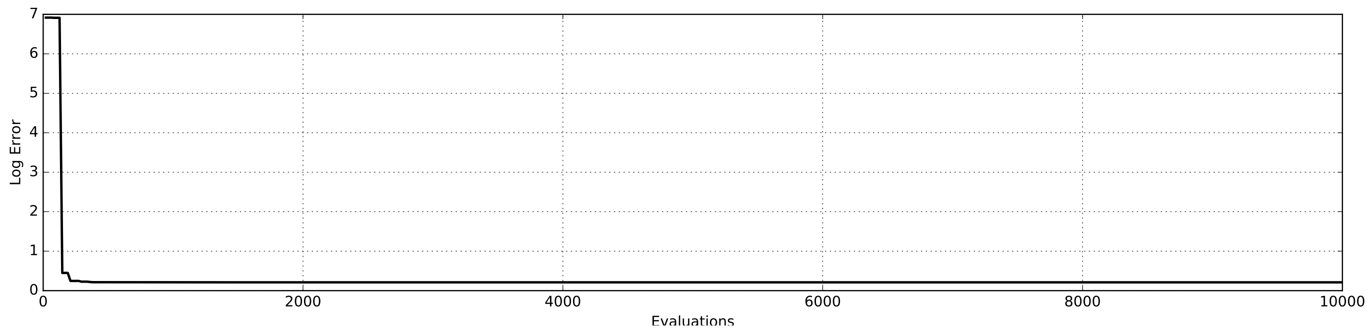


Figure 3.8: Convergence at median of 31 independent runs. Note the fast convergence in this problem.

Finally, Algorithm 3 was capable to converge fast (see Figure 3.8), which is convenient because this optimization problem is, relatively, computationally expensive, since for each constraint evaluation, it is necessary to solve the differential equations system (3.47).

3.3 Conclusions

In this chapter ECA was used to solve engineering problems with promising results. ECA was capable to solve those problems with fast convergence with good results and FR = 100%. Also, this approach outperformed CMA-ES and CGSA most of time. ECA obtained the minimum objective function value known and it is important in applications.

Chapter 4

Non-rigid Registration using ECA

We describe the application of ECA to an image processing problem, here we want to register two or more 2-D point set. The main idea is to find the parameter-based transformation via affine and quadratic polynomial transform. We provide a fast and accuracy algorithm for solving point-sets registration problem.

4.1 Introduction

Point set registration (PR) is commonly used in computer vision. The goal of point set registration is to estimate a transformation that maps one point set to the other. We need to consider some aspects for successful solution of this problem, i.e. large dimensionality of point set, outliers, noise, and transformation information. Also, point set registration can be used for Image Registration (IR). IR is an important research field in digital image processing. It is used to align a not empty set of images obtained, at different times, using different sensors or under different conditions [20, 30, 96].

The transformation can be rigid or non-rigid. Rigid registration does not change the distance between any two points, i.e., for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $T : \mathbb{R}^n \to \mathbb{R}^n$ a rigid transformation, then $d(\boldsymbol{x},\ \boldsymbol{y}) = d(T(\boldsymbol{x}),\ \boldsymbol{y})$, here d is a metric function. Usually, such a transformation consists of translation and rotation. Non-rigid transformation maps one point set to the other. The simplest non-rigid transformation is affine which also allows anisotropic scaling and skews. In mathematical terms PR can be defined as follows:

Definition 4.1.1. Point set registration is the process of finding a transformation $T : \mathbb{R}^n \to \mathbb{R}^n$, that aligns two finite point sets $A,\ B \subset \mathbb{R}^n$.

Here, *align* means that exists an injective function $\mu : A \to B$.

This chapter addresses the design of the mapping function for $n = 2$ by using affine transform and quadratic polynomials for accuracy. To solve this problem we use a Physics-inspired Metaheuristic [11]. The results obtained are superior by previous evolutionary approaches [6] on the same synthetic data set.

4.1.1 Related Work

This section presents a brief review of some of the most important approaches. There is variety of techniques for solving PR problems, i.e. classical methods and evolutionary algorithms based proposals.

Classical Methods

Most of the classical approaches for PR can be found in [7, 22, 31, 96]. Two famous state-of-the-art approaches from this category of methods are Random Sample Consensus (RANSAC) [31] and Robust Point Matching (TPS-RPM) [22].

RANSAC is an iterative algorithm used estimate parameters of a transformation from a set of data that contains outliers. It is assumed that outliers do not affect the values of the estimates. RANSAC is a stochastic method in the sense that it produces successful results whit certain probability measure. Also, RANSAC can be robust to estimate the model parameters and can be used for solving point set registration problem.

On the other hand, TPS-RPM [22] is a method for matching two point-sets in a deterministic annealing setting. TPS-RPM algorithms extend to non-rigid image registration by registering two sets of sparse features extracted from images. The intensity information of the extracted features are incorporated into feature matching in order to reduce the impact from outliers. It uses a fuzzy-like matrix instead of a binary permutation matrix to find the matching between two sets of points. In TPS-RPM, both the point correspondences and the transformations are computed interchangeably.

Evolutionary Methods

The first known application of evolutionary algorithms (EA) to image registration is reported in [32] where authors applied a genetic algorithm (GA) to medial images. For the subsequent, other EA approaches have been proposed by different authors, but most of them were based on the GAs. Such a GA has severe limitations when solving optimization problems in the continuous domain, especially due to the problem of Hamming cliffs originated from the discretization of real valued variables into binary coded values, to the fixed precision that depends on the number of bits used for each decision (design) variable, and for imposing lower and upper bounds for a variable's value. In [6] the authors use a real coded representation for a GA and got feasible results but with a high computational cost.

Most modern EA applications to PR use a direct real coded representation of solutions such as Particle Swarm Optimization [88] and others presented in this recent surveys on the topic [23, 74, 78].

Next section describes how ECA is used for solving PR problem (ECA-PR).

4.2 ECA-PR

This section presents the point set registration solution through the Evolutionary Centers Algorithm.

We need to match the point-sets as closely as possible while outliers are detected and ignored during matching process. The correspondence problem can be seen as as a linear assignment problem [62, 68]. The augmented part is used to take outliers into account. The correspondence matrix obtained by the linear assignment problem depends on the non-rigid transformation. Thus, the general problem solution is to minimize the following binary linear assignment-least squares error function:

$$\min_{\mu,\, T} E(\mu,\, T) = \min_{\mu,\, T} \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_{i,j} \|y_j - T(x_i)\|^2 \tag{4.1}$$

where matrix μ is the binary augmented correspondence matrix consisting of two parts. The inner $n \times m$ part of μ defines the correspondence. If a point y_j corresponds to a point x_j, $\mu_{i,j} = 1$, otherwise $\mu_{i,j} = 0$. Here, each row and column should contain a "1" to guarantee that the correspondence is one-to-one. An example of the correspondence matrix is given in Table 4.1. First, suppose $A,\ B \subset \mathbb{R}^n$ such that

$\mu_{i,j}$	x_1	x_2	x_3	x_4	outlier
y_1	0	0	0	0	1
y_2	0	0	0	0	1
y_3	0	1	0	0	0
y_4	0	0	1	0	0
y_5	0	0	0	1	0
outlier	1	0	0	0	0

Table 4.1: Points y_3, y_4 and y_5 correspond to $x_2,\ x_3$ and x_4, respectively, and the remaining points are outliers. μ is the binary correspondence matrix. Extra outlier row and outlier column are considered to identify outliers.

$|A| = |B| < \infty$, i.e., A and B are finite same-cardinality sets, and the correspondence matrix is known. Our objective is to find the best transformation T for solving this problem. We propose two kind of transformations:

- Affine transformations
- Low degree polynomial transformations

4.2.1 Affine Transformations

An affine transformation can be expressed by vector addition and matrix multiplication as shown in Equation 4.2,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = S \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix} \tag{4.2}$$

where S is the scaling parameter. By multiplying S with the rotation matrix, Equation 4.2 can be written as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix} \tag{4.3}$$

Finally, by using homogeneous coordinates, the affine transformation can be rewritten as Equation 4.3.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_5 \\ a_3 & a_4 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{4.4}$$

The affine transform has six parameters: $a_0,\ a_1,\ a_2,\ a_3,\ a_4$, and a_5. a_5 and a_6 specify the translation and a_1, a_2, a_3, and a_4 perform rotation, scaling, stretching, and shearing.

Now, we can define the transformation T_w as the matrix in Equation 4.4 with parameters $w = [a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6] \in \mathbb{R}^6$.

4.2.2 Low Degree Polynomial Transformations

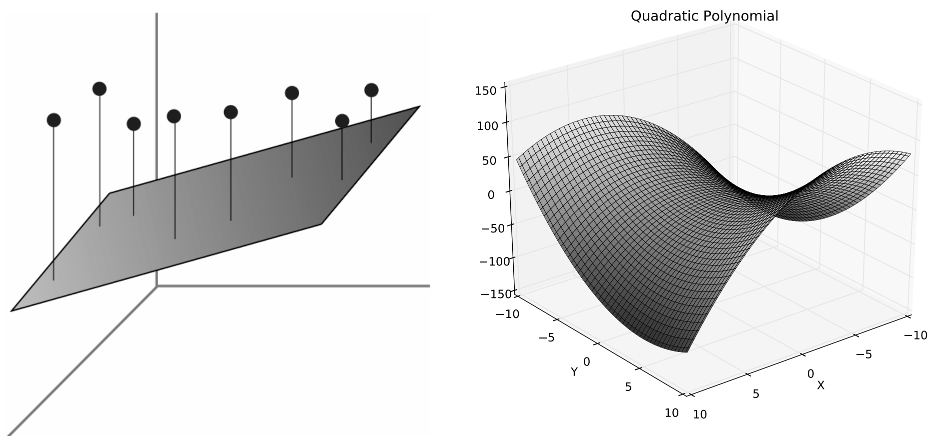


Figure 4.1: Left image shows how an affine transformation is performed. Right image represents a quadratic polynomial $p(x, y) = -0.62 + 0.87x - 0.85y - 0.80xy + 0.71x^2 - 0.87y^2$.

Here, we present a polynomial approach for point set registration for getting accuracy. Fist we define a quadratic polynomial transformation for $\mathbb{R}^2$ but can be easily extended to $\mathbb{R}^3$:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & \cdots & a_6 \\ b_1 & b_2 & \cdots & b_6 \end{bmatrix} \begin{bmatrix} y^2 \\ y^2 \\ xy \\ y \\ x \\ 1 \end{bmatrix} \tag{4.5}$$

$$= \begin{bmatrix} a_1x^2 + a_2y^2 + a_3xy + a_4x + a_5y + a_6 \\ b_1x^2 + b_2y^2 + b_3xy + b_4x + b_5y + b_6 \end{bmatrix}. \tag{4.6}$$

Note that if $a_1 = a_2 = a_3 = b_1 = b_2 = b_3 = 0$, then we have an affine transformation. Figure 4.1 shows a surface where points can be projected.

Finally, we can define the quadratic transformation as T_v as the matrix in Equation 4.5 where $v = [a_1, \ldots,\ a_6,\ b_1, \ldots,\ b_6] \in \mathbb{R}^{12}$. The next section presents some experiments.

4.3 Experiments

The PR minimization problem in Eq. 4.1 is solved using ECA (ECA-PR). Consider $A,\ B \subset \mathbb{R}^2$ where $A = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and $B = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m\}$:

- Affine approach
    1. Space Solution: $w = [a_1,\ a_2,\ \ldots,\ a_6] \in \mathbb{R}^6$.
    2. Objective function:

$$f(w;\ A,\ B,\ T_w) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{i,j} \|\boldsymbol{y}_j - T_w(\boldsymbol{x}_i)\|^2.$$

- Quadratic Approach

1. Space Solution: $v \in \mathbb{R}^{12}$.
2. Objective function:

$$f(v;\ A,\ B,\ T_v) = \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_{i,j} \|\boldsymbol{y}_j - T_v(\boldsymbol{x}_i)\|^2.$$

For each test-case, we supposed the μ matrix was given. Parameters for ECA were setted as in Algorithm 3, stop condition was $max\_evals = 1000D$ or std$(f(x)) < 10^{-10}$. The following Section presents some results using both approaches.

	RANSAC	**ECA using T_w**	**ECA using T_v**
Point-set 1	1.212963E–02	1.168271E–02	**1.606465E–03**
Point-set 2	2.662446E–03	2.624537E–03	**5.606001E–04**
Point-set 3	3.749500E–03	3.623729E–03	**1.236684E–03**
Point-set 4	3.748341E–03	3.622639E–03	**1.236255E–03**
Point-set 5	9.814332E–04	9.716634E–04	**1.819268E–04**

Table 4.2: Comparison of results between ECA-PR and a classical method. Note that ECA always outperformed RANSAC. This result was obtained running 31 times each algorithm and no variance was observed. A result in **boldface** indicates the best value found.

4.4 Results and Discussion

Using the previous information for experiments, we obtained some interesting results. Table 4.2 shows the minimum results for each transformation for each point-set. Note that, the quadratic approach always outperformed the affine approach.

Figures 4.2 and 4.3 plot approximated solutions for the registration problem using affine and quadratic transformations, respectively. We observed fast convergence on global optimal values of error, which is convenient for reducing computational cost.

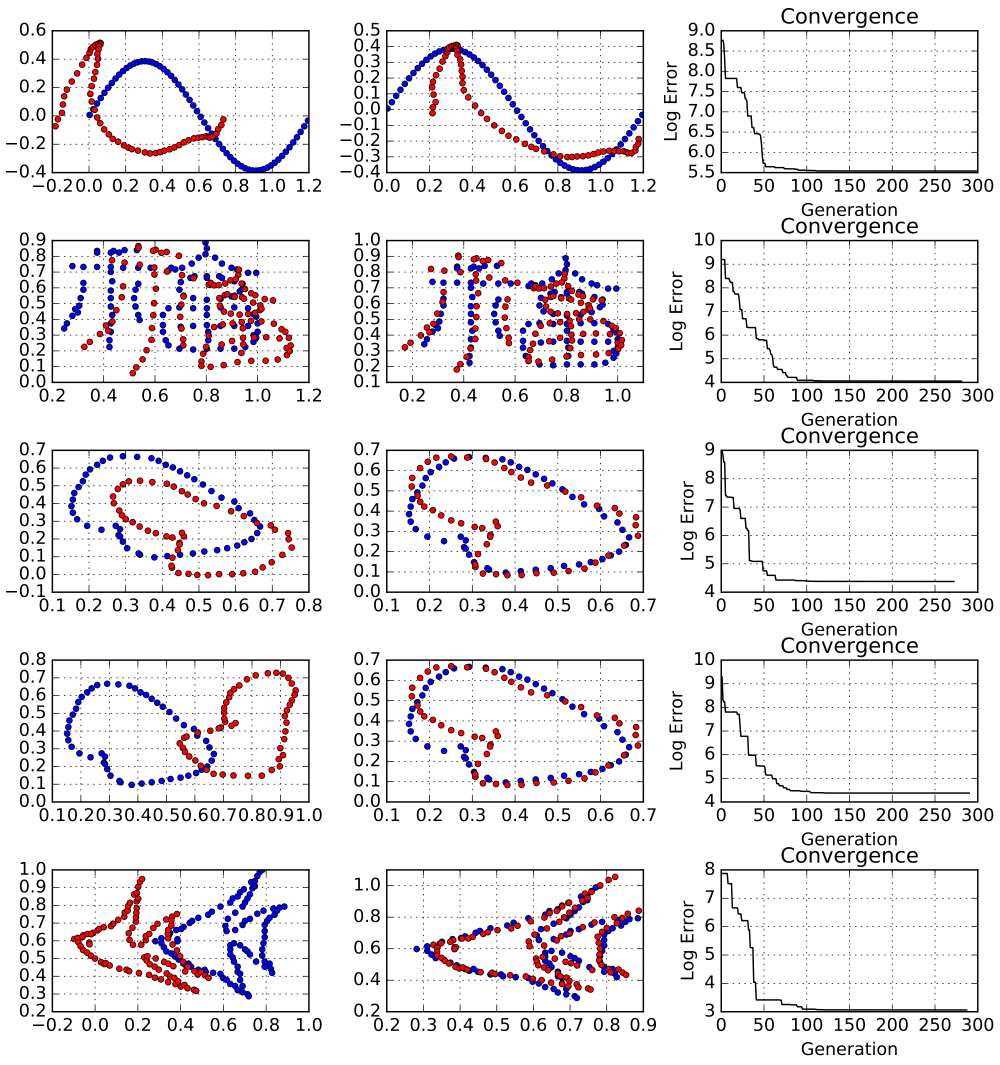


Figure 4.2: Results using affine transformation. Each row has a point-set input, output and convergence graph. Log error was computed for visualization purposes. Note fast convergence of ECA-PR using affine transform.

4.5 Conclusions

This chapter presented a new method based on ECA for solving the point-sets registration problem. Two strategies were detailed and compared against the RANSAC

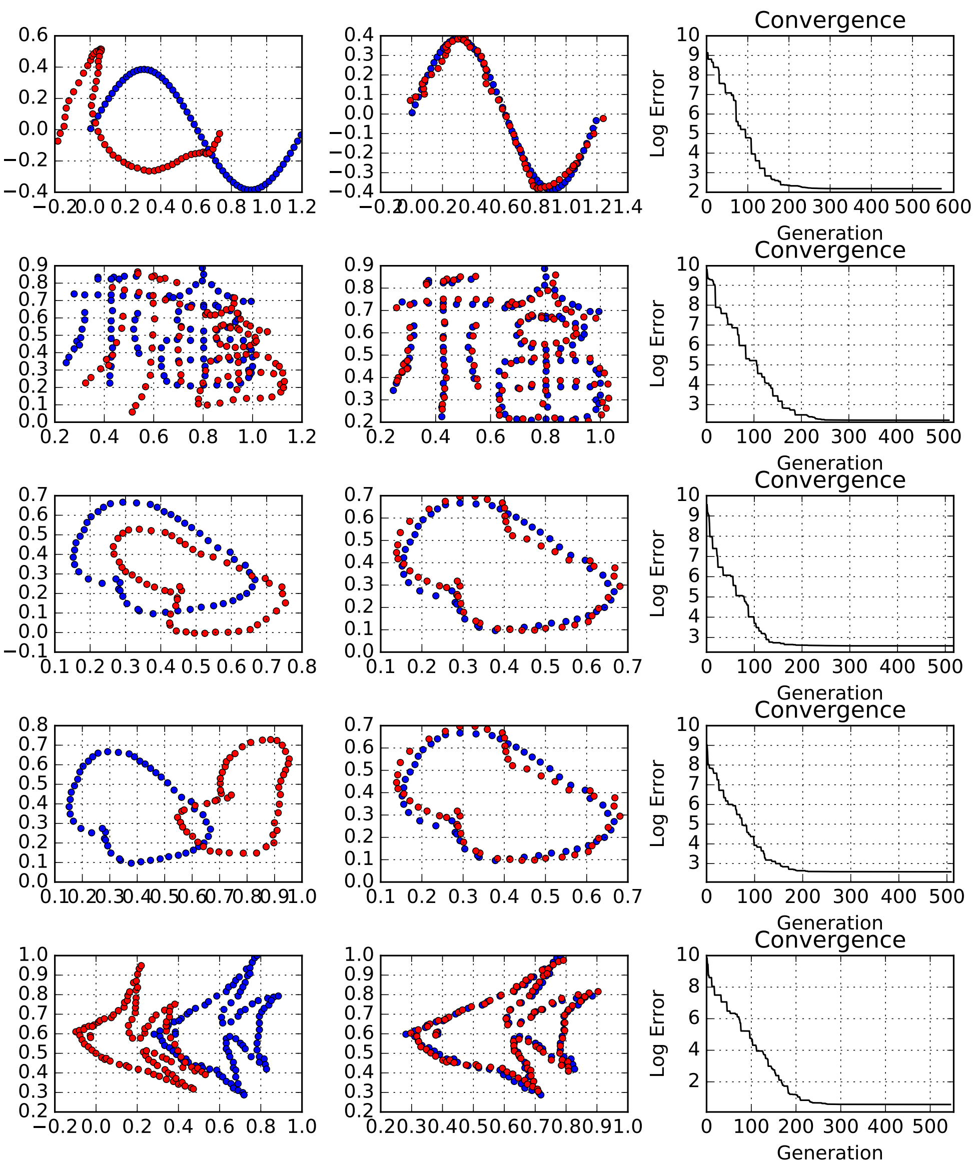


Figure 4.3: Results using quadratic transformation. Each row has a point-set input, output and convergence graph. Log error was computed for visualization purposes. Note fast convergence of ECA-PR using quadratic polynomials.

a competitive classical method. We obtained more accuracy and fast convergence using ECA-PR on well known point-sets.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This work presented a novel physics-inspired metaheuristic based on center of mass. Thirty test functions of a recent benchmark were employed in order to analyze the performance of the proposed algorithm in terms of precision, objective function evaluations and convergence. The results showed that ECA was able to provide highly competitive results compared to representative metaheuristics such as jSO, CMA-ES, CGSA. First, the results showed a good balance exploration-exploitation of the ECA algorithm, since the exploration ability of ECA was confirmed by the results.

Regarding engineering problems, ECA obtained the best known value in all problems. Moreover, our approach showed good performance on a low-dimensional problem but with a computational expensive objective function. Besides, ECA was capable to solve those problems with fast convergence with good results and feasibility ratio of 100%.

Finally, after this current research the hypothesis planted is validated since the given objectives were satisfied i.e. ECA was capable to solve complex problems with competitive results compared by using a non-parametric statistical hypothesis test. Also, ECA requires the fine-tuning of **just three parameters**, besides the population size. Therefore, ECA is simple (few parameters) but efficient in terms of the objective function evaluations.

5.2 Future Work

The future paths of research are the following:

- Make a mathematical study in order to to calibrate the parameters K, $\eta_{\max}$ and P_{exploit} or implement a self-adaptive technique.
- Study other constraint-handling techniques to improve results.
- Solve other constrained optimization problems.
- Propose mechanisms to improve ECA performance in high dimensionalities e.g. 100D.
- Implement a strategy for calculating the correspondence matrix, apply ECA-PR to real-world image registration problems and extend this algorithm for 3D points.
- Compare with other algorithms to verify the robustness of ECA-PR.

Appendices

Appendix A

CEC17 Benchmark

The CEC17 Benchmark is proposed in [3] which includes thirty test functions for global optimization. It contains three unimodal functions, seven simple multimodal functions, ten hybrid functions and ten composition functions. These problems where treated as black-box problems. This set of functions are shifted and rotated. The search range is $[-100,\ 100]^D$ where $D = 10, 30, 50$ and 100.

1. Bent Cigar Function

$$f_1(\boldsymbol{x}) = x_1 + 10^6 \sum_{i=2}^{D} x_i^2.$$

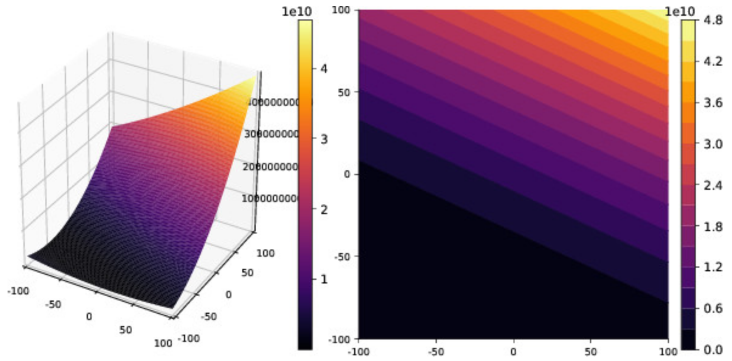


Figure A.1: Graph of $f_1(x,\ y)$ and its respective level curves.

2. Sum of Different Power Function

$$f_2(\boldsymbol{x}) = \sum_{i=1}^{D} |x_i|^{i+1}.$$

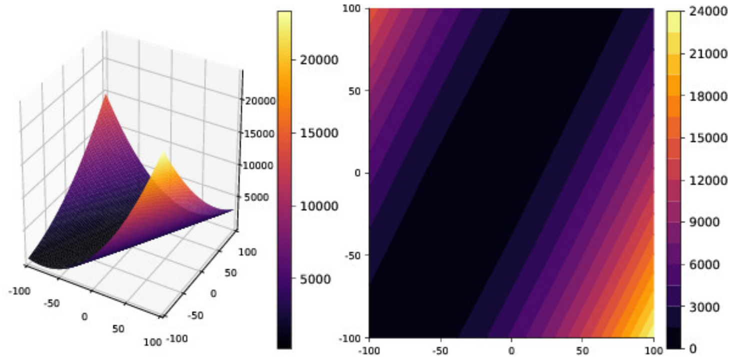


Figure A.2: Graph of $f_2(x,\ y)$ and its respective level curves.

3. Zakharov Function

$$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4.$$

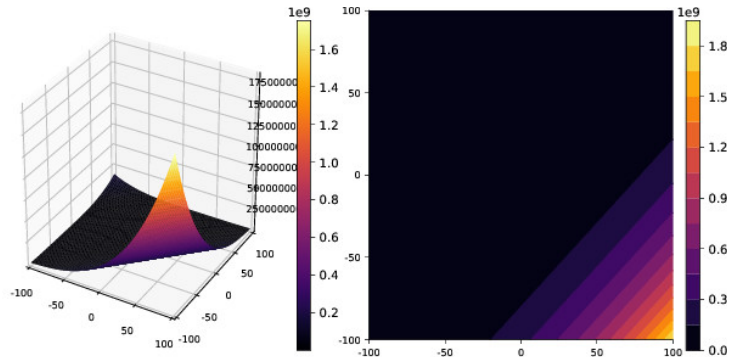


Figure A.3: Graph of $f_3(x,\ y)$ and its respective level curves.

4. Rosenbrock's Function

$$f_4(\boldsymbol{x}) = \sum_{i=1}^{D-1}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)].$$

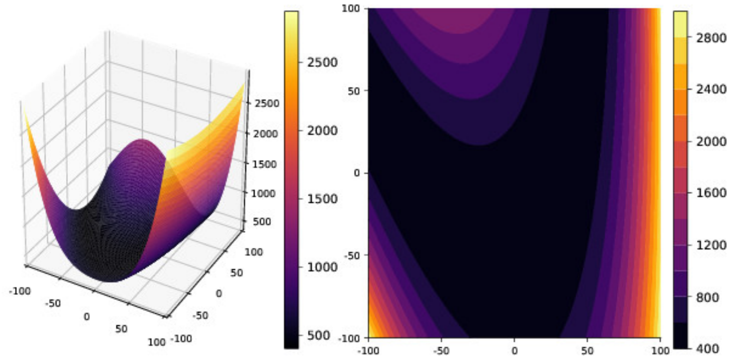


Figure A.4: Graph of $f_4(x,\ y)$ and its respective level curves.

5. Rastrigin Function

$$f_5(\boldsymbol{x}) = 10D + \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i)).$$

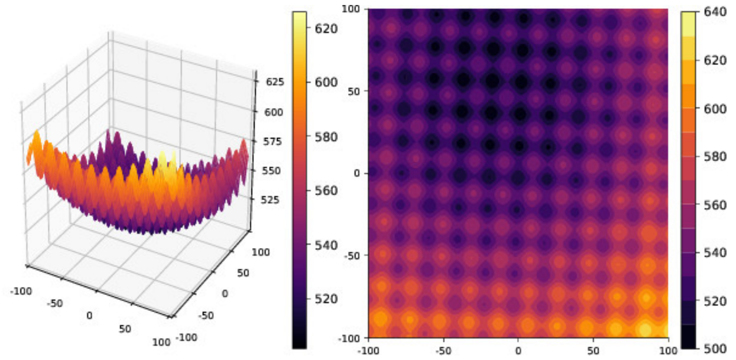


Figure A.5: Graph of $f_5(x,\ y)$ and its respective level curves.

6. Expanded Schaffer's F6 Function

$$f_6(\boldsymbol{x}) = g(x_1,\ x_2) + g(x_2,\ x_3) + \ldots + g(x_{D-1},\ x_D) + g(x_D,\ x_1),$$

where $g(x,\ y) = \dfrac{\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{(1+0.001(x^2+y^2))^2}.$

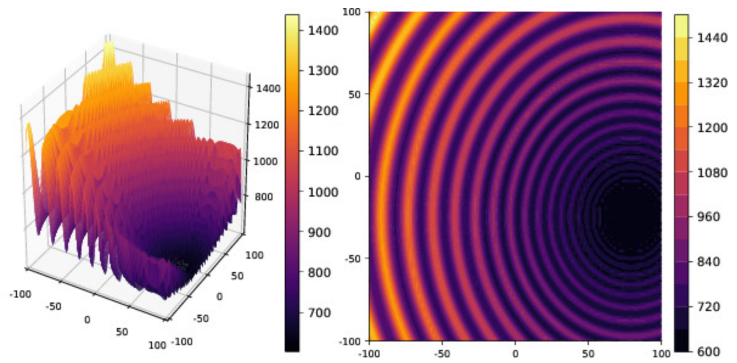


Figure A.6: Graph of $f_6(x,\ y)$ and its respective level curves.

7. Lunacek bi-Rastrigin Function

$$f_7(\boldsymbol{x}) = \min\left(\sum_{i=1}^{D}(xi-\mu_0)^2, dD + s\sum_{i=1}^{D}(x_i-\mu_1)^2) + 10(D - \sum_{i=1}^{D} cos(2\pi z_i))\right),$$

where

$$\mu_0 = 2.5, \qquad \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, \qquad s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, \qquad d = 1.$$

8. Non-continuous Rotated Rastrigin's Function

$$f_8(\boldsymbol{x}) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{13}*,$$

where

$$x = \frac{5.12}{100}M_1(x-o), \quad y_i = \begin{cases} x_i & \text{if } x_i \leq 0.5 \\ \text{round}(2x)/2, & \text{if } x_i > 0.5 \end{cases}, \quad z = M_1\Lambda^{10}M_2T_{asy}^{0.2}(T_{osy}(y))$$

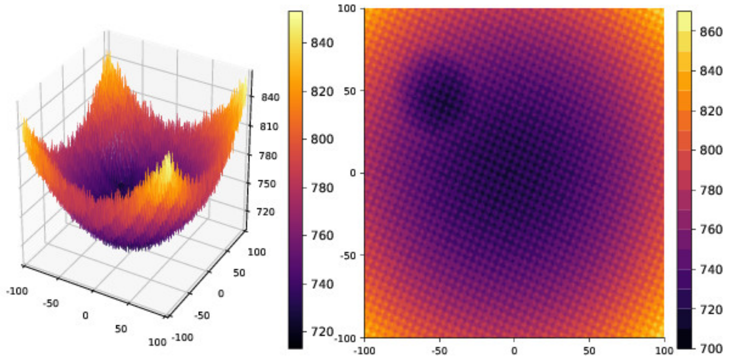


Figure A.7: Graph of $f_7(x,\ y)$ and its respective level curves.

where Λ^{α} is a diagonal D-dimensional matrix with i-th diagonal as $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}$ and $i = 1, \dots, D$ and $T_{asy},\ T_{osy}$ are defined in [3].

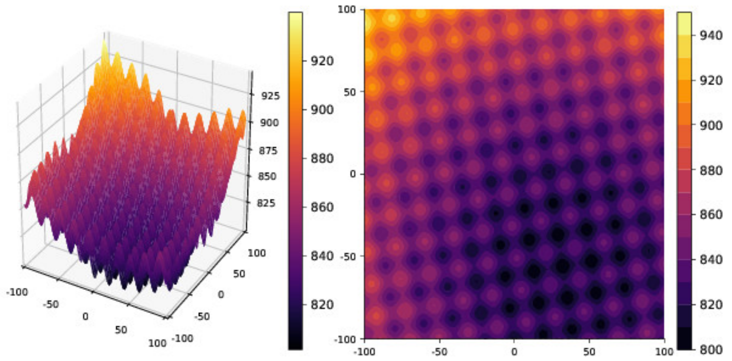


Figure A.8: Graph of $f_8(x,\ y)$ and its respective level curves.

9. Levy Function

$$f_9(\boldsymbol{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i-1)^2[1+10\sin^2(\pi w_i+1)] + (w_{D-1})^2[1+\sin^2(2\pi w_D)],$$

where $w_i = 1 + \dfrac{x_i - 1}{4}$ with $i = 1, \dots, D$.

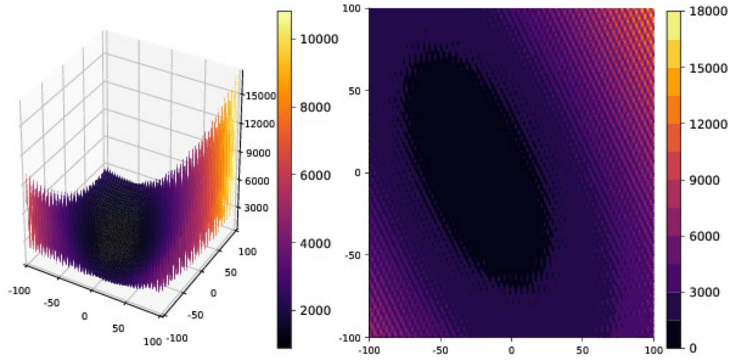


Figure A.9: Graph of $f_9(x,\ y)$ and its respective level curves.

10. Modified Schwefel's Function

$$f_{10}(\boldsymbol{x}) = 418.9829D - \sum_{i=1}^{D} g(\boldsymbol{z}),$$

where $z_i = x_i + 420.9687462275036$ and

$$g(z) = \begin{cases} z \sin\sqrt{|z|} & \text{if } |z| \leq 500 \\ (500 - \text{mod}(z,\ 500))h(z) - \dfrac{(z-500)^2}{100000D} & \text{if } z > 500 \\ (\text{mod}(z,\ 500) - 500)h(z) - \dfrac{(z-500)^2}{100000D} & \text{if } z < -500 \end{cases}$$

with $h(z) = \sin\sqrt{|500 - \text{mod}(|z|,\ 500)|}$.

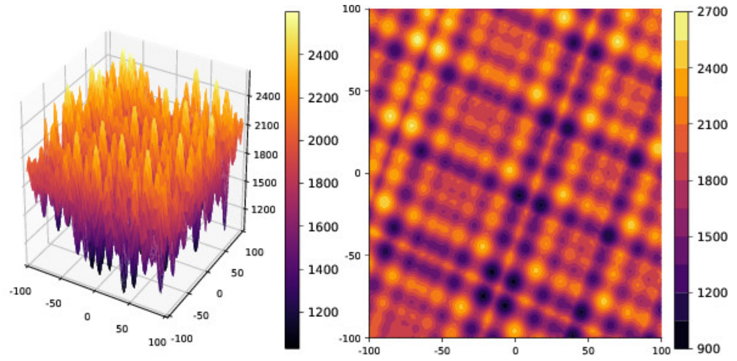


Figure A.10: Graph of $f_{10}(x,\ y)$ and its respective level curves.

11. High Conditioned Elliptic Function

$$f_{11}(\boldsymbol{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2.$$

12. Discus Function

$$f_{12}(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2.$$

13. Ackley's Function

$$f_{13}(\boldsymbol{x}) = -20 \exp\left(-0.2 \frac{1}{D} \sum_{i=1}^{D} x_i\right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e.$$

14. Weierstrass Function

$$f_{14}(\boldsymbol{x}) = \sum_{i=1}^{D} \left(\sum_{k=1}^{k_{\max}} [\cos(2\pi b^k (x_i + 0.5))]\right) - D \sum_{k=1}^{k_{\max}} [a^k \cos(\pi b^k)],$$

with $a = 0.5,\ b = 3$ and $k_{\max} = 20$.

15. Griewank's Function

$$f_{15}(\boldsymbol{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1.$$

16. Katsuura Function

$$f_{16}(\boldsymbol{x}) = \frac{10}{D^2} \left[\prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{D} \frac{|2^j x_i - \text{round}(2^j x_i)|}{2^j}\right)^{\frac{10}{D^{1.2}}} - 1\right].$$

17. HappyCat Function

$$f_{17}(\boldsymbol{x}) = \left|\sum_{i=1}^{D} x_i^2 - D\right|^{\frac{1}{4}} + \frac{1}{D} \sum_{i=1}^{D} \left(\frac{1}{2} x_i^2 + x_i\right) + \frac{1}{2}.$$

18. HGBat Function

$$f_{18}(\boldsymbol{x}) = \left| \left(\sum_{i=1}^{D} x_i^2 \right)^2 - \left(\sum_{i=1}^{D} x_i \right)^2 \right| + \frac{1}{D} \sum_{i=1}^{D} \left(\frac{1}{2} x_i^2 + x_i \right) + \frac{1}{2}.$$

19. Expanded Griewank's plus Rosenbrock's Function

$$f_{19}(\boldsymbol{x}) = f_7(f_4(x_1,\, x_2)) + f_7(f_4(x_2,\, x_3)) + \ldots + f_7(f_4(x_{D-1},\, x_D)) + f_7(f_4(x_D,\, x_1)).$$

20. Schaffer's f7 Function

$$f_{20}(\boldsymbol{x}) = \left[\frac{1}{D-1} \sum_{i=1}^{D-1} a_i [\sin\left(50 a_i^{0.2}\right) + 1] \right]^2,$$

where $a_i = \sqrt{x_i^2 + x_{i+1}^2}$.

Composition Functions

The ten remaining functions are composition of the previous mappings and they are detailed in [3].

Appendix B

Tables

f_n	Best	Median	Mean	Worst	Std.
1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
3	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
4	0.0000e+00	0.0000e+00	2.7712e-01	1.5920e+00	4.7483e-01
5	4.1884e+00	8.5518e+00	8.8980e+00	1.2867e+01	1.8272e+00
6	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
7	1.4953e+01	2.1225e+01	2.0989e+01	2.5910e+01	2.4926e+00
8	4.1977e+00	8.8088e+00	8.9524e+00	1.3975e+01	2.4317e+00
9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
10	2.8755e+02	6.8806e+02	6.6419e+02	9.3661e+02	1.7466e+02
11	0.0000e+00	0.0000e+00	9.8559e-05	3.0553e-03	5.4875e-04
12	0.0000e+00	2.0814e-01	2.4972e-01	6.4993e-01	1.9871e-01
13	0.0000e+00	1.3367e-02	4.6841e-01	5.5619e+00	1.3757e+00
14	0.0000e+00	0.0000e+00	6.4191e-02	9.9496e-01	2.4847e-01
15	4.5694e-07	2.4533e-02	1.7697e-01	4.9968e-01	2.2015e-01
16	6.2576e-02	7.0281e-01	7.2302e-01	1.6389e+00	3.1107e-01
17	1.1852e+00	4.8317e+00	1.1581e+01	2.9586e+01	9.8339e+00
18	8.0368e-05	2.9749e-01	2.6222e-01	5.0000e-01	2.1212e-01
19	0.0000e+00	2.1551e-04	1.0853e-02	4.6836e-02	1.3883e-02
20	0.0000e+00	3.0546e-05	2.9050e-01	1.3071e+00	4.2795e-01
21	1.0000e+02	1.0000e+02	1.1406e+02	2.1194e+02	3.7128e+01
22	1.0000e+02	1.0000e+02	1.0000e+02	1.0000e+02	0.0000e+00
23	3.0000e+02	3.0000e+02	3.0044e+02	3.0497e+02	1.1026e+00
24	0.0000e+00	1.0000e+02	1.6125e+02	3.3338e+02	1.0625e+02
25	3.9774e+02	3.9774e+02	3.9929e+02	4.4336e+02	8.1793e+00
26	3.0000e+02	3.0000e+02	3.0000e+02	3.0000e+02	0.0000e+00
27	3.9382e+02	3.9382e+02	3.9382e+02	3.9382e+02	0.0000e+00
28	3.0000e+02	3.0000e+02	3.0000e+02	3.0000e+02	0.0000e+00
29	2.3159e+02	2.4641e+02	2.4557e+02	2.6363e+02	6.9075e+00
30	3.9450e+02	3.9467e+02	3.9671e+02	4.0752e+02	4.8012e+00

Table B.1: Error values of 31 independent runs by jSO on CEC17 $10D$ problems.

f_n	Best	Median	Mean	Worst	Std.
1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
3	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
4	6.2824e-07	5.8562e+01	5.8509e+01	9.9233e+01	2.3109e+01
5	9.1343e+01	1.1040e+02	1.1035e+02	1.2761e+02	8.7782e+00
6	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
7	1.3077e+02	1.5071e+02	1.5046e+02	1.7051e+02	8.9760e+00
8	8.4033e+01	1.1291e+02	1.1282e+02	1.2853e+02	9.6769e+00
9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
10	5.5110e+03	5.9177e+03	5.9442e+03	6.4520e+03	2.3262e+02
11	6.7787e+00	1.8032e+01	1.7845e+01	2.4301e+01	3.7310e+00
12	1.6030e+02	4.4330e+02	4.6925e+02	1.4019e+03	2.8579e+02
13	1.0625e+01	4.8255e+01	4.6606e+01	7.0006e+01	1.4599e+01
14	3.4189e+01	4.2961e+01	4.2698e+01	5.1538e+01	3.8323e+00
15	3.8938e+00	1.3878e+01	1.3490e+01	2.1809e+01	4.9208e+00
16	3.1865e+02	7.7725e+02	7.7689e+02	1.1359e+03	1.6913e+02
17	7.0761e+01	1.4930e+02	1.4474e+02	1.9897e+02	3.3915e+01
18	2.0975e+01	2.4436e+01	2.4420e+01	2.9786e+01	1.8159e+00
19	6.7204e+00	1.8674e+01	1.5831e+01	2.2460e+01	4.9502e+00
20	3.9626e+01	1.4466e+02	1.4568e+02	3.2407e+02	6.8590e+01
21	2.8573e+02	3.0087e+02	3.0305e+02	3.1943e+02	9.9140e+00
22	1.0000e+02	1.0000e+02	1.0000e+02	1.0000e+02	0.0000e+00
23	4.1014e+02	4.3620e+02	4.3656e+02	4.5619e+02	1.3069e+01
24	4.5376e+02	4.9845e+02	4.9628e+02	5.1300e+02	1.2835e+01
25	3.8669e+02	3.8672e+02	3.8672e+02	3.8677e+02	2.2078e-02
26	2.0000e+02	1.6270e+03	1.4247e+03	1.8735e+03	4.9995e+02
27	4.7426e+02	4.8572e+02	4.8565e+02	4.9822e+02	6.6030e+00
28	3.0000e+02	3.0000e+02	3.0000e+02	3.0000e+02	2.5737e-13
29	4.5822e+02	6.5946e+02	6.2118e+02	7.2117e+02	7.8558e+01
30	1.9609e+03	2.0613e+03	2.0486e+03	2.1119e+03	4.3887e+01

Table B.2: Error values of 31 independent runs by jSO on CEC17 $30D$ problems.

f_n	Best	Median	Mean	Worst	Std.
1	0.0000e+00	0.0000e+00	2.4348e+08	2.8910e+09	7.6082e+08
2	0.0000e+00	0.0000e+00	1.0527e+02	3.2633e+03	5.8611e+02
3	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
4	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
5	9.9496e-01	1.5919e+01	2.3393e+01	7.4331e+01	2.0530e+01
6	5.0187e-05	1.8745e-01	6.1227e+00	4.6404e+01	1.3175e+01
7	1.3769e+01	2.1963e+01	2.1957e+01	3.2572e+01	4.8501e+00
8	2.9849e+00	1.0945e+01	1.1394e+01	3.2834e+01	6.2603e+00
9	0.0000e+00	8.9528e-02	6.8353e+00	1.7802e+02	3.1994e+01
10	2.5212e+02	1.2200e+03	1.1114e+03	1.6471e+03	3.7721e+02
11	4.9748e+00	2.9849e+01	5.5887e+01	4.3560e+02	8.1551e+01
12	1.1885e+02	5.6783e+02	2.0442e+06	6.1255e+07	1.0996e+07
13	5.3727e+01	3.3070e+02	3.5502e+02	7.0943e+02	1.7504e+02
14	2.4965e+01	7.0743e+01	1.2461e+02	6.2951e+02	1.4550e+02
15	9.0091e+00	5.5114e+01	1.1296e+02	1.3030e+03	2.2809e+02
16	1.1898e+00	2.4916e+02	2.3617e+02	5.6551e+02	1.7172e+02
17	2.4624e+01	5.2176e+01	6.9682e+01	2.8388e+02	4.7822e+01
18	2.2444e+01	9.3570e+01	2.1084e+05	1.7687e+06	4.5729e+05
19	7.1399e+00	5.8835e+01	9.5611e+02	1.1319e+04	2.3051e+03
20	2.4234e+01	1.2819e+02	1.2307e+02	2.7563e+02	5.3631e+01
21	1.1677e+02	1.7207e+02	1.7544e+02	2.2360e+02	3.6662e+01
22	1.0000e+02	1.0000e+02	1.0031e+02	1.0389e+02	7.1376e-01
23	3.1224e+02	3.8078e+02	3.7762e+02	4.0705e+02	2.0094e+01
24	1.0000e+02	3.3474e+02	3.1685e+02	3.7682e+02	4.9845e+01
25	3.9775e+02	4.4385e+02	4.3569e+02	4.5709e+02	2.0652e+01
26	2.0000e+02	3.0000e+02	3.4781e+02	1.2892e+03	1.9446e+02
27	3.9699e+02	4.1313e+02	4.2220e+02	4.9588e+02	2.2034e+01
28	3.0000e+02	5.8681e+02	5.4430e+02	6.8981e+02	1.0881e+02
29	2.5083e+02	3.4748e+02	3.6578e+02	5.4493e+02	6.9073e+01
30	4.8330e+02	2.8679e+03	7.1739e+05	4.0288e+06	1.0552e+06

Table B.3: Error values of 31 independent runs by CMA-ES on CEC17 $10D$ problems.

f_n	Best	Median	Mean	Worst	Std.
1	0.0000e+00	0.0000e+00	4.7213e+09	5.5575e+10	1.4742e+10
2	0.0000e+00	1.7464e+37	1.7176e+39	2.5103e+40	4.7641e+39
3	0.0000e+00	0.0000e+00	3.7971e+03	1.1771e+05	2.1141e+04
4	0.0000e+00	1.5617e+02	3.8031e+03	1.2130e+04	4.5722e+03
5	2.7859e+01	2.1889e+02	2.2655e+02	4.4773e+02	1.4786e+02
6	2.2917e+01	4.8519e+01	5.5748e+01	8.7972e+01	2.0231e+01
7	5.5909e+01	7.6139e+01	9.0464e+01	5.2049e+02	8.0555e+01
8	2.9849e+01	6.7657e+01	1.1130e+02	3.6675e+02	9.4586e+01
9	0.0000e+00	4.5145e+00	1.8762e+03	1.3862e+04	3.8586e+03
10	1.8833e+03	3.8881e+03	4.9945e+03	7.6331e+03	2.3182e+03
11	6.2689e+01	2.9132e+03	2.6729e+03	6.7577e+03	2.6280e+03
12	9.1629e+02	1.8411e+09	2.6107e+09	7.3741e+09	2.8467e+09
13	7.5800e+02	3.2186e+03	7.9236e+08	3.1380e+09	1.0467e+09
14	8.3677e+01	2.0506e+02	1.8592e+05	7.4125e+05	2.4513e+05
15	2.5067e+01	2.4378e+02	4.6134e+06	1.4301e+08	2.5685e+07
16	6.4913e+02	2.4912e+03	2.1963e+03	3.2139e+03	7.4147e+02
17	2.3996e+02	8.8977e+02	8.6677e+02	1.6097e+03	4.2595e+02
18	7.9524e+01	1.2781e+06	3.9128e+06	1.5495e+07	4.7239e+06
19	9.8518e+01	2.0524e+02	9.6085e+06	2.9786e+08	5.3496e+07
20	1.2848e+02	5.3867e+02	5.6785e+02	9.1704e+02	2.2955e+02
21	3.5205e+02	5.5018e+02	5.3913e+02	6.1428e+02	5.8369e+01
22	1.0000e+02	1.0000e+02	6.5648e+02	6.1915e+03	1.7296e+03
23	7.4745e+02	8.9287e+02	8.9144e+02	9.8895e+02	7.4890e+01
24	6.1659e+02	9.4600e+02	9.4080e+02	1.1600e+03	1.3029e+02
25	3.8347e+02	3.8737e+02	7.5352e+02	4.7251e+03	1.1353e+03
26	1.8830e+03	5.8685e+03	5.9769e+03	8.0520e+03	1.1832e+03
27	7.2685e+02	1.0237e+03	1.0240e+03	1.2305e+03	1.2525e+02
28	3.0000e+02	4.0329e+02	1.3230e+03	4.0790e+03	1.4808e+03
29	6.1559e+02	1.4837e+03	1.5870e+03	2.7880e+03	7.8002e+02
30	3.8195e+03	1.0411e+06	1.2426e+08	4.4009e+08	1.4371e+08

Table B.4: Error values of 31 independent runs by CMA-ES on CEC17 $30D$ problems.

f_n	Best	Median	Mean	Worst	Std.
1	2.0450e-01	1.8719e+02	1.3181e+07	4.0858e+08	7.3384e+07
2	5.8222e-07	1.8272e-05	1.9381e-05	7.5106e-05	1.5168e-05
3	2.1386e-07	9.3136e-07	1.1253e-06	3.6679e-06	9.4219e-07
4	4.0197e-03	5.9309e+01	5.7446e+01	1.5887e+02	3.9909e+01
5	1.6914e+01	2.5869e+01	2.6222e+01	4.0793e+01	6.1255e+00
6	4.2903e-04	1.7006e-03	5.1515e-01	9.4283e+00	1.7518e+00
7	1.1347e+01	1.6203e+01	1.6590e+01	2.1278e+01	2.6118e+00
8	5.9698e+00	1.4924e+01	1.5791e+01	2.4874e+01	4.7906e+00
9	1.0449e-08	1.1876e-07	1.2937e-07	4.3790e-07	8.5597e-08
10	5.9570e+02	1.2677e+03	1.3222e+03	2.0487e+03	3.1391e+02
11	9.9747e+00	2.9891e+01	3.5320e+01	9.3572e+01	1.8310e+01
12	2.1808e+03	6.9042e+03	7.4507e+03	1.8716e+04	3.8409e+03
13	2.2496e+03	6.9528e+03	7.4119e+03	1.6476e+04	3.2131e+03
14	7.0989e+01	3.4682e+03	3.6500e+03	7.4823e+03	2.2082e+03
15	2.1841e+01	2.2717e+03	2.5960e+03	6.7612e+03	1.8913e+03
16	3.5805e+02	4.5944e+02	4.5268e+02	7.2693e+02	9.3894e+01
17	2.9859e+01	9.5168e+01	1.6157e+02	4.7323e+02	1.2766e+02
18	3.9852e+02	6.1344e+03	1.9343e+05	2.2588e+06	4.7093e+05
19	1.1549e+02	3.8146e+03	3.7784e+03	1.1085e+04	2.3910e+03
20	1.4341e+02	1.6863e+02	2.2855e+02	4.6326e+02	9.8716e+01
21	1.0000e+02	2.2866e+02	2.2379e+02	2.6295e+02	3.5420e+01
22	1.0000e+02	1.0029e+02	1.1825e+02	2.3802e+02	3.3938e+01
23	3.4026e+02	3.9666e+02	4.0944e+02	5.9387e+02	5.8581e+01
24	1.0000e+02	2.0000e+02	2.3491e+02	4.5977e+02	1.3824e+02
25	3.9774e+02	4.4338e+02	4.3724e+02	5.1066e+02	2.5729e+01
26	2.0000e+02	2.0001e+02	2.3549e+02	3.0000e+02	4.8634e+01
27	4.3639e+02	5.0042e+02	4.9873e+02	5.3777e+02	2.3131e+01
28	3.0000e+02	6.3787e+02	6.3503e+02	7.1549e+02	6.9483e+01
29	2.5334e+02	3.2480e+02	4.0740e+02	6.8611e+02	1.2977e+02
30	1.5250e+03	3.5304e+03	6.4058e+03	3.0498e+04	6.5820e+03

Table B.5: Error values of 31 independent runs by CGSA on CEC17 $10D$ problems.

f_n	Best	Median	Mean	Worst	Std.
1	4.0052e+00	1.2572e+03	2.4301e+03	1.5765e+04	3.2219e+03
2	0.0000e+00	9.0230e-05	2.3449e+42	7.2690e+43	1.3055e+43
3	4.1817e-05	1.2110e-04	1.5727e-04	6.7661e-04	1.2690e-04
4	1.2820e+02	3.0658e+02	4.0259e+02	1.1981e+03	2.8388e+02
5	1.1144e+02	1.5024e+02	1.5178e+02	1.9800e+02	2.1970e+01
6	4.9447e+00	1.0949e+01	1.3647e+01	3.7366e+01	7.6915e+00
7	4.2762e+01	5.5302e+01	5.5260e+01	6.6419e+01	6.4683e+00
8	7.6612e+01	1.0845e+02	1.0948e+02	1.3929e+02	1.3298e+01
9	5.9129e-07	1.7427e-06	3.4286e-06	2.0236e-05	4.3907e-06
10	2.5954e+03	3.2780e+03	3.2854e+03	4.3571e+03	4.6095e+02
11	6.3695e+01	1.0150e+02	1.6805e+02	1.2938e+03	2.2536e+02
12	5.9100e+03	1.2220e+04	1.7286e+07	1.0225e+08	3.1156e+07
13	1.8867e+03	1.5305e+04	1.4608e+04	2.9076e+04	6.6137e+03
14	5.3173e+02	2.3510e+03	2.7151e+03	6.3779e+03	1.4139e+03
15	2.3171e+02	7.9532e+02	1.4326e+03	5.4236e+03	1.4256e+03
16	8.1421e+02	1.7243e+03	1.7109e+03	3.5874e+03	5.4419e+02
17	2.1330e+02	9.7264e+02	9.9596e+02	1.8236e+03	3.2713e+02
18	1.6082e+04	3.6757e+04	3.8069e+04	5.7744e+04	1.0429e+04
19	2.7133e+02	2.9584e+03	3.4305e+03	1.0608e+04	2.4010e+03
20	2.4616e+02	9.6457e+02	9.4309e+02	1.3566e+03	2.5204e+02
21	1.0000e+02	3.3759e+02	3.2436e+02	4.0225e+02	6.5411e+01
22	1.0000e+02	1.0001e+02	8.2481e+02	4.7791e+03	1.4807e+03
23	5.9630e+02	7.8617e+02	8.2813e+02	1.1037e+03	1.2769e+02
24	5.2020e+02	6.1352e+02	6.3770e+02	1.0603e+03	1.1618e+02
25	3.8355e+02	3.8748e+02	3.8961e+02	4.1236e+02	6.5821e+00
26	2.0002e+02	3.0001e+02	1.0957e+03	5.9157e+03	1.6793e+03
27	5.7993e+02	7.3908e+02	8.1208e+02	1.3202e+03	1.9583e+02
28	3.0000e+02	3.0000e+02	3.2042e+02	7.0723e+02	7.7190e+01
29	7.6682e+02	1.1836e+03	1.2617e+03	3.4789e+03	4.5115e+02
30	4.2943e+03	6.6753e+03	7.1514e+03	1.3438e+04	2.4074e+03

Table B.6: Error values of 31 independent runs by CGSA on CEC17 $30D$ problems.

Bibliography

[1] Hosein Abedinpourshotorban, Siti Mariyam Shamsuddin, Zahra Beheshti, and Dayang NA Jawawi. Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. *Swarm and Evolutionary Computation*, 26:8–22, 2016.

[2] Chang Wook Ahn. *Advances in evolutionary algorithms: theory, design and practice*, volume 18. Springer, 2007.

[3] N.H. Awad, M.Z. Ali, B.Y. Q., J.J. Liang, and P.N. Suganthan. Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. Technical report, 2016.

[4] D. Bajer, G. Martinović, and J. Brest. A population initialization method for evolutionary algorithms based on clustering and cauchy deviates. *Expert Systems with Applications*, 60:294–310, 2016.

[5] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic programming: an introduction*, volume 1. Morgan Kaufmann San Francisco, 1998.

[6] M. Bazargani, António dos Anjos, and F.G. Lobo. Affine image registration transformation estimation using a real coded genetic algorithm with sbx. *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 1459–1460, 2012.

[7] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *Sensor Fusion IV: Control Paradigms and Data Structures*, 1611:586–607, 1992.

[8] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[9] S. Binitha, S. Siva Sathya, and et al. A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*, 2(2):137–151, 2012.

[10] Ş İlker Birbil and Shu-Chering Fang. An electromagnetism-like mechanism for global optimization. *Journal of global optimization*, 25(3):263–282, 2003.

[11] A. Biswas, K.K. Mishra, S. Tiwari, and A.K. Misra. Physics-inspired optimization algorithms: A survey. *Journal of Optimization*, vol. 2013, 2013.

[12] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[13] Ilhem BoussaïD, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

[14] O. Bozorg-Haddad, Mohammad Solgi, and H. A. Loáiciga. Water cycle algorithm. *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization*, pages 231–240.

[15] J. Brest, M. Sepesy-Maučec, and B. Bošković. Single objective real-parameter optimization: Algorithm jso. *IEEE Congress on Evolutionary Computation (CEC)*, pages 1311 – 1317, 2017.

[16] J. Brownlee. *Clever algorithms: nature-inspired programming recipes*. Lulu.com, 2011.

[17] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[18] M. B. Calva-Yáñez, P. A. Niño-Suárez, M. G. Villarreal-Cervantes, G. Sepúlveda-Cervantes, and E. A. Portilla-Flores. Differential evolution for the control gain's optimal tuning of a four-bar mechanism. *Polibits*, (47):67–73, 2013.

[19] J. Cao and H. Gao. A quantum-inspired bacterial swarming optimization algorithm for discrete optimization problems. In *International Conference in Swarm Intelligence*, pages 29–36. Springer, 2012.

[20] L. Cao, H. Liu, and Y. Zhou. An image registration method for surgical robots based on human-robot cooperation. *IEEE International Conference on Mechatronics and Automation*, pages 1107–1112, 2016.

[21] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 76. John Wiley & Sons, 2013.

[22] H. Chui. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003.

[23] S. Damas, O. Cordón, and J. Santamaría. Medical image registration using evolutionary computation. *IEEE Computational Intelligence Magazine*, 6(4):26–42, 2011.

[24] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2-4):311–338, 2000.

[25] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[26] Dongsheng Ding, Donglian Qi, Xiaoping Luo, Jinfei Chen, Xuejie Wang, and Pengyin Du. Convergence analysis and performance of an extended central force optimization algorithm. *Applied Mathematics and Computation*, 219(4):2246–2259, 2012.

[27] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[28] O.K. Erol and Ibrahim. Eksin. A new optimization method: big bang–big crunch. *Advances in Engineering Software*, 37(2):106–111, 2006.

[29] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[30] B. Fischer and J. Modersitzki. Ill-posed medicine an introduction to image registration. *IOP Publishing Ltd Inverse Problems*, 24(3):034008, 2008.

[31] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[32] J. Fitzpatrick, J.J. Grefenstette, and D. Gucht. Image registration by genetic search. *Proceedings of IEEE Southeast Conference*, pages 460–464, 1984.

[33] P.J. Fleming and R.C. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control engineering practice*, 10(11):1223–1241, 2002.

[34] J. J. Flores, R. López, and J. Barrera. Gravitational interactions optimization. In *International Conference on Learning and Intelligent Optimization*, pages 226–237. Springer, 2011.

[35] R. A. Formato. Central force optimization: a new metaheuristic with applications in applied electromagnetics. *Progress in Electromagnetics Research*, 77:425–491, 01 2007.

[36] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 3261–3362. Springer, 2013.

[37] K. H. Han and J. H. Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE transactions on evolutionary computation*, 6(6):580–593, 2002.

[38] H. R. Hassanzadeh and M. Rouhani. A multi-objective gravitational search algorithm. In *Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on*, pages 7–12. IEEE, 2010.

[39] B. Hernández-Ocaña, M. P. Pozos-Parra, E. Mezura-Montes, E. A. Portilla-Flores, E. Vega-Alvarado, and M. B. Calva-Yáñez. Two-swim operators in the modified bacterial foraging algorithm for the optimal synthesis of four-bar mechanisms. *Computational intelligence and neuroscience*, 2016:17, 2016.

[40] M. Jamil and X.S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[41] Behzad Javidy, Abdolreza Hatamlou, and Seyedali Mirjalili. Ions motion algorithm for solving optimization problems. *Applied Soft Computing*, 32:72–79, 2015.

[42] S. Jiang, Z. Ji, and Y. Shen. A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints. *International Journal of Electrical Power & Energy Systems*, 55:628–644, 2014.

[43] Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, and Dusan Fister. A brief review of nature-inspired algorithms for optimization. *CoRR*, abs/1307.4186, 2013.

[44] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[45] Ali Husseinzadeh Kashan. A new metaheuristic for optimization: optics inspired optimization (oio). *Computers & Operations Research*, 55:99–125, 2015.

[46] A. Kaveh and M. Khayatazad. A new meta-heuristic method: Ray optimization. *Computers & structures*, 112:283–294, 2012.

[47] A. Kaveh and S. Talatahari. A novel heuristic optimization method: Charged system search. *Acta Mechanica*, 213(3):267–289, 2010.

[48] J. Kennedy. Particle swarm optimization. *IEEE International Conference on Neural Network*, pages 1942–1948, 1995.

[49] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[50] D. Kleppner and R. Kolenkow. *An Introduction to Mechanics*. McGraw-Hill, 2nd edition, 1973.

[51] W. Li, Q. Yin, and X. Zhang. Continuous quantum ant colony optimization and its application to optimization and analysis of induction motor structure. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, pages 313–317. IEEE, 2010.

[52] Y. Li and L. Jiao. Quantum-inspired immune clonal algorithm. In *International Conference on Artificial Immune Systems*, pages 304–317. Springer, 2005.

[53] Z. Li and Q. Zhang. An efficient rank-1 update for cholesky cma-es using auxiliary evolution path. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 913–920. IEEE, 2017.

[54] Yong Liu and Liang Ma. Improved gravitational search algorithm based on free search differential evolution. *Journal of systems Engineering and Electronics*, 24(4):690–698, 2013.

[55] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

[56] A. Malossini, E. Blanzieri, and T. Calarco. Quantum genetic optimization. *IEEE Transactions on Evolutionary Computation*, 12(2):231–241, 2008.

[57] M. J. McCarthy. *Geometric design of linkages*, volume 11. Springer Science & Business Media, 2006.

[58] Z. Michalewicz. Evolution strategies and other methods. In *Genetic algorithms+ data structures= evolution programs*, pages 159–177. Springer, 1996.

[59] S. Mirjalili and A. H. Gandomi. Chaotic gravitational constants for the gravitational search algorithm. *Applied Soft Computing*, 53:407–419, 2017.

[60] M. Mitchell. An introduction to genetic algorithms. *Cambridge, MA: MIT Press*, 1996.

[61] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[62] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[63] Venkataraman Muthiah-Nakarajan and Mathew Mithra Noel. Galactic swarm optimization: A new global optimization metaheuristic inspired by galactic motion. *Applied Soft Computing*, 38:771–787, 2016.

[64] D.H. Myszka. *Machines and Mechanisms, Applied Kinematic Analysis*. Pearson Prentice Hall, 2005.

[65] Ajit Narayanan and Mark Moore. Quantum-inspired genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 61–66. IEEE, 1996.

[66] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 2006.

[67] K. F. Pál. Hysteretic optimization for the sherrington–kirkpatrick spin glass. *Physica A: Statistical Mechanics and its Applications*, 367:261–268, 2006.

[68] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, Inc, 1982.

[69] M. D. Platel, S. Schliebs, and N. Kasabov. A versatile quantum-inspired evolutionary algorithm. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 423–430. IEEE, 2007.

[70] S. Rahnamayan, H.R.Tizhoosh, and M.M.A.Salama. A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, 53(10):1605–1614, 2007.

[71] S. S. Rao and S. S. Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

[72] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi. Bgsa: binary gravitational search algorithm. *Natural Computing*, 9(3):727–745, 2010.

[73] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248, 2009.

[74] J. Santamaría, O. Cordón, , and S. Damas. A comparative study of state-of-the-art evolutionary image registration methods for 3d modeling. *Computer Vision and Image Understanding*, 115(9):1340–1354, 2011.

[75] R.A. Serway and J.W. Jewett. *Principles of Physics: a Calculus-Based Text*. Thomson Learning, 4th edition, 2016.

[76] A. D. Sofge. Prospective algorithms for quantum evolutionary computation. *arXiv preprint arXiv:0804.1133*, 2008.

[77] K. Sörensen. Metaheuristics–the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015.

[78] A. Sotiras, C. Davatzikos, and N. Paragios. Deformable medical image registration: A survey. *IEEE transactions on medical imaging*, 32(7):1153–1190, 2013.

[79] James C. Spall. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons Inc., 2003.

[80] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Berkeley: ICSI*, 1995.

[81] J. Sun, W. Xu, and B. Feng. A global search strategy of quantum-behaved particle swarm optimization. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 111–116. IEEE, 2004.

[82] J. Tao and J. P. Sadler. Constant speed control of a motor driven mechanism system. *Mechanism and Machine Theory*, 30(5):737–748, 1995.

[83] A. Tom and A. Zilinskas. *Global Optimization*. Springer-Verlag New York, Inc., 1989.

[84] G. Toussaint. Simple proofs of a geometric property of four-bar linkages. *The American mathematical monthly*, 110(6):482, 2003.

[85] Christos Voudouris and Edward P. K. Tsang. Guided local search. In *Handbook of metaheuristics*, pages 185–218. Springer, 2003.

[86] R. Walter. *Principles of Mathematical Analysis*. International Series in Pure and Applied Mathematics. McGraw-Hill,, New York, 3rd edition, 1976.

[87] Y. Wang, X.Y. Feng, Y. X. Huang, D. B. Pu, W. G. Zhou, Y. C. Liang, and C. G. Zhou. A novel quantum swarm evolutionary algorithm and its applications. *Neurocomputing*, 70(4-6):633–640, 2007.

[88] S. Winter, B. Brendel, I. Pechlivanis, K. Schmieder, and C. Igel. Registration of ct and intraoperative 3-d ultrasound images of the spine using evolutionary and gradient-based methods. *IEEE Transactions on Evolutionary Computation*, 12(3):284–296, 2008.

[89] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[90] C. Wook-Ahn. *Advances in Evolutionary Algorithms*. Springer-Verlag, 2015.

[91] L. Xie, J. Zeng, and Z. Cui. The vector model of artificial physics optimization algorithm for global optimization problems. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 610–617. Springer, 2009.

[92] Liping Xie, Jianchao Zeng, and Zhihua Cui. General framework of artificial physics optimization algorithm. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 1321–1326. IEEE, 2009.

[93] D. H. Zerigat, L. Benasla, A. Belmadani, and M. Rahli. Galaxy-based search algorithm to solve combined economic and emission dispatch. *UPB Scientific Bulletin, Series C: Electrical Engineering*, 76(1):209–220, 2014.

[94] R. Zhang and H. Gao. Improved quantum evolutionary algorithm for combinatorial optimization problem. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 6, pages 3501–3505. IEEE, 2007.

[95] Yu Zhang, Lihua Wu, Ying Zhang, and Jianxin Wang. Immune gravitation inspired optimization algorithm. In *International Conference on Intelligent Computing*, pages 178–185. Springer, 2011.

[96] B. Zitová and J. Flusse. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.