Universidade Federal do Espírito Santo
Centro Tecnológico
Programa de Pós-Graduação em Informática

Mauro Cesar Martins Campos

Development of an Entropy-Based Swarm Algorithm for Continuous Dynamic Constrained Optimization

Vitória ES, Brazil
2017

Mauro Cesar Martins Campos

Development of an Entropy-Based Swarm Algorithm for Continuous Dynamic Constrained Optimization

Tese apresentada ao Programa de Pós-Graduação em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Vitória ES, Brazil
2017

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Campos, Mauro Cesar Martins, 1972-
C198d Development of an entropy-based swarm algorithm for continuous dynamic constrained optimization / Mauro Cesar Martins Campos. – 2017.
121f. : il.

Orientador: Renato Antonio Krohling.
Tese (Doutorado em Ciência da Computação) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Entropia. 2. Enxame de partículas – PSO (Particle Swarm Optimization) 3. Otimização em ambientes dinâmicos. 4. Índice de diversidade. 5. TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution). I. Krohling, Renato Antonio. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU:

Development of an Entropy-Based Swarm Algorithm for Continuous Dynamic Constrained Optimization

Mauro Cesar Martins Campos

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 08 de maio de 2017 por:

Renato Krohling

Prof. Dr. Renato Antônio Krohling (Orientador)
Universidade Federal do Espírito Santo - Brasil

Claudine Badue

Profª. Drª. Claudine Santos Badue Gonçalves (Examinador Interno)
Universidade Federal do Espírito Santo - Brasil

Eduardo Zambon

Prof. Dr. Eduardo Zambon (Examinador Interno)
Universidade Federal do Espírito Santo - Brasil

Prof. Dr. Helio José Corrêa Barbosa (Examinador Externo)
Laboratório Nacional de Computação Cientifica LNCC/MCT, RJ

Prof. Dr. Renato Tinós (Examinador Externo)
Universidade de São Paulo - USP/FFCLRP, SP

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória-ES, 08 de maio de 2017.

Agradecimentos

A Deus, meu Pai, por me abençoar em todos as situações da minha vida. Graças te dou por me proteger, me ouvir e, apesar das minhas limitações, me propor caminhos para ter uma vida em comunhão com o Senhor e com seu Filho, Jesus Cristo, único Intercessor e Salvador.

"Tomé disse: Senhor, não sabemos para onde vais. Como podemos conhecer o caminho? Jesus respondeu: Eu sou o caminho, a verdade e a vida. Ninguém vai ao Pai senão por mim. Se me conhecestes, conhecereis também o meu Pai. Desde já o conheceis e o tendes visto."
(Jo 14, 5-7)

Aos meus Pais (Joel e Maria) por tudo desde sempre. Pela vida, pelos esforços dedicados à minha criação e, principalmente, pelo amor incondicional em todos os momentos.

À minha linda namorada (Daniela) pelo amor, carinho, apoio e paciência, especialmente nos últimos anos.

Ao Professor Renato Krohling pela indicação do tema de tese, pela orientação durante o desenvolvimento dos estudos e trabalhos, e pela disponibilidade em todos os momentos.

Aos Membros da Banca Examinadora pela participação na defesa deste trabalho e também pelos questionamentos, contribuições e correções que melhoraram a versão final da tese.

Ao Programa de Pós-Graduação em Informática (PPGI) da Universidade Federal do Espírito Santo (UFES) pela estrutura acadêmica durante o desenvolvimento do trabalho.

Aos Professores do PPGI/UFES por tudo que aprendi nas disciplinas do programa. Oportunidade verdadeira para ampliar meus conhecimentos e crescer profissionalmente.

A UFES pela fundamental autorização de afastamento das minhas atividades docentes e administrativas, junto ao Departamento de Estatística (DEST) do Centro de Ciências Exatas (CCE) da Universidade, para fins de estudos e aperfeiçoamento no PPGI/UFES.

Aos Professores do DEST/CCE/UFES pelo apoio ao meu afastamento.

Statement of Originality

The research work contained in this thesis has not been previously submitted for any degree or diploma at any other university. This thesis is an original work and does not contain any material previously published by another person, except where appropriate references are made. It is the product of my own work and all sources and received assistance in its development have been cited and recognized.

Mauro Cesar Martins Campos
May 8, 2017
Vitória ES, Brazil.

Abstract

Dynamic constrained optimization problems form a class of problems where the objective function or the constraints can change over time. In static optimization, finding a global optimum is considered as the main goal. In dynamic optimization, the goal is not only to find an optimal solution, but also track its trajectory as closely as possible over time. Changes in the environment must be taken into account during the optimization process in such way that these problems are to be solved online. Many real-world problems can be formulated within this framework. This thesis proposes an entropy-based bare bones particle swarm for solving dynamic constrained optimization problems. The Shannon's entropy is established as a phenotypic diversity index and the proposed algorithm uses the Shannon's index of diversity to aggregate the global-best and local-best bare bones particle swarm variants. The proposed approach applies the idea of mixture of search directions by using the index of diversity as a factor to balance the influence of the global-best and local-best search directions. High diversity promotes the search guided by the global-best solution, with a normal distribution for exploitation. Low diversity promotes the search guided by the local-best solution, with a heavy-tailed distribution for exploration. A constraint-handling strategy is also proposed, which uses a ranking method with selection based on the technique for order preference by similarity to ideal solution to obtain the best solution within a specific population of candidate solutions. Mechanisms to detect changes in the environment and to update particles' memories are also implemented into the proposed algorithm. All these strategies do not act independently. They operate related to each other to tackle problems such as: diversity loss due to convergence and outdated memories due to changes in the environment. The combined effect of these strategies provides an algorithm with ability to maintain a proper balance between exploration and exploitation at any stage of the search process without losing the tracking ability to search an optimal solution which is changing over time. An empirical study was carried out to evaluate the performance of the proposed approach. Experimental results show the suitability of the algorithm in terms of effectiveness to find good solutions for the benchmark problems investigated. Finally, an application is developed, where the proposed algorithm is applied to solve the dynamic economic dispatch problem in power systems.

Abbreviations

BBPSO	Bare bones particle swarm optimization
CMAES	Covariance matrix adaptation evolution strategy
COPs	Constrained optimization problems
DCOPs	Dynamic constrained optimization problems
DCTC	Dynamic constrained TCell
DMOOPs	Dynamic multi-objective optimization problems
EBBPSO-T	Entropy-based BBPSO
ES	Evolution strategies
FIPS	Fully informed particle swarm
GA	Genetic algorithm
Gbest	Global best
GBBPSOwJ	Generalized BBPSO with jumps
GSA	Gravitational search algorithm
G24	G24 benchmark set of DCOPs
HC	Hill-climbing
Lbest	Local best
ML	Maximum likelihood
MCDC	Multi-criteria decision making
MOOPs	Multi-objective optimization problems
MSE	Mean square error
pdf	Probability density function
PSO	Particle swarm optimization
SA	Simulated annealing
SI	Swarm intelligence
SMN	Scale mixtures of normal distributions
SMABBPSO	Scale matrix adaptation BBPSO
SUOPs	Static unconstrained optimization problems
TOPSIS	Technique for order of preference by similarity to ideal solution
UOPs	Unconstrained optimization problems

Notation

$\mathbb{R}$	Set of real numbers
$\mathbb{R}^D$	D-dimensional real Euclidean space, where $\mathbb{R}^D = \mathbb{R} \times \cdots \times \mathbb{R}$
T	Set of discrete times
t	Discrete time
$\mathbb{S}$	Search space, where $\mathbb{S} \subset \mathbb{R}^D$
$\mathbb{F}$	Feasible space, where $\mathbb{F} \subset \mathbb{S}$
$\mathbb{F}(t)$	Feasible space, where $\mathbb{F}(t) \subset \mathbb{S}$ for all $t \in T$
$\mathbf{x}$	Vector of decision variables, where $\mathbf{x} = (x_1, \ldots, x_D)' \in \mathbb{R}^D$
$N(\mathbf{x}, r)$	Neighborhood of center $\mathbf{x} \in \mathbb{R}^D$ and radius $r > 0$
L_d	Lower bound of x_d
U_d	Upper bound of x_d
$[\mathbf{U}, \mathbf{L}]$	Set defined as $[L_1, U_1] \times \cdots \times [L_D, U_D]$
$f(\mathbf{x}), f(\mathbf{x}, t)$	Objective function
$g_i(\mathbf{x}), g_i(\mathbf{x}, t)$	Constraint function
$h_i(\mathbf{x}), h_i(\mathbf{x}, t)$	Constraint function
$L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda})$	Lagrangian function
$L(\mathbf{x}, t, \boldsymbol{\mu}, \boldsymbol{\lambda})$	Lagrangian function
$\bar{\mathbf{x}}$	Feasible optimal solution
$\mathbf{x}^*$	Best solution found by an optimization algorithm
$\bar{f} = f(\bar{\mathbf{x}})$	Objective value of $\bar{\mathbf{x}}$
$f^* = f(\mathbf{x}^*)$	Objective value of $\mathbf{x}^*$
$\bar{f} = f(\bar{\mathbf{x}}, t)$	Objective value of $\bar{\mathbf{x}}$ at time t
$f^* = f(\mathbf{x}^*, t)$	Objective value of $\mathbf{x}^*$ at time t
$s(\mathbf{x}, t)$	Degree of constraint violation of $\mathbf{x}$ at time t
$u(\mathbf{x}, t)$	Number of constraints violated by $\mathbf{x}$ at time t
κ	Parameter of the G24 set that determines the severity of objective function changes
S	Parameter of the G24 set that determines the severity of constraint changes
Δ	Window where the dynamic problem remains constant
δ, C_1, C_2, R_a	Other numerical parameters of the G24 set
$\tau = 0, 1, 2, 3, \ldots$	Iteration counter
$\mathbf{P}$	Swarm of particles
K	Number of particles in the swarm
$\mathcal{N}$	Neighborhood system
$\mathcal{N}_k$	Neighborhood of a particle, where $k = 1, \ldots, K$
$\mathbf{x}_k$	Position of a particle
$\mathbf{v}_k$	Velocity of a particle
$\mathbf{p}_k$	Personal-best position of a particle (pbest)
$\mathbf{n}_k$	Local-best position of a particle (lbest)
$\mathbf{g}$	Global-best position (gbest)
$\mathrm{BEST}(\mathbf{x}_1, \mathbf{x}_2, \ldots)$	Best solution with respect to objective f
$x, y, \ldots$	Random variables
$x \sim p(x)$	Density of x
$x, y \sim p(x, y)$	Joint density of x and y
$x \mid y \sim p(x \mid y)$	Conditional density of x given y

$\mathrm{E}(x)$	Mean of x
$\mathrm{Var}(x)$	Variance of x
$\mathrm{Cov}(x,y)$	Covariance of x and y
$\mathrm{Unif}(x\|a,b)$	Uniform distribution in (a,b)
$Ga(x\|a,b)$	Gamma distribution
$N(x\|\mu,\sigma^2)$	Normal distribution
$t(x\|\nu,\mu,\sigma^2)$	t-distribution
$r_{1,2} \sim \mathrm{Unif}(0,1)$	Random numbers in $(0,1)$
$z \sim N(0,1)$	Standard normal distribution
$\mathbf{x},\mathbf{y},\ldots$	Random vectors
$\mathbf{x} \sim p(\mathbf{x})$	Density of $\mathbf{x}$
$\boldsymbol{\mu} = \mathrm{E}(\mathbf{x})$	Mean vector of $\mathbf{x}$
$\boldsymbol{\Sigma} = \mathrm{Cov}(\mathbf{x})$	Covariance matrix of $\mathbf{x}$
$\mathbf{N}(\mathbf{x}\|\boldsymbol{\mu},\boldsymbol{\Sigma})$	Multivariate normal distribution
$\mathbf{t}(\mathbf{x}\|\nu,\boldsymbol{\mu},\boldsymbol{\Sigma})$	Multivariate t-distribution
$\boldsymbol{\pi} = (\pi_1,\ldots,\pi_M)$	Discrete distribution with M states
$\mathscr{D}_q(\boldsymbol{\pi})$	Measure of diversity of $\boldsymbol{\pi}$
$\mathscr{H}(\boldsymbol{\pi})$	Shannon's entropy of $\boldsymbol{\pi}$
$\mathscr{I}(\boldsymbol{\pi})$	Information of $\boldsymbol{\pi}$, where $\mathscr{I}(\boldsymbol{\pi}) = 1 - \mathscr{H}(\boldsymbol{\pi})$
$\boldsymbol{\theta}$	Parameter (can be a scalar, vector, or matrix)
$L(\boldsymbol{\theta}\|\mathbf{x})$	Likelihood function of $\boldsymbol{\theta}$ for fixed $\mathbf{x}$
$\hat{\boldsymbol{\theta}}$	Estimator of $\boldsymbol{\theta}$
$\hat{\mathscr{H}}$	Shannon's entropy estimator (Shannon's index of diversity)
$\hat{\mathscr{W}}$	Information estimator, where $\hat{\mathscr{W}} = 1 - \hat{\mathscr{H}}$
$A_1,\ldots,A_m$	Feasible alternatives
$C_1,\ldots,C_n$	Evaluation criterion
w_j	Weight of C_j
x_{ij}	Performance rating of A_i under C_j
$\mathbf{D},\mathbf{D}_n,\mathbf{D}_w$	decision matrices
$\mathbf{p}^+$	Positive ideal solution
$\mathbf{p}^-$	Negative ideal solution
$\nabla_{\mathbf{x}} f(\bar{\mathbf{x}})$	Gradient of f
$\nabla^2_{\mathbf{x}} f(\bar{\mathbf{x}})$	Hessian of f
$\nabla_{\mathbf{x}} \mathbf{g}(\bar{\mathbf{x}})$	Jacobian of $\mathbf{g}$

List of Figures

List of Tables

Contents

Part I

Introduction

Chapter 1

Contextualization and Objectives

1.1 Overview of optimization

Many problems of operations research, data analysis, and engineering may be formulated as optimization problems. Managers aim to maximize efficiency in the operation of their production processes. Data analysts aim to maximize or minimize decision criteria to build data-driven models for different random processes. Investors seek to create portfolios that avoid excessive risk while achieving a high rate of return. Many other examples can be mentioned, where it is necessary to identify the most desirable solution to be applied in a given situation to produce the best possible response. Optimization problems motivate and generate the development of a scientific field known as optimization theory.

To precisely capture the concept of optimization, the concept of system is introduced. A system is a collection of interdependent elements working together with the purpose of processing an input $\mathbf{x} = (x_1, \ldots, x_D)'$ to produce a response (or an output) y[1]. The problem of identifying the best feasible input to produce the best possible response for a given system is called optimization[2].

The solution to an optimization problem is usually obtained through a process that occurs in stages. The first stage of this process is known as modelling, which is the construction of a model for the system under study. This means translating the original system to a mathematical structure that can be handled through analytical or computational methods. The model is a mathematical representation of the main features of the system that emulate the observable reality associated with it. In optimization, a possible model for a system is initially formulated as a function f relating the input $\mathbf{x} = (x_1, \ldots, x_D)'$ to the response $y = f(\mathbf{x})$. This function is called the objective function and the components of the input are usually called decision variables. Since the decision variables affect the quality of the response of the system, the objective function is built in such a way that the most desirable values for the decision variables correspond to the best possible responses of the system, i.e., the extremal values of the objective function. Thus, the original problem is transformed to an equivalent problem of minimization (or maximization) of a function. The main goal is to find values for the decision variables that minimize (or maximize) the objective function. Often the decision variables are constrained in some way. As a result, a set of constraints establishes the possible values that each decision variable can assume. In summary, modelling is the process of identifying of the decision variables, objective function, and constraints for a given real-world system under study. The problem of minimizing (or maximizing) the objective function subject to constraints on its variables works as the related optimization problem. If the model is too simplistic, it cannot provide useful information on the system. If the model is too complex, it may be very difficult to solve. A good model represents a compromise between complexity and capacity to capture the essential features of the system under study. The main advantage of a well-constructed model is that it offers a means of identifying and modifying the features of a system to produce the best possible response without requiring its actual construction, thus saving time and cost.

The second stage of an optimization process is to find a solution for the model. Once the model has been formulated, a method can be used to find its solution. There is no universal method for this task but rather a collection of analytical and computational methods, which are applied to different classes of optimization problems. The choice of the method that is appropriate for a specific application often falls on the model builder.

[1] A system can also have a feedback mechanism beyond input and output.

[2] Note that the word "optimization" is being used both to identify a scientific field and to identify its main problem.

The last stage of an optimization process occurs after a solution has been obtained. This stage is to check the suitability of the proposed solution. In many cases, it is possible to use expressions known as optimality conditions for checking whether the current set of values for the decision variables is indeed a solution of the problem. The model may be improved by applying techniques such as sensitivity analysis, which reveal the sensitivity of a solution to changes in the model. Interpretation of the solution in terms of the application may also suggest ways in which the model can be improved. Finally, if any change is made in the model, a new optimization procedure is again applied following what has been described here.

The following notation will be used to describe the general form of an optimization problem. The discussion will be restricted to problems with continuous decision variables:

Decision variables. The vector $\mathbf{x} = (x_1, \ldots, x_D)' \in \mathbb{R}^D$ represents a container for D decision variables that affect the value of the objective function.

Objective. The function f represents the objective function that is a real-valued function of $\mathbf{x}$ to be minimized (or maximized) and $f(\mathbf{x})$ is the objective function value of a solution $\mathbf{x}$. Sometimes, f is also called of cost function or fitness function and, consequently, $f(\mathbf{x})$ is called cost of $\mathbf{x}$ or fitness of $\mathbf{x}$.

Constraints. The numbers U_d and L_d are respectively the upper and lower bounds of the dth component of $\mathbf{x}$. These constraints are boundary constraints that define the domain of values for each decision variable and these bounds define the search space of the problem as

$$\mathbb{S} = [L_1, U_1] \times \cdots \times [L_D, U_D] = [\mathbf{L}, \mathbf{U}] \subset \mathbb{R}^D. \tag{1.1}$$

However, constraints can be more complex. In this case, g_i and h_j are real-valued functions of $\mathbf{x}$ that define certain equations of inequality ($g_i(\mathbf{x}) \leq 0$ for $i = 1, \ldots, I$) and equality ($h_j(\mathbf{x}) = 0$ for $j = 1, \ldots, J$) that a candidate solution $\mathbf{x}$ must satisfy. That is, g_i and h_j are constraint functions and $g_i(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$ are constraints that further restrict the values that can be assigned to $\mathbf{x}$. The set

$$\mathbb{F} = \{\mathbf{x} \in \mathbb{S} : g_1(\mathbf{x}) \leq 0, \ldots, g_I(\mathbf{x}) \leq 0, h_1(\mathbf{x}) = 0, \ldots, h_J(\mathbf{x}) = 0\} \tag{1.2}$$

is called the feasible space and a point $\mathbf{x} \in \mathbb{S}$ that satisfies all the constraints (i.e., a point $\mathbf{x} \in \mathbb{F} \subset \mathbb{S}$) is called a feasible solution to the problem.

Optimal solution. A point $\bar{\mathbf{x}} \in \mathbb{F}$ is a global minimum of f on $\mathbb{F}$ if $f(\bar{\mathbf{x}}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{F}$. If $\bar{\mathbf{x}}$ is a global minimum of f on $\mathbb{F}$, then $\bar{\mathbf{x}}$ is also called a minimizer of f on $\mathbb{F}$. A point $\bar{\mathbf{x}} \in \mathbb{F}$ is a local minimum of f on $\mathbb{F}$ if $f(\bar{\mathbf{x}}) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in neighborhood[3] of $\bar{\mathbf{x}}$. Every global minimum is also a local minimum, but the opposite is not true.

Using this notation, a constrained optimization problem (COP) is usually written as follows:

$$\begin{array}{rll} \min & f(\mathbf{x}) & \text{where } \mathbf{x} = (x_1, \ldots, x_D)' \in \mathbb{R}^D \\ \text{subject to} & g_i(\mathbf{x}) \leq 0 & i = 1, \ldots, I \\ & h_j(\mathbf{x}) = 0 & j = 1, \ldots, J \\ & L_d \leq x_d \leq U_d & d = 1, \ldots, D. \end{array} \tag{1.3}$$

A COP is solved when a global minimum of f on $\mathbb{F}$ is found (when such solution exist). If there is a global minimum $\bar{\mathbf{x}}$ for the problem, then $\bar{f} = f(\bar{\mathbf{x}}) = \min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{F}\}$ is the global minimum value of f on $\mathbb{F}$ and that value is unique. However, note that the global minimum value may be reached by many global minimizers. In practice, it may be difficult to find a global minimizer, then a good alternative solution can be when a local minimizer is found. In addition, it is important to emphasize that a point $\bar{\mathbf{x}}$ is a maximum of a function f_2 on $\mathbb{F}$ if and only if $\bar{\mathbf{x}}$ is a minimum of f_1 on $\mathbb{F}$, where f_1 is a function whose value at any point $\mathbf{x} \in \mathbb{F}$ is given by $f_1(\mathbf{x}) = -f_2(\mathbf{x})$. This means that a maximization problem can be converted into a minimization problem by multiplying the objective function by -1.

An unconstrained optimization problem (UOP) occurs when $I = J = 0$ and $\mathbb{S} = \mathbb{F}$. Therefore, formally an UOP is written as follows:

$$\begin{array}{rll} \min & f(\mathbf{x}) & \text{where } \mathbf{x} = (x_1, \ldots, x_D)' \in \mathbb{R}^D \\ \text{subject to} & L_d \leq x_d \leq U_d & d = 1, \ldots, D. \end{array} \tag{1.4}$$

Optimization problems can be categorized into different classes based on the properties of the objective function and the search space. Standard classes of problems are discussed as follows:

[3] A neighborhood of center $\bar{\mathbf{x}} \in \mathbb{R}^D$ and radius $r > 0$ is defined by $N(\bar{\mathbf{x}}, r) = \{\mathbf{x} \in \mathbb{R}^D : d(\bar{\mathbf{x}}, \mathbf{x}) < r\}$. For more details, see Appendix A.

Single- versus multi-objective optimization. When the quantity to be optimized is expressed using only one objective function, then the problem is referred to as single-objective optimization. Multi-objective optimization problems specify more than one objective to be optimized simultaneously. In this case, having several objective functions, the concept of optimal solution for the problem changes to the concept of trade-off (or compromise), i.e., a solution in which one must balance the objectives that are in conflict and cannot be satisfied at the same time. In these problems, the main goal is to find a set of trade-offs (or compromises) between the objectives rather than an optimal solution as in single-objective optimization.

Static versus dynamic optimization. When the objective function and the constraints do not vary over time, then the problem is referred to as static optimization. Dynamic optimization problems refer to problems that involve either time-varying objective functions or time-varying constraints. The goal in this case is to track the position of the optimal solution as soon as it moves in the search space as a consequence of changes in objective function or constraints.

Unconstrained versus constrained optimization. When the problem uses only boundary constraints, then the problem is referred to as unconstrained optimization. Constrained optimization problems have additional equality or inequality constraints. The goal in this case is to find an optimal solution satisfying all constraints of the problem, i.e., a feasible optimal solution.

Univariate versus multivariate optimization. When the objective function and the constraints are influenced by only a single decision variable, then the problem is referred to as univariate optimization. Multivariate optimization problems involve more than one decision variable.

Continuous versus discrete optimization. When the decision variables are continuous-valued variables, then the problem is referred to as continuous optimization. Discrete optimization problems involve decision variables that assume values in a countable set, i.e., a set whose elements can be put into a one-to-one correspondence with the natural numbers $1, 2, 3, \ldots$. Note that the size of a countable set does not have to be finite, it can be infinite as well.

It is important to note that these problem classes have intersections. For example, it is possible to have a dynamic constrained optimization problem (DCOP) to be solved that involves only continuous decision variables. This classification is summarized in Table 1.1.

Table 1.1: Optimization problem classification

Optimization				
Objective(s)	Environment	Search Space	Problem	Abbreviation
Single-objective	Static	Unconstrained	Unconstrained optimization problem	UOP
		Constrained	Constrained optimization problem	COP
	Dynamic	Unconstrained	Dynamic UOP	DUOP
		Constrained	Dynamic COP	DCOP
Multi-objective	For more details about this class of problems, see Section C.4 (Abbreviation: MOOP)			
Optimization problem (OP)				
Single-objective optimization problem (SOOP); Multi-objective optimization problem (MOOP)				
Static environment (S); Dynamic environment (D)				
Unconstrained search space (U); Constrained search space (C)				

The difficulty in finding a solution for an optimization problem is heavily dependent on the form and mathematical properties of the objective function and the constraints describing the problem. On the other hand, optimization is a scientific field extremely active. Many development cores are target of study and several methods have been proposed for different classes of optimization problems.

It is possible that a solution can be obtained through an analytical method for a given optimization problem. Appendix A presents a review of the optimization theory including its basic concepts, existence and uniqueness theorems, and optimality conditions that characterize optimal solutions. For a detailed development of this theory, the reader is referred to Luenberger & Ye (2008); Sundaram (1996), and Bazaraa, Sherall & Shetty (1993). Unfortunately, analytical methods have a limited scope of application. In most real-world problems, complex systems are described by complicated multidimensional functions that cannot be easily addressed by such methods. In this

cases, computational methods can be implemented to solve such optimization problems. It is clear that, under this scope, only approximations of optimal solutions can be obtained. Appendix A also presents some algorithms that exploit mathematical properties of the functions describing an optimization problem, such as continuity and differentiability. These approaches use first- and second-order derivatives to achieve approximations of optimal solutions. However, the necessary assumptions for their application are not usually met in practice. As a result, derivative-based methods also have a limited scope of application.

Many real-world optimization problems have characteristics such as existence of discontinuity, lack of analytical representation of the objective function and the constraints, and existence of many optimal solutions (many global minimizers). In addition, some real-world optimization problems also have time-varying objective functions or time-varying constraints, which establish a dynamic environment where the problem must be solved. Given all these circumstances, the applicability and effectiveness of classical algorithms are questionable, thus opening up the opportunity for the development of different optimization algorithms for several classes of problems. Many advances have been achieved in the last decades, especially in the field of nature-inspired metaheuristics for optimization problems. Two categories of metaheuristic algorithms can be highlighted at this point: swarm and evolutionary computation. These terms are used to describe classes of algorithms that lie in the intersection of two scientific fields: optimization and computational intelligence. These metaheuristics are usually adopted in problems where classical optimization methods cannot be applied such as black-box optimization problems or optimization problems in which probably neither the objective function nor the constraints are differentiable.

Evolutionary computation proposes optimization algorithms that are inspired by the process of natural evolution: the fittest members of a population have high chance of surviving to integrate the next generation or high chance of transferring a part of their genetic material to their offspring that probably will be part of the next generation. In general, an evolutionary algorithm starts with a population of candidate solutions for a given optimization problem, which are usually called individuals. This population is manipulated by selection, crossover and mutation operators to generate a new population (or the next generation). Then, the objective function of the problem is evaluated for each individual in the new population and those individuals associated with the best objective function values will be selected with high probability to repeat the process again. Appendix A presents a generic evolutionary algorithm that summarizes the main paradigms within evolutionary computation: genetic algorithms, evolution strategies, and evolutionary programming.

Parallel to the development of evolutionary computation, a new category of optimization algorithms appeared in the mid-90s. Instead of modelling the evolutionary process in microscopic level, this new category aimed to model populations in a macroscopic level, in terms of social structures and collective behaviour. Once again, the nature was offering inspiration and motivation to researchers. Since then, this new category of optimization algorithms has developed under the name of swarm computation[4]. As in evolutionary algorithms, swarm algorithms use a population of candidate solutions during the optimization process. The population is referred to as swarm and its individuals are referred to as particles. Each particle (or potential solution) moves in the search space of the problem obeying dynamic rules to update its position. The particles are capable of interacting with the environment and with other particles, namely those particles in its neighborhood. Each particle searches an optimal solution to a given optimization problem learning from its own past experience and from the experiences of its neighbors. The swarm as a whole explores the search space, first at random, and then, when better solutions are found and communicated, the swarm begins to converge by refining its search until a good enough solution is found.

This section ends with the discussion of two swarm algorithms to find approximations of optimal solutions for optimization problems, namely: particle swarm optimization (PSO) (Kennedy & Eberhart, 1995; Shi & Eberhart, 1998; Clerc & Kennedy, 2002) and bare bones PSO (BBPSO) (Kennedy, 2003). The discussion of these algorithms will be directed for solving UOPs. A detailed review on PSO and its variants for UOPs is provided in Appendix B. Appendix C provides a survey on constraint-handling methods that can be incorporated in PSO and its variants for solving COPs. For more details on swarm computation, the reader is referred to Engelbrecht (2007); Parsopoulos & Vrahatis (2010), and Simon (2013). PSO and BBPSO are described as follows:

[4]Swarm computation is also known as swarm intelligence (SI) in the context of computational intelligence. SI is concerned with the design of intelligent systems with many agents whose collective behaviour is inspired by the behaviour of social insects (ants, bees, and wasps) and other animal societies such as bird flocks or fish schools. Examples of such systems include swarm robotic systems and swarm algorithms for optimization problems. White & Pagurek (1998) define SI as a property of systems of unintelligent agents of limited individual capabilities of exhibiting collectively intelligent behaviour. SI is also a scientific field where systems with many individuals are investigated in order to exploit their collective behaviour to solve complex problems. Generally, this collective behaviour emerges from relatively simple actions and interactions between the individual members of the systems. Even though a single member is a non-sophisticated individual, the system as whole is able to achieve complex tasks in cooperation. This process of self-organization to realize complex tasks is spontaneous and it is not coordinated by a centralized control, but it arises out of decisions in group.

Swarm. Let $\mathbb{S} \subset \mathbb{R}^D$ be the search space of an objective function f to be minimized over $\mathbb{S}$. To tackle this task, consider a swarm $\mathbf{P}$ with K particles. The position of a particle is denoted by a D-dimensional vector $\mathbf{x}_k^\tau = (x_{k1}^\tau, \ldots, x_{kD}^\tau)'$ in $\mathbb{S}$. The index k ($k = 1, \ldots, K$) labels the k-th particle in $\mathbf{P}$ and the index τ ($\tau = 1, 2, \ldots$) represents the iteration counter of the algorithm.

Neighborhood. Each particle has a neighborhood consisting of a set of particles which it can communicate with. The neighborhood of a particle is denoted by $\mathscr{N}_k$ and the neighborhood system $\mathscr{N} = \{\mathscr{N}_k : k = 1, \ldots, K\}$ represents a communication structure often thought of as a social network. There is a number of different schemes to connect the particles. Most implementations use one of two simple sociometric principles. The first, called global topology (also known as global-best model or Gbest model), connects each particle in the population with all others, i.e., $\mathscr{N}_k = \mathbf{P}$ for all k. The second, called local topology (also known as local-best model or Lbest model), creates a neighborhood for each particle comprising generally of the particle itself and L neighbors in the population $\mathbf{P}$, i.e., $\mathscr{N}_k \subset \mathbf{P}$ for all k, with $|\mathscr{N}_k| = L + 1 < K$. Section B.2 (see Appendix B) shows examples of neighborhood systems commonly used in swarm algorithms.

Memories. Each particle keeps the memory of the best solution found by itself during the search process (the personal-best position found up to the current iteration or pbest position). In addition, the particles use the neighborhood system to exchange information between them. As a result, each particle also keeps the memory of the best solution found by any particle in its neighborhood during the search process (the local-best position found up to the current iteration or lbest position). The personal-best and local-best positions are respectively denoted by $\mathbf{p}_k^\tau = (p_{k1}^\tau, \ldots, p_{kD}^\tau)'$ and $\mathbf{n}_k^\tau = (n_{k1}^\tau, \ldots, n_{kD}^\tau)'$. If the global-best model (or Gbest model) is used, then the global-best position found so far (or gbest position) is defined as the best personal-best position. If the local-best model (or Lbest model) is used, then the gbest position is defined as the best local-best position. The global-best position is denoted by $\mathbf{g}^\tau = (g_1^\tau, \ldots, g_D^\tau)'$.

PSO. Clerc & Kennedy (2002) introduced the PSO with constriction factor. This variant of PSO is currently known as standard PSO. The local-best PSO (or Lbest PSO) is initialized with a population of particles with random positions and velocities. On initialization, $\mathbf{p}_k^0 = \mathbf{x}_k^0$ for all k and

$$\mathbf{n}_k^0 = \text{BEST}(\mathbf{p}_l^0 : l \in \mathscr{N}_k) = \arg\min\{f(\mathbf{p}_l^0) : l \in \mathscr{N}_k\} \tag{1.5}$$

for all k. The particles are influenced by their own previous experiences and by the experiences of their neighbors. Thus, the velocity of a particle is updated by

$$v_{kd}^{\tau+1} = \chi[v_{kd}^\tau + c_1(p_{kd}^\tau - x_{kd}^\tau) \cdot r_1 + c_2(n_{kd}^\tau - x_{kd}^\tau) \cdot r_2] \tag{1.6}$$

for all k, d, and $\tau \geq 0$. The change in its position is given by

$$x_{kd}^{\tau+1} = x_{kd}^\tau + v_{kd}^{\tau+1} \tag{1.7}$$

for all k, d, and $\tau \geq 0$. In Eq. (1.6), r_1 and r_2 are random numbers uniformly distributed in the range $(0, 1)$, $\varphi = c_1 + c_2 > 4$, and $\chi = 2/|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|$ is the constriction factor. After updating the position, the personal-best position is given by

$$\mathbf{p}_k^{\tau+1} = \text{BEST}(\mathbf{x}_k^{\tau+1}, \mathbf{p}_k^\tau). \tag{1.8}$$

Finally, the local-best position is given by

$$\mathbf{n}_k^{\tau+1} = \text{BEST}(\mathbf{p}_l^{\tau+1} : l \in \mathscr{N}_k). \tag{1.9}$$

The process is repeated until some stopping criterion is met. At the end of the search process, the Lbest PSO returns the global-best position $\mathbf{g} = \text{BEST}(\mathbf{n}_k : k = 1, \ldots, K)$ and the value of the objective function in $\mathbf{g}$. The parameter φ is commonly set to 4.1, $c_1 = c_2 = 2.05$, and $\chi \approx 0.7298$ (Clerc & Kennedy, 2002). The global-best PSO (or Gbest PSO) updates the velocity of a particle as

$$v_{kd}^{\tau+1} = \chi[v_{kd}^\tau + c_1(p_{kd}^\tau - x_{kd}^\tau) \cdot r_1 + c_2(g_d^\tau - x_{kd}^\tau) \cdot r_2] \tag{1.10}$$

for all k, d, and $\tau \geq 0$, where g_d^τ is the d-th component of the global-best position that is defined as $\mathbf{g}^\tau = \text{BEST}(\mathbf{p}_k^\tau : k = 1, \ldots, K)$ in this case. The essential steps of the PSO are summarized as the pseudo code shown in Algorithm 1.

BBPSO. Kennedy (2003) introduced a variant of PSO named bare bones PSO (BBPSO). Kennedy proposed to change the position of a particle according to a probability distribution rather than to add a velocity in the current position, as is done in PSO. In BBPSO, the position of a particle is updated as

$$x_{kd}^{\tau+1} = \mu_{kd}^{\tau} + \sigma_{kd}^{\tau} \cdot z \tag{1.11}$$

for all k, d, and $\tau \geq 0$, where

- $\mu_{kd}^{\tau} = 0.5(p_{kd}^{\tau} + n_{kd}^{\tau})$ or $\mu_{kd}^{\tau} = 0.5(p_{kd}^{\tau} + g_{d}^{\tau})$
- $\sigma_{kd}^{\tau} = |p_{kd}^{\tau} - n_{kd}^{\tau}|$ or $\sigma_{kd}^{\tau} = |p_{kd}^{\tau} - g_{d}^{\tau}|$
- $z \sim N(0,1)$.

The symbol $\sim$ indicates that z is random variable which has normal distribution with mean 0 and variance 1 (for more details, see Chapter 3). The swarm explores the search space of a given problem by sampling of explicit probabilistic models constructed from the information associated with promising candidate solutions. The search for solutions is the result of a constructive cooperation between particles by using probabilistic models whose parameters are defined in terms of the information that is obtained during the optimization process (pbest, lbest, and gbest positions). The essential steps of the BBPSO are summarized as the pseudo code shown in Algorithm 1.

Algorithm 1 PSO and BBPSO for UOPs.

Input: $D, K, \mathcal{N}, \chi, c_1, c_2$, and f

{Local-best PSO}
$\tau \leftarrow 0$
for $k \in \{1,\ldots,K\}$ **do**
  Initialize $\mathbf{x}_k$ and set $\mathbf{v}_k \approx \mathbf{0}$
  $\mathbf{p}_k \leftarrow \mathbf{x}_k$
end for
for $k \in \{1,\ldots,K\}$ **do**
  $\mathbf{n}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
end for
repeat
  $\tau \leftarrow \tau + 1$
  for $k \in \{1,\ldots,K\}$ **do**
    for $d \in \{1,\ldots,D\}$ **do**
      $r_1, r_2 \sim \text{Unif}(0,1)$
      $v_{kd} \leftarrow \chi[v_{kd} + c_1(p_{kd} - x_{kd}) \cdot r_1 + c_2(n_{kd} - x_{kd}) \cdot r_2]$
      $x_{kd} \leftarrow x_{kd} + v_{kd}$
    end for
    $\mathbf{p}_k \leftarrow \text{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  for $k \in \{1,\ldots,K\}$ **do**
    $\mathbf{n}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
  end for
until some termination condition is met
$\mathbf{g} \leftarrow \text{BEST}(\mathbf{n}_k : k = 1,\ldots,K)$
return $\mathbf{x}^* = \mathbf{g}$ and $f^* = f(\mathbf{g})$.

Input: $D, K, \mathcal{N}$, and f

{Global-best BBPSO}
$\tau \leftarrow 0$
for $k \in \{1,\ldots,K\}$ **do**
  Initialize $\mathbf{x}_k$
  $\mathbf{p}_k \leftarrow \mathbf{x}_k$
end for
$\mathbf{g} \leftarrow \text{BEST}(\mathbf{p}_k : k = 1,\ldots,K)$
repeat
  $\tau \leftarrow \tau + 1$
  for $k \in \{1,\ldots,K\}$ **do**
    for $d \in \{1,\ldots,D\}$ **do**
      $z \sim N(0,1)$
      $\mu_{kd} \leftarrow 0.5(p_{kd} + g_d)$
      $\sigma_{kd} \leftarrow |p_{kd} - g_d|$
      $x_{kd} \leftarrow \mu_{kd} + \sigma_{kd} \cdot z$
    end for
    $\mathbf{p}_k \leftarrow \text{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  $\mathbf{g} \leftarrow \text{BEST}(\mathbf{p}_k : k = 1,\ldots,K)$
until some termination condition is met
return $\mathbf{x}^* = \mathbf{g}$ and $f^* = f(\mathbf{g})$.

The decision to stop a swarm algorithm (PSO, BBPSO, and variants) can depend on several criteria related to the problem information, resources, or the ability of the algorithm to obtain a good approximation of a global optimum. Section A.7 (see Appendix A) describes some criteria that can also be used in swarm algorithms.

1.2 Dynamic constrained optimization

Dynamic constrained optimization problems (DCOPs) form a class of problems where the objective function or the constraints can change over time. In static optimization problems, finding a global optimum is considered as

the main goal. In dynamic optimization problems, the goal is not only to find an optimal solution, but also track its trajectory as closely as possible over time. Changes in the environment of the problem must be taken into account during the optimization process in such way that these problems are to be solved online.

Many real-world applications are dynamic and consistently modelled as DCOPs. Examples of such problems include:

1. economic dispatch of quantities produced by production units, when the total demand varies over time.
2. data analysis, where the contents of the database are continuously updated.
3. investment portfolio evaluation, where the assessment of investment risk varies over time.
4. financial trading models, where market conditions can change abruptly.
5. target recognition, where the sensor performance varies based on environmental conditions.

A prototype of the first problem listed above can be briefly described as follows. Consider a production system composed by n units designed specifically to meet a total demand of quantities produced, which varies over time. The main objective of this problem is to reduce the operational costs of the production system without violating a set of security and efficiency constraints simultaneously. Therefore, a good solution is to find for each time t a minimum cost production schedule, establishing the quantity x_{it} that the unit U_i must produce at time t, for all $i = 1,\ldots,n$, so that the demand D_t is met, taking into account losses during the delivery process and security constraints for each production unit. Figure 1.1 shows a schematic diagram of this problem, which may be stated formally as follows:

$$\begin{array}{rll} \min & \sum_{i=1}^{n} \text{Cost}(x_{it},t) & \text{where } t = 1,2,\ldots,t_{\max} \\ \text{subject to} & \sum_{i=1}^{n} x_{it} = D_t + L_t & \text{for all } t \\ & 0 \leq x_{it} - x_{i(t-1)} \leq A_i & \text{for all } i,t \\ & 0 \leq x_{i(t-1)} - x_{it} \leq B_i & \text{for all } i,t \\ & x_i^{\min} \leq x_{it} \leq x_i^{\max} & \text{for all } i,t \end{array} \tag{1.12}$$

where $A_i, B_i, x_i^{\min}, x_i^{\max}$ are constants representing operational characteristics of the unit U_i.

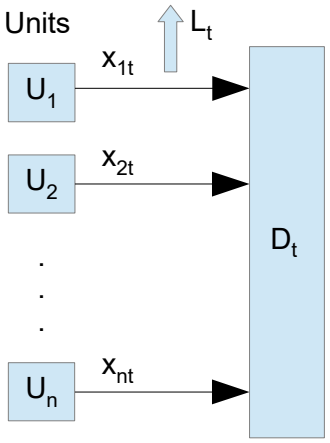

Figure 1.1: Economic dispatch of quantities produced by a system composed by n units.

In general, a DCOP can be stated as follows (Liu, 2008; Nguyen & Yao, 2009; Nguyen, 2010; Nguyen & Yao, 2012; Aragón, Esquivel & Coello Coello, 2013; Pal et al., 2013; Fu et al., 2014; Campos & Krohling, 2014, 2016; Mavrovouniotis, Li & Yang, 2017):

$$\begin{array}{rll} \min & f(\mathbf{x},t) & \text{where } \mathbf{x} = (x_1,\ldots,x_D)' \in \mathbb{R}^D \text{ and } t \in T = \{0,1,2,\ldots,t_{\max}\} \\ \text{subject to} & g_i(\mathbf{x},t) \leq 0 & i = 1,\ldots,I \\ & L_d \leq x_d \leq U_d & d = 1,\ldots,D. \end{array} \tag{1.13}$$

The decision variables contained in $\mathbf{x}$ are bounded by upper and lower bounds that define the search space $\mathbb{S}$ of the problem and the variable t represents the discrete time. The function $f(\mathbf{x},t)$ represents an objective function to be minimized and $g_i(\mathbf{x},t) \leq 0$ are I inequality constraints. The functions f and g_i are linear or nonlinear real-valued functions. The inequality constraints define the feasible space $\mathbb{F}(t)$, such that $\mathbb{F}(t) \subset \mathbb{S}$ for all $t \in T$. For each value of $t \in T$, solutions in $\mathbb{F}(t)$ are called of feasible solutions. A specific environment of a DCOP is defined by its full configuration:

$$\langle D, \mathbf{x} \in \mathbb{R}^D, t, f(\mathbf{x},t), g_i(\mathbf{x},t), L_d, U_d : i = 1,\ldots,I \text{ and } d = 1,\ldots,D \rangle. \tag{1.14}$$

For a fixed value of t, if there exists a solution $\bar{\mathbf{x}} \in \mathbb{F}(t)$ such that $f(\bar{\mathbf{x}},t) \leq f(\mathbf{x},t)$ for all $\mathbf{x} \in \mathbb{F}(t)$, then $\bar{\mathbf{x}}$ is called a feasible optimal solution and $f(\bar{\mathbf{x}},t)$ is the optimal objective value of $\bar{\mathbf{x}}$ at time t. The main goal is to find a feasible optimal solution for all $t \in T$. The reader is referred to Chapter 2 for a more detailed discussion on dynamic constrained optimization.

DCOPs are clearly difficult problems. To deal with such optimization problems, algorithms must be able to tackle various challenges throughout the optimization process, such as: detecting changes in the environment, responding efficiently to the changed environment, and updating memories that are implemented to keep promising solutions. In addition, algorithms must be capable of maintaining a good balance between exploration (diversification) and exploitation (intensification). Too much stress on exploration would result in pure global search and slow convergence speed. On the other hand, too much stress on exploitation would result in pure local search and fast loss of tracking ability (traceability) to search an optimal solution which is changing over time. As a consequence, algorithms for DCOPs must contain several mechanisms to adequately address such problems, namely:

1. mechanism to detect changes in the environment.
2. mechanism to monitor the population diversity.
3. mechanism to maintain the proper balance between exploration and exploitation.
4. mechanism to deal with constraints and its dynamics.
5. mechanism to update memories keeping promising solutions.

1.3 Objectives

Swarm algorithms are traditionally designed to deal with optimization problems in static environments. However, although swarm algorithms have shown potential to solve static problems, little attention has been paid to adapt these algorithms to deal with dynamic optimization problems, which surely represents an important research gap to be explored in swarm computation. The main objective of this thesis is to understand the characteristics of DCOPs and to propose a new algorithmic methodology based on swarm computation for solving DCOPs. Formalized the new algorithm, its performance should be compared to the performance of other approaches already established as state-of-the-art in the field of dynamic constrained optimization. In addition, through experimental studies, the main advantages and disadvantages of the new approach should be identified.

1.4 Contributions

This thesis proposes an entropy-based BBPSO (EBBPSO-T for short) for solving DCOPs. BBPSO has shown potential to solve static optimization problems, but until recently it has not been adapted to deal with DCOPs. In EBBPSO-T, the Shannon's entropy (Shannon, 1948) is established as a phenotypic diversity index and the proposed algorithm uses the Shannon's index of diversity to aggregate the global-best and local-best BBPSO variants. The proposed approach applies the idea of mixture of search directions, using the index of diversity as a factor to balance the influence of the global-best and local-best search directions. High diversity promotes the search guided by the global-best solution, with a normal distribution for exploitation. Low diversity promotes the search guided by the local-best solution, with a heavy-tailed distribution for exploration (Andrews & Mallows, 1974; Choy & Chan, 2008). This dynamic rule works as a mechanism to maintain and introduce diversity during the search process. A constraint-handling strategy is also proposed. It treats the objective function and the constraints separately using a ranking method with selection based on technique for order of preference by similarity to ideal solution (TOPSIS) (Hwang & Yoon, 1981; Behzadian et al., 2012) to obtain the best solution within a specific population of

candidate solutions (priv. comm., Krohling, 2012-2017). This constraint-handling strategy balances the objective function against the degree of constraint violation in such a way that neither of them is dominant. Mechanisms to detect changes in the environment and to update the particles' memories are also implemented into the proposed algorithm. The mechanism to detect changes in the environment is based on a fixed set of detectors uniformly distributed in the search space (Richter, 2009; Richter & Dietel, 2010). The population of detectors differs fully from particles that constitute the swarm, which means that this mechanism is not based on performance drop. When a change in the environment is detected, a random-immigrants scheme acts to introduce diversity. Part of the swarm is replaced with randomly generated particles and this strategy acts only on the first iteration of the changed environment. Re-evaluation of fitness values and the ranking method combined with TOPSIS are also used to update the particles' memories. In summary, EBBPSO-T is endowed with mechanisms to maintain and introduce diversity, handling constraints, detect changes in the environment, and update memories. It is important to emphasize that these mechanisms do not act independently. They operate related to each other to tackle problems such as: loss of diversity, outdated memories, and loss of tracking ability.

The main contribution of this thesis is to provide an adaptation of the BBPSO to deal with DCOPs using the Shannon's entropy as a decisive factor to maintain the proper balance between diversification and intensification at any stage of the search process without loss of traceability to search an optimal solution which is changing over time. The use of entropy is motivated by the relationship between entropy and population diversity. Experimental results show the suitability of the new algorithm in terms of effectiveness to find good solutions for most of the benchmark problems investigated. In addition, an application is developed, where EBBPSO-T is applied to solve the dynamic economic dispatch problem (Abouheaf, Lee & Lewis, 2013), which is one of major problems in power system operation.

1.5 Thesis structure

The remainder of this thesis is organized as follows:

Chapter 2 presents the background information on DCOPs. The problem is formally defined, its essential characteristics are described, and optimality conditions are discussed. A literature review is presented, focusing on recent works on swarm and evolutionary computation for dynamic optimization problems. A set of benchmark problems is presented and the advantages and limitations of this set of problems are critically discussed. Finally, algorithmic attributes to deal with DCOPs are discussed along with performance measures used in empirical studies designed to evaluate the performance of algorithms for solving DCOPs.

Chapter 3 compiles some definitions, concepts, and methods that are important for this thesis. It serves as a quick reference for several topics as: random variables, entropy of a random variable, entropy estimation procedures, relationship between diversity and entropy, and an exploratory method to rank decision alternatives based on multi-criteria. These concepts and methods are essential to understand the subsequent development of the proposed algorithm in this thesis for solving DCOPs.

Chapter 4 describes a new algorithmic methodology for solving DCOPs. The proposed algorithm, named as EBBPSO-T, explores the principles of swarm computation jointly with the concepts and methods described in Chapter 3 to design an approach for dealing with DCOPs. EBBPSO-T is endowed with different mechanisms that are described in the context of dynamic optimization.

Chapter 5 presents an experimental analysis conducted to investigate the performance of EBBPSO-T when solving DCOPs. Results of experiments designed to evaluate the performance of the proposed algorithm are reported and discussed along with experimental results obtained by other algorithms for comparison purposes. An application is also developed, where EBBPSO-T is applied to solve the dynamic economic dispatch problem in power systems.

Chapter 6 ends the work with conclusions and directions for future research.

Appendix A reviews the fundamentals of optimization theory. Basic concepts of theory, existence and uniqueness of solutions, optimality conditions, and some computational methodologies are presented and discussed for unconstrained and constrained optimization. The review aims to provide concepts and results related to the main topic of research of this thesis.

Appendix B presents a short review about swarm algorithms for solving UOPs. Basic concepts are discussed, including neighborhood systems for exchanging information between particles. A convergence analysis of the standard PSO is presented along with guidelines for parameter selection. The review aims to provide concepts and results related to the main topic of research of this thesis.

Appendix C provides a survey on constraint handling methods that have been adopted over the years to deal with optimization problems in constrained search spaces. Special attention is given on how these methods can be incorporated in swarm algorithms to solve COPs.

Appendix D summaries, for comparison purposes, several experimental results obtained by other algorithms when their performances were tested on the benchmark problems of the G24 set of DCOPs.

Part of the research work presented in this thesis has been published in the following works:

1. M. Campos and R. A. Krohling. *Entropy-based bare bones particle swarm for dynamic constrained optimization*. **Knowledge-Based Systems**, v. 97, pp. 203-223, 2016.
   http://dx.doi.org/10.1016/j.knosys.2015.12.017

2. R. A. Krohling, R. Lourenzutti, and M. Campos. *Ranking and comparing evolutionary algorithms with Hellinger-TOPSIS*. **Applied Soft Computing**, v. 37, pp. 217-226, 2015.
   http://dx.doi.org/10.1016/j.asoc.2015.08.012

3. M. Campos, R. A. Krohling, and I. Enriquez. *Bare bones particle swarm optimization with scale matrix adaptation*. **IEEE Transactions on Cybernetics**, v. 44(9), pp. 1567-1578, 2014.
   http://dx.doi.org/10.1109/TCYB.2013.2290223

4. M. Campos and R. A. Krohling. *Bare bones particle swarm with scale mixtures of Gaussians for dynamic constrained optimization problems*. In **Proceedings of IEEE Congress on Evolutionary Computation**, pp. 202-209, 2014.
   http://dx.doi.org/10.1109/CEC.2014.6900256

5. M. Campos and R. A. Krohling. *Hierarchical bare bones particle swarm for solving constrained optimization problems*. In **Proceedings of IEEE Congress on Evolutionary Computation**, pp. 805-812, 2013.
   http://dx.doi.org/10.1109/CEC.2013.6557651

Part II

Problem Definition and Methodology

Chapter 2

Dynamic Constrained Optimization

This chapter presents the background information on dynamic constrained optimization problems (DCOPs). The problem is formally defined, its essential characteristics are described, and optimality conditions are discussed. A literature review is presented, focusing on recent works on swarm and evolutionary computation for dynamic optimization problems. A set of benchmark problems is presented and the advantages and limitations of this set of problems are critically discussed. Finally, algorithmic attributes to deal with DCOPs are discussed along with performance measures used in empirical studies designed to evaluate the performance of algorithms for solving DCOPs.

2.1 Problem definition

Without loss of generality, the general form of a dynamic constrained optimization problem (DCOP) is usually written as follows (Liu, 2008; Nguyen & Yao, 2009; Nguyen, 2010; Nguyen & Yao, 2012; Aragón, Esquivel & Coello Coello, 2013; Pal et al., 2013; Fu et al., 2014; Campos & Krohling, 2014, 2016; Mavrovouniotis, Li & Yang, 2017):

$$\begin{array}{rll} \min & f(\mathbf{x},t) & \text{where } \mathbf{x} = (x_1,\ldots,x_D)' \in \mathbb{R}^D \text{ and } t \in T = \{0,1,2,\ldots,t_{\max}\} \\ \text{subject to} & g_i(\mathbf{x},t) \leq 0 & i = 1,\ldots,I \\ & L_d \leq x_d \leq U_d & d = 1,\ldots,D. \end{array} \tag{2.1}$$

The vector $\mathbf{x}$ represents a container for the decision variables and t represents the discrete time. The decision variables are bounded by upper and lower bounds that define the search space $\mathbb{S}$ of the problem, i.e., $\mathbb{S}$ is written as

$$\mathbb{S} = [L_1,U_1] \times \cdots \times [L_D,U_D] = [\mathbf{L},\mathbf{U}]. \tag{2.2}$$

The function $f(\mathbf{x},t)$ is the objective function to be minimized and $g_i(\mathbf{x},t) \leq 0$ are I inequality constraints. The functions f and g_i are linear or nonlinear real-valued functions. The inequality constraints define the feasible space $\mathbb{F}(t)$ of the problem, such that $\mathbb{F}(t) \subseteq \mathbb{S}$ for all $t \in T$. Formally, the feasible space is written as

$$\mathbb{F}(t) = \{\mathbf{x} \in \mathbb{S} : g_1(\mathbf{x},t) \leq 0,\ldots,g_I(\mathbf{x},t) \leq 0\} \qquad \text{for all } t \in T \tag{2.3}$$

and solutions in $\mathbb{F}(t)$ are called feasible solutions.

In words, a DCOP can be stated as a sequence of constrained optimization problems indexed by time, which are to be solved on-line by an optimization algorithm as time goes by. For each value of t, a specific environment of a given problem is defined by its full configuration:

$$\langle D,\mathbf{x} \in \mathbb{R}^D,t,f(\mathbf{x},t),g_i(\mathbf{x},t),L_d,U_d : i = 1,\ldots,I \text{ and } d = 1,\ldots,D\rangle. \tag{2.4}$$

For each value of t, if there exists a solution $\bar{\mathbf{x}} \in \mathbb{F}(t)$ such that $f(\bar{\mathbf{x}},t) \leq f(\mathbf{x},t)$ for all $\mathbf{x} \in \mathbb{F}(t)$, then $\bar{\mathbf{x}}$ is a feasible optimal solution (or a global optimum) and $f(\bar{\mathbf{x}},t)$ is the feasible optimal value of $\bar{\mathbf{x}}$ when the time is equal to t (or the global optimum value of f on $\mathbb{F}(t)$). The main goal is to find a feasible optimal solution for all $t \in T$. A DCOP is solved optimally if and only if each of the constrained problems is solved optimally. This formulation is

certainly true for most of the real-world optimization problems in dynamic environments. An alternative notation to define a DCOP is given by:

$$\begin{array}{rl} \min & f(\mathbf{x},t) \text{ where } \mathbf{x} \in \mathbb{R}^D \text{ and } t \in T \\ \text{subject to} & \mathbf{g}(\mathbf{x},t) \leq \mathbf{0} \\ & \mathbf{x} \in \mathbb{S} \subset \mathbb{R}^D. \end{array} \tag{2.5}$$

Note that $f : \mathbb{R}^D \times T \to \mathbb{R}$ and $\mathbf{g} : \mathbb{R}^D \times T \to \mathbb{R}^I$.

2.2 Essential characteristics of dynamic optimization problems

In DCOPs, the objective function and the constraints can be combined in three different ways:

1. both the objective function and the constraints are dynamic.
2. the objective function is dynamic and the constraints are static.
3. the objective function is static and the constraints are dynamic.

If the objective function changes over time, this can affect the location of the global optimum. For example, the global optimum can move from one disconnected feasible subregion of the search space to another one. If the constraints are dynamic, this can affect the structure of the feasible region. For example, the size of the feasible space, its shape, and possibly the number of disconnected feasible subregions can change over time. In problems with a static objective function and dynamic constraints, a change in the feasible region can expose a new global optimum without changing the value of the old optimum. In addition, DCOPs might also have the common characteristics of constrained problems such as: global optima in the boundaries of feasible regions or global optima in the search boundary. Regardless of the scenario to be considered, DCOPs are clearly difficult problems.

2.3 Optimality conditions

Consider a DCOP such as stated in Eq. (2.1). The Lagrangian function $L : \mathbb{R}^D \times T \times \mathbb{R}^I \to \mathbb{R}$ associated with this problem is given by:

$$L(\mathbf{x},t,\boldsymbol{\mu}) = f(\mathbf{x},t) + \boldsymbol{\mu}'\mathbf{g}(\mathbf{x},t) = f(\mathbf{x},t) + \sum_{i=1}^{I} \mu_i g_i(\mathbf{x},t). \tag{2.6}$$

Observe that

$$\nabla_{\mathbf{x}} L(\mathbf{x},t,\boldsymbol{\mu}) = \nabla_{\mathbf{x}} f(\mathbf{x},t) + \boldsymbol{\mu}'\nabla_{\mathbf{x}}\mathbf{g}(\mathbf{x},t) = \nabla_{\mathbf{x}} f(\mathbf{x},t) + \sum_{i=1}^{I} \mu_i \nabla_{\mathbf{x}} g_i(\mathbf{x},t) \tag{2.7}$$

$$\nabla_{\boldsymbol{\mu}} L(\mathbf{x},t,\boldsymbol{\mu}) = \mathbf{g}(\mathbf{x},t) \tag{2.8}$$

$$\nabla^2_{\mathbf{x}} L(\mathbf{x},t,\boldsymbol{\mu}) = \nabla^2_{\mathbf{x}} f(\mathbf{x},t) + \boldsymbol{\mu}'\nabla^2_{\mathbf{x}}\mathbf{g}(\mathbf{x},t) = \nabla^2_{\mathbf{x}} f(\mathbf{x},t) + \sum_{i=1}^{I} \mu_i \nabla^2_{\mathbf{x}} g_i(\mathbf{x},t). \tag{2.9}$$

If there is a feasible point $\tilde{\mathbf{x}}$ such that $g_i(\tilde{\mathbf{x}},t) = 0$ for some $i = 1,\ldots,I$ and $t \in T$, then g_i is said to be active at $\tilde{\mathbf{x}}$ when the time is t. A feasible point $\tilde{\mathbf{x}}$ is said to be a regular point of the constraints of the problem if $\nabla_{\mathbf{x}} g_k(\tilde{\mathbf{x}},t)$ for $k \in A_{\tilde{\mathbf{x}}} = \{k : g_k(\tilde{\mathbf{x}},t) = 0\}$ are linearly independent vectors. Optimality conditions for a DCOP are given by Results 2.1, 2.2, 2.3, and 2.4.

Result 2.1 (First-order necessary conditions)**.** *Consider a DCOP such as stated in Eq. (2.1). Suppose that $f, \mathbf{g} \in C^1$ (with respect to the decision variables of the problem) and let $\bar{\mathbf{x}}$ be a regular point of the constraints. If $\bar{\mathbf{x}}$ is a local minimum of f on $\mathbb{F}(t)$, then there is a vector $\bar{\boldsymbol{\mu}}$ such that*

$$\nabla_{\mathbf{x}} L(\bar{\mathbf{x}},t,\bar{\boldsymbol{\mu}}) = \mathbf{0} \tag{2.10}$$

$$\bar{\boldsymbol{\mu}}'\mathbf{g}(\bar{\mathbf{x}},t) = 0 \tag{2.11}$$

$$\bar{\boldsymbol{\mu}} \geq \mathbf{0}. \tag{2.12}$$

Eqs. (2.10), (2.11), and (2.12) represent the well-known Karush-Kuhn-Tucker conditions (see Appendix A).

For the next result, consider the following notation. Let $\bar{\mathbf{x}}$ be a local minimum of f on $\mathbb{F}(t)$ and let $A_{\bar{\mathbf{x}}} = \{k : g_k(\bar{\mathbf{x}},t) = 0\}$. In addition, denote $A^+_{\bar{\mathbf{x}}} = \{k \in A_{\bar{\mathbf{x}}} : \bar{\mu}_k > 0\}$ and $A^0_{\bar{\mathbf{x}}} = \{k \in A_{\bar{\mathbf{x}}} : \bar{\mu}_k = 0\}$.

Result 2.2 (Second-order necessary conditions)**.** *Consider a DCOP such as stated in Eq. (2.1). Suppose that $f, \mathbf{g} \in C^2$ (with respect to the decision variables of the problem) and let $\bar{\mathbf{x}}$ be a regular point of the constraints. If $\bar{\mathbf{x}}$ is a local minimum of f on $\mathbb{F}(t)$, then there is a vector $\bar{\boldsymbol{\mu}}$ such that the KKT conditions are satisfied by $(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}})$ and $\mathbf{y}'\nabla^2_{\mathbf{x}} L(\bar{\mathbf{x}},t,\bar{\boldsymbol{\mu}})\mathbf{y} \geq 0$ for all $\mathbf{y} \in Y$, where $Y = \{\mathbf{y} : \nabla_{\mathbf{x}} g_{k\in A^+_{\bar{\mathbf{x}}}}(\bar{\mathbf{x}},t)'\mathbf{y} = 0, \nabla_{\mathbf{x}} g_{k\in A^0_{\bar{\mathbf{x}}}}(\bar{\mathbf{x}},t)'\mathbf{y} \leq 0\}$.*

Result 2.3 (Second-order sufficient conditions)**.** *Consider a DCOP such as stated in Eq. (2.1). Suppose that $f, \mathbf{g} \in C^2$ (with respect to the decision variables of the problem) and let $\bar{\mathbf{x}}$ be a regular point of the constraints. If there is a vector $\bar{\boldsymbol{\mu}}$ such that*

$$\nabla_{\mathbf{x}} L(\bar{\mathbf{x}},t,\bar{\boldsymbol{\mu}}) = \mathbf{0} \tag{2.13}$$

$$\bar{\boldsymbol{\mu}}'\mathbf{g}(\bar{\mathbf{x}},t) = 0 \tag{2.14}$$

$$\bar{\boldsymbol{\mu}} \geq \mathbf{0} \tag{2.15}$$

and

$$\mathbf{y}'\nabla^2_{\mathbf{x}} L(\bar{\mathbf{x}},t,\bar{\boldsymbol{\mu}})\mathbf{y} > 0 \tag{2.16}$$

for all $\mathbf{y} \in Y - \{\mathbf{0}\}$, then $\bar{\mathbf{x}}$ is a strict local minimum of f on $\mathbb{F}(t)$.

Result 2.4 (First-order condition)**.** *Consider a DCOP such as stated in Eq. (2.1). Assume that $\mathbb{S}$ is a convex set and $f, \mathbf{g}$ are convex functions on $\mathbb{S}$. Assume also that $f, \mathbf{g} \in C^1$ (with respect to the decision variables of the problem) and let $\bar{\mathbf{x}}$ be a regular point of the constraints. Then, there is a vector $\bar{\boldsymbol{\mu}}$ such that the KKT conditions are satisfied by $(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}})$ if and only if $\bar{\mathbf{x}}$ is a global minimum of f on $\mathbb{F}(t)$.*

2.4 Population-based metaheuristics for dynamic optimization

Nguyen, Yang & Branke (2012) reported an in-depth survey of the state-of-the-art of academic research on evolutionary computation to deal with dynamic optimization problems. Richter & Yang (2012) developed a comparative study by discussing analytical and evolutionary approaches for dynamic optimization problems that occur in dynamic programming, optimal control, and evolutionary optimization. Recently, Mavrovouniotis, Li & Yang (2017) presented a broad review on swarm computation for dynamic optimization focused on several classes of dynamic problems, real-world applications, and considerations about future directions in this subject. These studies can be seen as appropriate general introductions to the field of dynamic optimization.

In the literature on dynamic optimization, many algorithms have been designed and tested on unconstrained optimization problems in dynamic environments (Eberhart & Shi, 2001; Morrison, 2004; Jin & Branke, 2005; Blackwell & Branke, 2006; Lung & Dumitrescu, 2007; Du & Lin, 2008; Lung & Dumitrescu, 2010; Liu, Yang & Wang, 2010; Li & Yang, 2012, 2013; Yazdani et al., 2013). Most of these approaches have been tested on the moving peaks benchmark problem proposed by Branke (1999). However, there are few studies about constrained optimization problems in dynamic environments. This gap has been filled with recent studies that focus their attention on DCOPs (Liu, 2008; Nguyen & Yao, 2009; Nguyen, 2010; Nguyen & Yao, 2012; Aragón, Esquivel & Coello Coello, 2013; Pal et al., 2013; Campos & Krohling, 2014, 2016; Mavrovouniotis, Li & Yang, 2017).

Liu (2008) developed one of the first studies in this field and proposed three benchmark problems called DCT 1, 2, and 3, respectively. Liu introduced an approach based on particle swarm optimization (PSO) for DCOPs. The performance of the proposed algorithm by Liu was empirically tested on the DCT benchmark problems. The DCT benchmark problems consider the time variable as the only time-dependent parameter. Therefore, the dynamic is created by simply increasing this variable.

Nguyen & Yao (2009); Nguyen (2010) and Nguyen & Yao (2012) studied the characteristics that might make DCOPs difficult to be solved by some existing dynamic optimization and constraint-handling algorithms. Nguyen and Yao also introduced a set of benchmark problems, named the G24 set, with these characteristics and tested different versions of genetic algorithms (GA) on these problems, including some new variants proposed by the authors themselves. The G24 set of benchmark problems to simulate DCOPs is discussed in Section 2.5 of this chapter. The DCT benchmark problems differ from the G24 benchmark set in the following aspects: (1) the DCT problems only capture linear change, while problems in the G24 set depend on parameters that are time-dependent functions determining how the objective function and the constraint functions change over time; (2) unlike the G24

benchmark set, the DCT problems do not reflect common situations like dynamic objective and fixed constraints, fixed objective and dynamic constraints, and other common properties of DCOPs.

Aragón, Esquivel & Coello Coello (2013) investigated the behaviour of an adaptive immune system for solving DCOPs. The approach proposed by Aragón and co-authors is called dynamic constrained TCell (DCTC) and it is an adaptation of an existing algorithm, which was originally designed to solve static constrained problems. The performance of DCTC was compared with respect to two GA-based approaches tested on the G24 benchmark set.

Pal et al. (2013) introduced a new approach for DCOPs, combining the gravitational search algorithm (GSA) with a modified version of the repair method (GSARepair). GSARepair was also tested on the G24 benchmark set and its performance was compared with respect to several GA-based approaches tested on this benchmark set.

Li & Yang (2013) presented a review of different approaches based on PSO for DOPs. In fact, PSO has been applied as an effective tool to solve many global optimization problems in static environments: there are variants of PSO for unconstrained optimization (Clerc & Kennedy, 2002; Mendes, Kennedy & Neves, 2004; Kennedy & Mendes, 2006; Liang et al., 2006; Richer & Blackwell, 2006; Parsopoulos & Vrahatis, 2007; Mendel, Krohling & Campos, 2011; Krohling, Mendel & Campos, 2011; Campos, Krohling & Enriquez, 2014) and variants of PSO for constrained optimization (Krohling & Coelho, 2006; He & Wang, 2007; Liu, Cai & Wang, 2010; Campos & Krohling, 2013; Jordehi, 2015). However, there are difficulties with applying PSO to solve DOPs. The difficulties lie in two aspects: diversity loss due to convergence and outdated memories due to changes in the environment. The second difficulty can be solved by re-evaluating particles over time. However, it is hard to solve the diversity loss issue due to the difficulty of balancing the exploration and exploitation during the search process. Hence, to address the diversity loss issue, different kinds of approaches have been proposed to enhance the performance of PSO in dynamic environments. As already mentioned, Li & Yang (2013) discussed several approaches based on PSO for dynamic optimization and categorized the main ideas in different groups in terms of their main characteristics: diversity maintaining schemes, multi-population schemes, adaptive schemes, and hybrid schemes. Memory schemes to tackle diversity loss have rarely been studied in PSO, since each particle in the swarm has its own memory.

Campos & Krohling (2016) proposed an entropy-based bare bones particle swarm (EBBPSO-T) for solving DCOPs. In this work, the Shannon's entropy is established as a diversity index and the proposed algorithm applies the idea of mixture of search directions by using the Shannon's index of diversity as a factor to balance the influence of the global-best and local-best search directions. High diversity promotes the search guided by the global-best solution and low diversity promotes the search guided by the local-best solution. A strategy to handle constraints is proposed using a ranking method combined with the technique for order of preference by similarity to ideal solution (Hwang & Yoon, 1981; Behzadian et al., 2012) to select the best solution within a population of candidate solutions. Mechanisms to detect changes in the environment and to update particles' memories are also implemented into the proposed algorithm. All these strategies operate related to each other to tackle problems such as diversity loss and outdated memories. The combined effect of these strategies provides an algorithm with ability to maintain a proper balance between exploration and exploitation at any stage of the search process without losing the tracking ability to search an optimal solution that is changing over time. EBBPSO-T was tested on the G24 benchmark set and its performance was compared to the performances of GSARepair and GA-based approaches. For more details about this algorithm the reader is referred to Chapter 4.

Bu, Luo & Yue (2017) adapted the dynamic species-based PSO (DSPSO), a representative multipopulation algorithm, to locate and track multiple feasible regions in parallel using an ensemble of different strategies specifically developed to solve DCOPs with this characteristic. Experiments show that the DSPSO with the ensemble of strategies performs significantly better than the original DSPSO and other compared algorithms.

2.5 Benchmark problems

One useful way to create dynamic benchmark problems is to combine existing static benchmark problems with dynamic rules found in dynamic constrained applications (Nguyen & Yao, 2009; Nguyen, 2010; Nguyen & Yao, 2012). In fact, this can be done by applying the dynamic rules to the parameters of the static problems. Formally, this idea can be described as follows. Consider a static function $f(\mathbf{x}|\Theta)$ with a set of parameters $\Theta = \{\theta_1, \theta_2, \ldots\}$. It is possible to generalize $f(\mathbf{x}|\Theta)$ to its dynamic version by replacing each static parameter in Θ by a time-dependent expression $\theta_i(t)$. The dynamic of the time-dependent problem then depends on how $\theta_i(t)$ varies over time. As a result, a dynamic function $f(\mathbf{x},t) = f(\mathbf{x}|\Theta(t))$ is defined. Similar reasoning can be applied to the constrained functions.

Using this idea, Nguyen & Yao (2009) introduced a set of 6 benchmark problems named the G24 set. Subsequently, Nguyen (2010) and Nguyen & Yao (2012) expanded this class of DCOPs to a set of 18 benchmark problems. The general form for each problem in the G24 set is defined as in Eq. (2.1), where $\mathbf{x} = (x_1, x_2) \in \mathbb{F}(t) \subseteq \mathbb{S} = [0,3] \times [0,4]$ and $t \in T = \{0, 1, \ldots, t_{\max}\}$. The objective function can take one of the following functional forms:

1. $f^{(1)} = -(F_{1,t} + F_{2,t})$
2. $f^{(2)} = -3\exp\left(-\sqrt{\sqrt{F_{1,t}^2 + F_{2,t}^2}}\right)$

where $F_{j,t} = F_{j,t}(x_j, t) = p_j(t)(x_j + q_j(t))$ with $p_j(t)$ and $q_j(t)$, $j = 1,2$, as the dynamic parameters which determine how the objective function of each benchmark problem changes over time. The constraint functions can take the following functional forms:

1. $g^{(1)} = -2G_{1,t}^4 + 8G_{1,t}^3 - 8G_{1,t}^2 + G_{2,t} - 2$
2. $g^{(2)} = -4G_{1,t}^4 + 32G_{1,t}^3 - 88G_{1,t}^2 + 96G_{1,t} + G_{2,t} - 36$
3. $g^{(3)} = 2G_{1,t} + 3G_{2,t} - 9$
4. $g^{(4)} = -1$ if $(0 \leq G_{1,t} \leq 1)$ or $(2 \leq G_{1,t} \leq 3)$; $g^{(4)} = 1$ otherwise
5. $g^{(5)} = -1$ if $(0 \leq G_{1,t} \leq 0.5)$ or $(2 \leq G_{1,t} \leq 2.5)$; $g^{(5)} = 1$ otherwise
6. $g^{(6)} = -1$ if $[(0 \leq G_{1,t} \leq 1)$ and $(2 \leq G_{2,t} \leq 3)]$ or $(2 \leq G_{1,t} \leq 3)$; $g^{(6)} = 1$ otherwise

where $G_{j,t} = G_{j,t}(x_j, t) = r_j(t)(x_j + s_j(t))$ with $r_j(t)$ and $s_j(t)$, $j = 1,2$, as the dynamic parameters which determine how the constraint functions of each benchmark problem changes over time.

Each benchmark problem in the G24 set has a different mathematical expression for $p_j(t), q_j(t), r_j(t)$ and $s_j(t)$:

1. G24-u. $f = f^{(1)}$ with $p_1(t) = \sin(\kappa\pi t + \pi/2)$, $p_2(t) = 1$, and $q_{1,2}(t) = 0$.
2. G24-1. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_1(t) = \sin(\kappa\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, and $s_{1,2}(t) = 0$.
3. G24-f. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_{1,2}(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, and $s_{1,2}(t) = 0$.
4. G24-uf. $f = f^{(1)}$ with $p_{1,2}(t) = 1$ and $q_{1,2}(t) = 1$.
5. G24-2. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_1(t) = \sin(\kappa\pi t/2 + \pi/2)$,

$$p_2(t) = \begin{cases} 0 & \text{if } t = 0 \\ p_2(t-1) & \text{if } t \bmod 2 = 0 \\ \sin(\kappa\pi(t-1)/2 + \pi/2) & \text{if } t \bmod 2 \neq 0 \end{cases}$$

   $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, and $s_{1,2}(t) = 0$.
6. G24-2u. $f = f^{(1)}$ with $p_1(t) = \sin(\kappa\pi t/2 + \pi/2)$,

$$p_2(t) = \begin{cases} 0 & \text{if } t = 0 \\ p_2(t-1) & \text{if } t \bmod 2 = 0 \\ \sin(\kappa\pi(t-1)/2 + \pi/2) & \text{if } t \bmod 2 \neq 0 \end{cases}$$

   $q_{1,2}(t) = 0$.
7. G24-3. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_{1,2}(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = 2 + (\delta/S)t$.
8. G24-3b. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_1(t) = \sin(\kappa\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = 2 + (\delta/S)t$.

9. G24-3f. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_{1,2}(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = 2$.
10. G24-4. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_1(t) = \sin(\kappa\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = (\delta/S)t$.
11. G24-5. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_1(t) = \sin(\kappa\pi t/2 + \pi/2)$,

$$p_2(t) = \begin{cases} 0 & \text{if } t = 0 \\ p_2(t-1) & \text{if } t \bmod 2 = 0 \\ \sin(\kappa\pi(t-1)/2 + \pi/2) & \text{if } t \bmod 2 \neq 0 \end{cases}$$

$q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = (\delta/S)t$.

12. G24-6a. $f = f^{(1)}$, $g_1 = g^{(3)}$, and $g_2 = g^{(6)}$ with $p_1(t) = \sin(\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_{1,2}(t) = 0$.
13. G24-6b. $f = f^{(1)}$ and $g_1 = g^{(3)}$ with $p_1(t) = \sin(\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_{1,2}(t) = 0$.
14. G24-6c. $f = f^{(1)}$, $g_1 = g^{(3)}$, and $g_2 = g^{(4)}$ with $p_1(t) = \sin(\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_{1,2}(t) = 0$.
15. G24-6d. $f = f^{(1)}$, $g_1 = g^{(5)}$, and $g_2 = g^{(6)}$ with $p_1(t) = \sin(\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_{1,2}(t) = 0$.
16. G24-7. $f = f^{(1)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_{1,2}(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = (\delta/S)t$.
17. G24-8a. $f = f^{(2)}$ with $p_{1,2}(t) = -1$, $q_1(t) = -(C_1 + R_a\cos(\kappa\pi t))$, and $q_2(t) = -(C_2 + R_a\sin(\kappa\pi t))$.
18. G24-8b. $f = f^{(2)}$, $g_1 = g^{(1)}$, and $g_2 = g^{(2)}$ with $p_{1,2}(t) = -1$, $q_1(t) = -(C_1 + R_a\cos(\kappa\pi t))$, and $q_2(t) = -(C_2 + R_a\sin(\kappa\pi t))$, $r_{1,2}(t) = 1$, and $s_{1,2}(t) = 0$.

The parameter $\kappa \in \{1, 1/2, 1/4\}$ determines the severity of objective function changes and the parameter $S \in \{10, 20, 50\}$ determines the severity of constraint changes. In addition, consider that $\delta = 4$, $C_1 = 1.470561702$, $C_2 = 3.442094786232$, and finally $R_a = 0.858958496$.

2.6 Examples of benchmark problems

This section presents some examples of problems in the G24 benchmark set of DCOPs.

Example 2.1. The G24-1 problem is defined as follows:

$$\begin{array}{rl} \min & -\sin[\kappa\pi(t+1)]x_1 - x_2 \\ \text{subject to} & -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0 \\ & -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0 \end{array} \tag{2.17}$$

where $\mathbf{x} = (x_1, x_2)' \in \mathbb{F}(t) \subset \mathbb{S} = [0,3] \times [0,4]$ and $t \in \{0, 1, 2, \ldots\}$. Figure 2.1 shows a graphical representation of this problem in different environments by considering $\kappa = 1/2$, where κ determines the severity of objective function changes. The objective function is represented by contour lines, the constraints are represented by lines with blue and red colors, and the global optimum is represented by a black dot. The objective function is dynamic and the constraints are static. The location of the global optimum is affected, i.e., it is switched (or transferred) between two disconnected feasible regions. Using the optimality conditions discussed in Section 2.3 it is possible to obtain the global optimum of this dynamic problem for different environments. Table 2.1 shows the results for the environments corresponding to $t = 0, 1, 2, \ldots, 10$: $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)'$ is the global optimum, $\bar{\mu}_1$ and $\bar{\mu}_2$ are the associated Lagrange multiplier, $f(\bar{\mathbf{x}}, t)$ is the global optimum value of $\bar{\mathbf{x}}$ for each t, and both constraints are active at $\bar{\mathbf{x}}$ for each t.

Example 2.2. The G24-3 problem is defined as follows:

$$\begin{array}{rl} \min & -x_1 - x_2 \\ \text{subject to} & -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - (\delta/S)t \leq 0 \\ & -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - (\delta/S)t - 34 \leq 0 \end{array} \tag{2.18}$$

where $\mathbf{x} = (x_1, x_2)' \in \mathbb{F}(t) \subset \mathbb{S} = [0,3] \times [0,4]$ and $t \in \{0,1,2,\dots\}$. Figure 2.2 shows a graphical representation of this problem in different environments by considering $\delta = 4$ and $S = 20$, where S determines the severity of constraint changes. The objective function is static and the constraints are dynamic. As a result, the size and the shape of the feasible regions change over time and a new global optimum is revealed whenever the environment is changed. However, note that the old optimum value is not changed, since the objective function is static. Using the optimality conditions discussed in Section 2.3 it is possible to obtain the global optimum of this dynamic problem for different environments. Table 2.2 shows the results for the environments corresponding to $t = 0,1,2,\dots,10$: $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)'$ is the global optimum, $\bar{\mu}_1$ and $\bar{\mu}_2$ are the associated Lagrange multiplier, $f(\bar{\mathbf{x}}, t)$ is the global optimum value of $\bar{\mathbf{x}}$ for each t, and both constraints are active at $\bar{\mathbf{x}}$ for each t.

Example 2.3. The G24-4 problem is defined as follows:

$$\begin{array}{rl} \min & -\sin[\kappa\pi(t+1)]x_1 - x_2 \\ \text{subject to} & -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 + (\delta/S)t - 2 \leq 0 \\ & -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 + (\delta/S)t - 36 \leq 0 \end{array} \tag{2.19}$$

where $\mathbf{x} = (x_1, x_2)' \in \mathbb{F}(t) \subset \mathbb{S} = [0,3] \times [0,4]$ and $t \in \{0,1,2,\dots\}$. Figure 2.3 shows a graphical representation of this problem in different environments by considering $\kappa = 1/2$, $\delta = 4$, and $S = 20$, where κ determines the severity of objective function changes and S determines the severity of constraint changes. Note that the objective function and the constraints change over time. As a result, the location of the global optimum is affected, i.e., it is switched between two disconnected feasible regions. In addition, the size and the shape of the feasible regions also change over time. Using the optimality conditions discussed in Section 2.3 it is possible to obtain the global optimum of this dynamic problem for different environments. Table 2.3 shows the results for the environments corresponding to $t = 0,1,2,\dots,10$: $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)'$ is the global optimum, $\bar{\mu}_1$ and $\bar{\mu}_2$ are the associated Lagrange multiplier, $f(\bar{\mathbf{x}}, t)$ is the global optimum value of $\bar{\mathbf{x}}$ for each t, and both constraints are active at $\bar{\mathbf{x}}$ for each t.

2.7 Advantages and limitations of the benchmark problems

It is important to argue the advantages and limitations of the G24 set of benchmark problems for dynamic constrained optimization. Table 2.4 shows the main properties of each problem in this set, which shows a diversity of problems presenting all the essential characteristics that were discussed in Section 2.2. A particular problem can be interpreted as follows. For instance, the G24-4 problem has objective function and constraints that change over time. Depending of the current environment (i.e., of the current value of t), this problem has the feasible space composed of up to 3 disconnected feasible regions (DFR). The global optimum is switched between the disconnected feasible regions (SwO). For all environments, the global optimum is in the constraint boundary (OICB). Once again, the G24-1 problem has dynamic objective function and static constraints. For all environments, this problem has the feasible space consisting of exactly 2 disconnected feasible regions (DFR). The global optimum is switched between the disconnected feasible regions (SwO). For all environments, the global optimum is in the constraint boundary (OICB). The G24-3 problem has static objective function and dynamic constraints. Depending of the current environment (i.e., of the current value of t), this problem has the feasible space composed of up to 3 disconnected feasible regions (DFR). A new global optimum is revealed whenever the environment is changed, but the old optimum value is not changed since the objective function is static (bNAO). For all environments, the global optimum is in the constraint boundary (OICB). The G24-7 problem has similar characteristics to the G24-3 problem. For the G24-7 problem, a new global optimum is also revealed whenever the environment is changed, but in this case, the old global optimum becomes an infeasible solution for the new environment (bNAO). Analogous interpretations can be made for all other problems.

The different characteristics presented by the G24 set of DCOPs are its main advantage when compared to the DCT set of DCOPs (see Liu, 2008), which in turn does not contain this variety of problems. On the other hand, some limitations of the G24 set can be discussed. As already mentioned in Section 2.1, a DCOP can be seen as a sequence of constrained optimization problems indexed by time. Observing this fact, the main limitation of

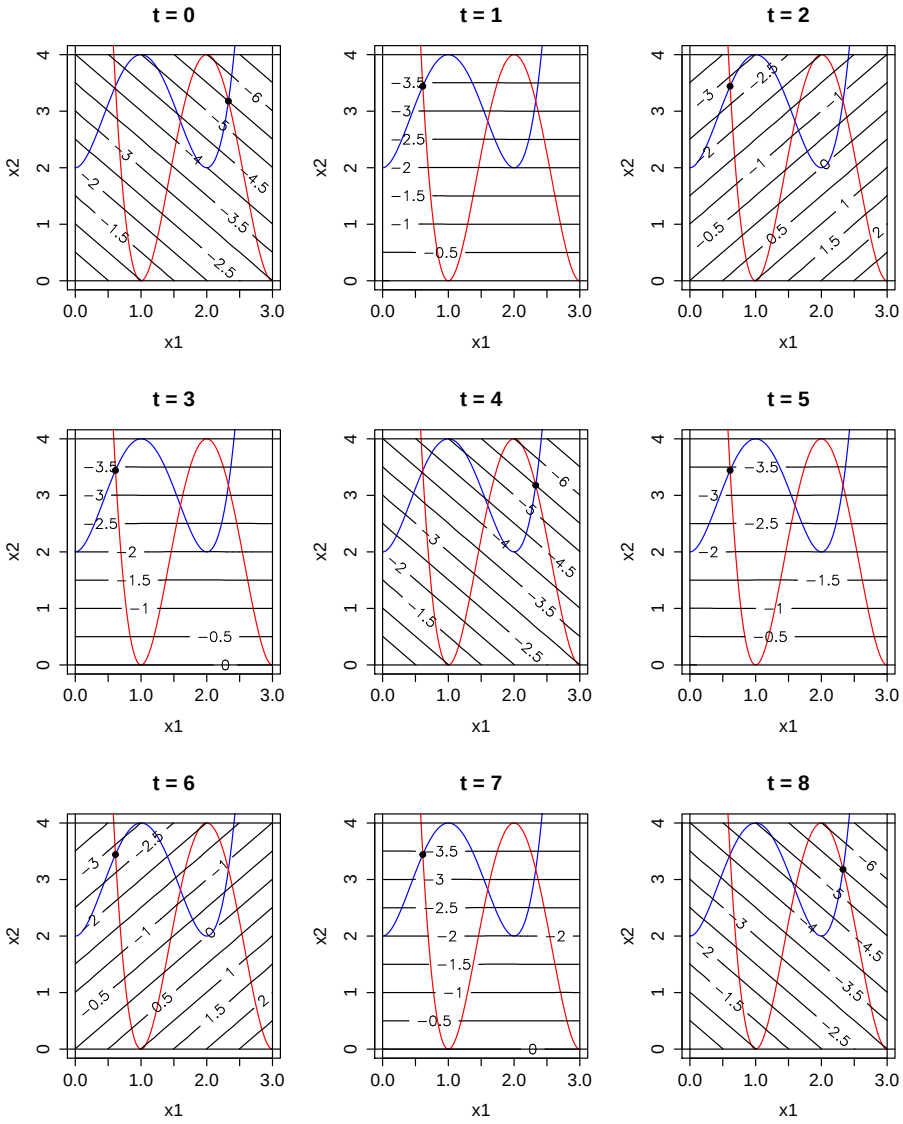

Figure 2.1: Graphical representation of the G24-1 problem (Example 2.1).

Table 2.1: Global optimum of the G24-1 problem for different environments (see Example 2.1).

t	$\bar{\mu}_1$	$\bar{\mu}_2$	$\bar{x}_1$	$\bar{x}_2$	$f(\bar{\mathbf{x}},t)$	$g_1(\bar{\mathbf{x}},t)$	$g_2(\bar{\mathbf{x}},t)$
0	0.28760	0.71240	2.32952	3.17849	-5.50801	0	0
1	0.88650	0.11350	0.61160	3.44210	-3.44210	0	0
2	0.92952	0.07048	0.61160	3.44210	-2.83050	0	0
3	0.88650	0.11350	0.61160	3.44210	-3.44210	0	0
4	0.28760	0.71240	2.32952	3.17849	-5.50801	0	0
5	0.88650	0.11350	0.61160	3.44210	-3.44210	0	0
6	0.92952	0.07048	0.61160	3.44210	-2.83050	0	0
7	0.88650	0.11350	0.61160	3.44210	-3.44210	0	0
8	0.28760	0.71240	2.32952	3.17849	-5.50801	0	0
9	0.88650	0.11350	0.61160	3.44210	-3.44210	0	0
10	0.92952	0.07048	0.61160	3.44210	-2.83050	0	0

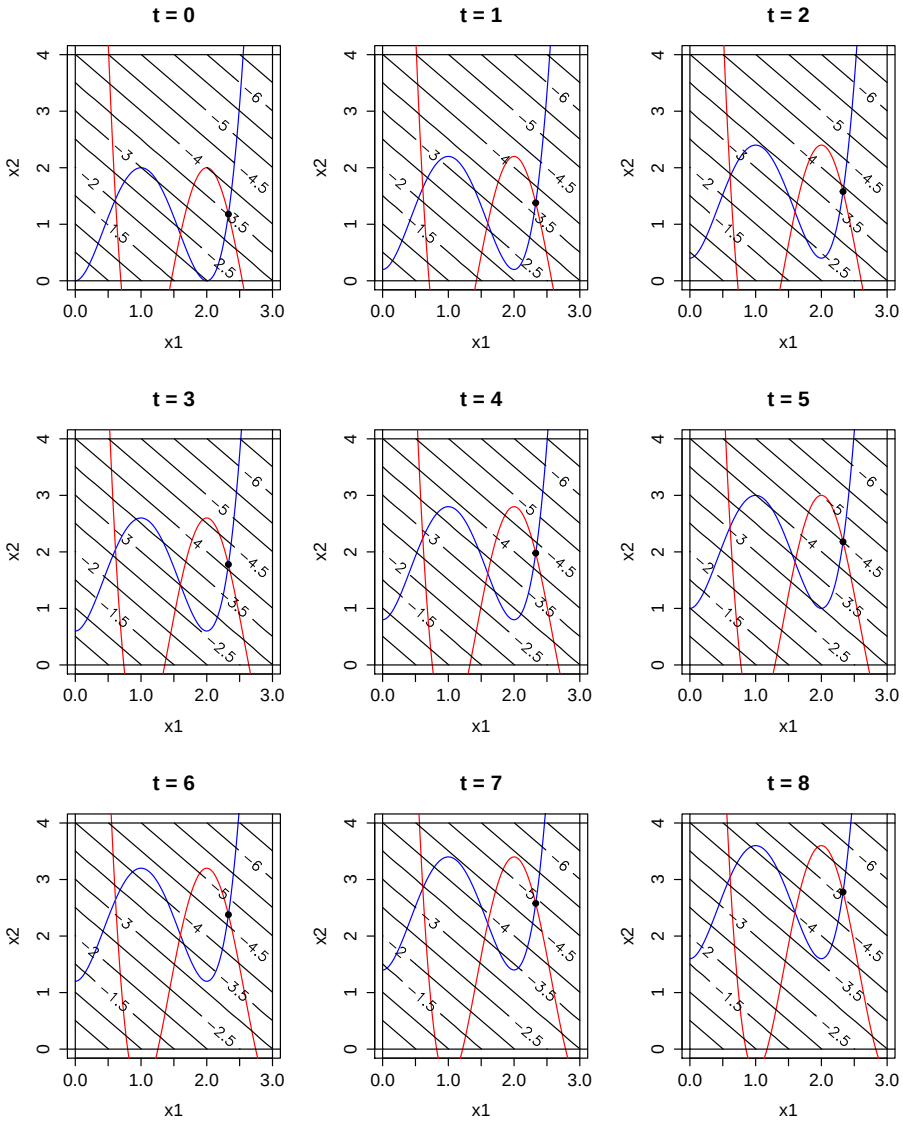

Figure 2.2: Graphical representation of the G24-3 problem (Example 2.2).

Table 2.2: Global optimum of the G24-3 problem for different environments (see Example 2.2).

t	$\bar{\mu}_1$	$\bar{\mu}_2$	$\bar{x}_1$	$\bar{x}_2$	$f(\bar{\mathbf{x}},t)$	$g_1(\bar{\mathbf{x}},t)$	$g_2(\bar{\mathbf{x}},t)$
0	0.28760	0.71240	2.32952	1.17849	-3.50801	0	0
1	0.28760	0.71240	2.32952	1.37849	-3.70801	0	0
2	0.28760	0.71240	2.32952	1.57849	-3.90801	0	0
3	0.28760	0.71240	2.32952	1.77849	-4.10801	0	0
4	0.28760	0.71240	2.32952	1.97849	-4.30801	0	0
5	0.28760	0.71240	2.32952	2.17849	-4.50801	0	0
6	0.28760	0.71240	2.32952	2.37849	-4.70801	0	0
7	0.28760	0.71240	2.32952	2.57849	-4.90801	0	0
8	0.28760	0.71240	2.32952	2.77849	-5.10801	0	0
9	0.28760	0.71240	2.32952	2.97849	-5.30801	0	0
10	0.28760	0.71240	2.32952	3.17849	-5.50801	0	0

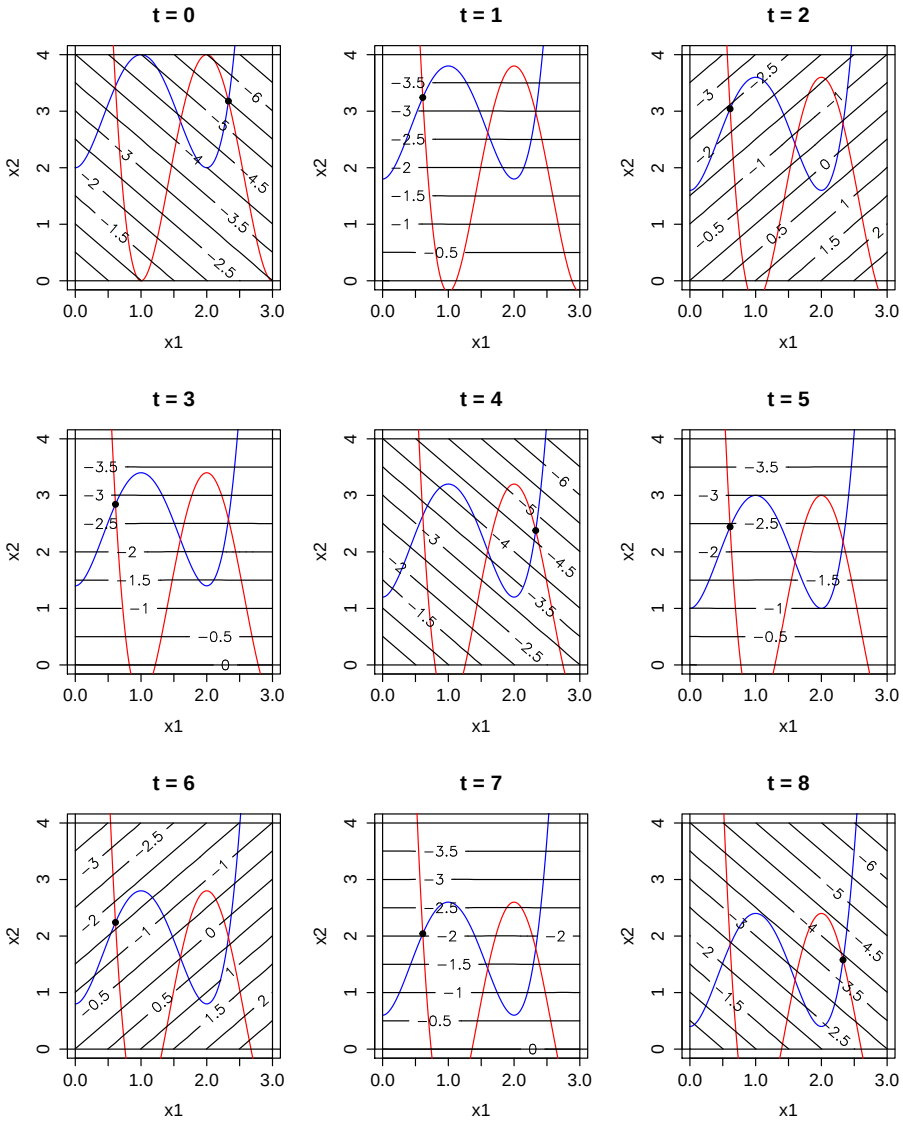

Figure 2.3: Graphical representation of the G24-4 problem (Example 2.3).

Table 2.3: Global optimum of the G24-4 problem for different environments (see Example 2.3).

t	$\bar{\mu}_1$	$\bar{\mu}_2$	$\bar{x}_1$	$\bar{x}_2$	$f(\bar{\mathbf{x}},t)$	$g_1(\bar{\mathbf{x}},t)$	$g_2(\bar{\mathbf{x}},t)$
0	0.28760	0.71240	2.32952	3.17849	-5.50801	0	0
1	0.88650	0.11350	0.61160	3.24210	-3.24210	0	0
2	0.92952	0.07048	0.61160	3.04210	-2.43050	0	0
3	0.88650	0.11350	0.61160	2.84210	-2.84210	0	0
4	0.28760	0.71240	2.32952	2.37849	-4.70801	0	0
5	0.88650	0.11350	0.61160	2.44210	-2.44210	0	0
6	0.92952	0.07048	0.61160	2.24210	-1.63050	0	0
7	0.88650	0.11350	0.61160	2.04210	-2.04210	0	0
8	0.28760	0.71240	2.32952	1.57849	-3.90801	0	0
9	0.88650	0.11350	0.61160	1.64210	-1.64210	0	0
10	0.92952	0.07048	0.61160	1.44210	-0.83050	0	0

Table 2.4: Properties of each test problem in the G24 benchmark set.

n	Name	Obj. function	Constraint	DFR	SwO	bNAO	OICB	OISB	Path
1	G24-u(dF,noC)	Dynamic	No constraint	1	No	No	No	Yes	NA
2	G24-1(dF,fC)	Dynamic	Static	2	Yes	No	Yes	No	NA
3	G24-f(fF,fC)	Static	Static	2	No	No	Yes	No	NA
4	G24-uf(fF,noC)	Static	No constraint	1	No	No	No	Yes	NA
5	G24-2(dF,fC)	Dynamic	Static	2	Yes	No	Yes/No	Yes/No	NA
6	G24-2u(dF,noC)	Dynamic	No constraint	1	No	No	No	Yes	NA
7	G24-3(fF,dC)	Static	Dynamic	2-3	No	Yes	Yes	No	NA
8	G24-3b(dF,dC)	Dynamic	Dynamic	2-3	Yes	No	Yes	No	NA
9	G24-3f(fF,fC)	Static	Static	1	No	No	Yes	No	NA
10	G24-4(dF,dC)	Dynamic	Dynamic	2-3	Yes	No	Yes	No	NA
11	G24-5(dF,dC)	Dynamic	Dynamic	2-3	Yes	No	Yes/No	Yes/No	NA
12	G24-6a(dF,fC)	Dynamic	Static	2	Yes	No	No	Yes	Hard
13	G24-6b(dF,fC)	Dynamic	Static	1	No	No	No	Yes	NA
14	G24-6c(dF,fC)	Dynamic	Static	2	Yes	No	No	Yes	Easy
15	G24-6d(dF,fC)	Dynamic	Static	2	Yes	No	No	Yes	Hard
16	G24-7(fF,dC)	Static	Dynamic	2	No	No	Yes	No	NA
17	G24-8a(dF,noC)	Dynamic	No constraint	1	No	No	No	No	NA
18	G24-8b(dF,fC)	Dynamic	Static	2	Yes	No	Yes	No	NA

DFR: number of disconnected feasible regions
SwO: switched global optimum between disconnected feasible regions
bNAO: better newly appear optimum without changing existing ones
OICB: global optimum is in the constraint boundary
OISB: global optimum is in the search boundary
Path: indicates if it is easy or hard to travel between feasible regions

the problems in the G24 set is that the sequence of time-indexed optimization problems associated with a given dynamic problem is constituted of unlinked problems. This means that, for a given problem, the environment corresponding to time t is not influenced by previous environments corresponding to times before t and also does not affect subsequent environments corresponding to times after t. In other words, the parameters that define the environment corresponding to time t is not linked with the parameters that define the environment corresponding to time s, where s is different from t. As a result, this limitation does not clarify the connection that must exist between dynamic optimization problems as defined by Eq. (2.1) and dynamic optimization problems that occur in dynamic programming and optimal control (Hinderer, Rieder, & Stieglitz, 2016). Another important limitation that should be emphasized is that all problems in the G24 set are unimodal optimization problems. In summary, it is possible to state that the G24 set of DCOPs represents a great improvement over the DCT set of DCOPs, but certainly does not synthesize all the variety of optimization problems in dynamic environments that exist in different fields of knowledge.

2.8 Algorithmic attributes to deal with dynamic optimization problems

Besides the parameters κ and S that determine, respectively, the severity of changes in the objective function and in the constraints, another important parameter must be defined when a DCOP is solved by an optimization algorithm. This parameter, denoted by Δ, determines the window where the dynamic problem remains constant. In other words, for a given fixed window, the environment of the dynamic problem is frozen and remains static within that window.

Once again, a DCOP can be seen as a sequence of constrained optimization problems which are received on-line in discrete times, each one having a window Δ to be solved. The relationship between the time t and the iteration counter τ of an algorithm used to solve a DCOP is given by

$$t = \lfloor \tau/\Delta \rfloor = \text{the largest integer not greater than the corresponding value of } \tau/\Delta. \tag{2.20}$$

For solving a

$$\text{DCOP} = \langle D, \mathbf{x} \in \mathbb{R}^D, t, f(\mathbf{x},t), g_i(\mathbf{x},t), L_d, U_d : i = 1,\ldots,I \text{ and } d = 1,\ldots,D \rangle_{t=\lfloor (0:\tau_{\max})/\Delta \rfloor} \tag{2.21}$$

an optimization algorithm has a window of Δ iterations to find the solution of the current problem (or the current environment) which is associated with the time t, before receiving the new problem (or the new environment) which is associated with time $(t+1)$. A DCOP is solved in

$$\tau_{\max} + 1 = \Delta \cdot (t_{\max} + 1) \text{ iterations.} \tag{2.22}$$

To deal with DCOPs, optimization algorithms must be able to detect changes in the environment and efficiently respond to the changed environment. Algorithms must be capable of maintaining a good balance between exploration (diversification) and exploitation (intensification). Too much stress on exploration would result in pure global search and slow convergence speed. On the other hand, too much stress on exploitation would result in pure local search and fast loss of tracking ability to search an optimal solution which is changing over time.

2.9 Performance measures

To investigate the performance of an algorithm on a particular problem it is necessary to define a performance measure. There are two classes of performance measures in dynamic optimization: behaviour-based performance measures and optimality-based performance measures (Nguyen, Yang & Branke, 2012).

Behaviour-based performance measures usually describe the population diversity in population-based algorithms when these algorithms are solving dynamic optimization problems. Chapter 3 will establish the Shannon's entropy as a index of phenotypic diversity of a particle population and will discuss an estimation procedure for this index from small-sample data. In the following chapter (Chapter 4), the Shannon's index will be used to monitor the population diversity of an optimization algorithm and will have an important role as a mechanism to maintain or introduce diversity during the search process.

Optimality-based performance measures usually describe the performance of an algorithm by using fitness values associated with potential solutions and reference solutions. One optimality-based measure widely used is the offline error, which is defined by:

$$\text{Offline Error} = \frac{1}{\tau_{\max}} \sum_{\tau=1}^{\tau_{\max}} e(\tau) \tag{2.23}$$

where $\tau_{\max}$ is the number of iterations and $e(\tau)$ is the best error (or minimum error) obtained by the algorithm since the last change at the iteration τ. The error means the absolute difference between the fitness of a solution and the fitness of the global optimum. This measure is always greater than or equal to zero and would be zero for a perfect performance. Researchers also use a measure named by offline performance that takes the following form:

$$\text{Offline Performance} = \frac{1}{\tau_{\max}} \sum_{\tau=1}^{\tau_{\max}} F(\tau) \tag{2.24}$$

where $F(\tau)$ is the best fitness obtained by the algorithm since the last state change at the iteration τ. This measure is provided to evaluate the performance of an algorithm in cases where exact values of the global optimum are not known. Another measure widely used is the best-error-before-change, which is defined by:

$$\text{Best-Error-Before-Change} = \frac{1}{NC} \sum_{l=1}^{NC} e(l) \tag{2.25}$$

where NC is the number of changes and $e(l)$ is the best error just before the l-th change happens. Such optimality-based measures will be used to measure the performance of different algorithms in Chapter 5.

Chapter 3

Densities, Entropy, and Ranking

This chapter compiles some definitions, concepts, and methods that are important for this thesis. It serves as a quick reference for several topics as: random variables, entropy of a random variable, entropy estimation procedures, relationship between diversity and entropy, and an exploratory method to rank decision alternatives based on multi-criteria. These concepts and methods are essential to understand the subsequent development of the proposed algorithm in this thesis for solving dynamic constrained optimization problems, whose detailed development will be given in Chapter 4.

3.1 Probability densities

Let $(\Omega, \mathscr{F}, \Pr)$ be a probability space for a given random experiment. It means that $(\Omega, \mathscr{F})$ is a measurable space and Pr is a finite measure on $\mathscr{F}$ with $\Pr(\Omega) = 1$. The set Ω represents the set of all possible outcomes of the experiment and $\mathscr{F}$ is a σ-algebra of subsets of Ω. The set Ω is called of sample space. Elements of the class $\mathscr{F}$ are called of events and $\Pr(\mathsf{A})$ represents the probability of an event A in $\mathscr{F}$. Suppose that an event B has occurred and that it is important to compute the probability of another event A taking into account that B has occurred. This probability is called the conditional probability of A given that the event B has occurred and it is denoted by $\Pr(\mathsf{A}|\mathsf{B})$. If $\Pr(\mathsf{B}) > 0$, this probability is calculated as $\Pr(\mathsf{A}|\mathsf{B}) = \Pr(\mathsf{A} \cap \mathsf{B}) / \Pr(\mathsf{B})$; otherwise (i.e., if $\Pr(\mathsf{B}) = 0$) $\Pr(\mathsf{A}|\mathsf{B})$ can be arbitrarily defined, for example, $\Pr(\mathsf{A}|\mathsf{B}) = \Pr(\mathsf{A})$. Events A and B are called statistically independent with respect to the probability Pr, if $\Pr(\mathsf{A} \cap \mathsf{B}) = \Pr(\mathsf{A}) \Pr(\mathsf{B})$, and consequently $\Pr(\mathsf{A}|\mathsf{B}) = \Pr(\mathsf{A})$ in this case.

A random variable is a random point in the real line $\mathbb{R}$. Formally, a real-valued random variable x is a function defined in Ω and taking values in $\mathbb{R}$ such that the set $[x \in B] = \{\omega \in \Omega : x(\omega) \in B\}$ is an event in $\mathscr{F}$ for any Borel set $B \subseteq \mathbb{R}$. It means that x is a $\mathscr{F}$-measurable function and $\Pr(x \in B)$ is a well-defined quantity for any Borel set $B \subseteq \mathbb{R}$. A probability measure P_x defined as

$$P_x(B) = \Pr(x \in B) \tag{3.1}$$

for any Borel set $B \subseteq \mathbb{R}$ is called the probability distribution of x. A random variable is called discrete if its distribution is concentrated on a countable set and

$$P_x(B) = \int_B P_x(dx) = \sum_{\{m : x_m \in B\}} p_m \tag{3.2}$$

for any Borel set B, where $\sum_m p_m = 1$ and $p_m = \Pr(x = x_m)$. A random variable is called absolutely continuous if there is a non-negative function $p(x)$, called density or probability density function (pdf), such that

$$P_x(B) = \int_B P_x(dx) = \int_{\{x : x \in B\}} p(x)dx \tag{3.3}$$

for any Borel set B, where $\int p(x)dx = P_x(\mathbb{R}) = 1$.

Let x be a random variable and suppose that at least one of the following integrals is finite: $\mathrm{E}_- = \int_{x \leq 0} xP_x(dx)$, $\mathrm{E}_+ = \int_{x \geq 0} xP_x(dx)$. Then the expectation (or mean) of x is said to exist and is defined as $\mathrm{E}(x) = \int xP_x(dx)$. If E_- and

E_+ are infinite, then $E(x)$ does not exist. The variance of x is defined by $\text{Var}(x) = E[(x - E(x))^2] = E(x^2) - [E(x)]^2$ (if $E(x) < \infty$). If x has infinite mean or if the mean of x does not exist, then $\text{Var}(x)$ does not exist. The mean of a random variable indicates its central value, being an useful summary value of the distribution. The variance of a random variable measures the variability of the distribution around its mean.

The concept of distribution of a random variable can be generalized to the joint distribution of two random variables. From now on, the discussion will be focused on the case of continuous random variables, however the discrete and hybrid cases are similar. Two random variables x and y have an absolutely continuous joint distribution if there is a non-negative function $p(x,y)$, called joint density or joint pdf, such that

$$P_{x,y}(B) = \int_B P_{x,y}(dxdy) = \int_{\{(x,y):(x,y)\in B\}} p(x,y)dxdy \tag{3.4}$$

for any Borel set B in $\mathbb{R}^2$, where $\int p(x,y)dxdy = 1$. Two important rules can be presented involving the case of joint pdf:

Product rule. $p(x,y) = p(x|y)p(y)$ where $p(x|y) = p(x,y)/p(y)$ if $p(y) > 0$ (for $x \in \mathbb{R}$).

Sum rule. $p(x) = \int p(x,y)dy = \int p(x|y)p(y)dy$.

Assuming that the expectations are defined, the covariance of x and y, denoted by $\text{Cov}(x,y)$, is defined as follows:

$$\text{Cov}(x,y) = E[(x - E(x))(y - E(y))] = E(xy) - E(x)E(y). \tag{3.5}$$

The value of $\text{Cov}(x,y)$ can be positive, negative, or zero and it is a measure of the linear association between x and y. Note that, $\text{Cov}(x,x) = \text{Var}(x)$. In addition, if $\text{Var}(x) > 0$ and $\text{Var}(y) > 0$, the number

$$-1 \leq \frac{\text{Cov}(x,y)}{\sqrt{\text{Var}(x)\text{Var}(y)}} \leq 1 \tag{3.6}$$

is the correlation coefficient of x and y. If $\text{Cov}(x,y) = 0$, then x and y are uncorrelated. Another point to be discussed is the extension of the concept of independence for random variables. Random variables x and y are independent if and only if the following factorization

$$p(x,y) = p(x)p(y) \tag{3.7}$$

is satisfied for all real numbers x and y. If x and y are independent, then $\text{Cov}(x,y) = 0$, because $E(xy) = E(x)E(y)$. Finally, it is important to note that uncorrelated random variables can be dependent.

A random vector is a vector whose elements are random variables, i.e., a D-dimensional random vector is a random point in $\mathbb{R}^D$. In fact, if $\mathbf{x}$ is a random vector in $\mathbb{R}^D$ and $\text{Proj}_i(\cdot)$ is the projection of $\mathbb{R}^D$ on the ith coordinate axis, then $\mathbf{x}$ can be represented in the form $\mathbf{x} = (x_1, \ldots, x_D)'$, where $x_i = \text{Proj}_i(\mathbf{x})$ is a real-valued random variable. Therefore, $\mathbf{x}$ has a joint pdf, denoted by $p(\mathbf{x})$, such that

$$P_{\mathbf{x}}(B) = \int_B P_{\mathbf{x}}(d\mathbf{x}) = \int_{\{\mathbf{x}:\mathbf{x}\in B\}} p(\mathbf{x})d\mathbf{x} \tag{3.8}$$

for any Borel set B in $\mathbb{R}^D$, where $\int p(\mathbf{x})d\mathbf{x} = 1$. Each element of $\mathbf{x}$ has its own probability distribution $p(x_i)$, mean $\mu_i = E(x_i)$, and variance $\sigma_i^2 = \sigma_{ii} = \text{Var}(x_i)$. The behaviour of any pair of random variables, such as x_i and x_j, is described by their joint pdf $p(x_i,x_j)$ and a measure of the linear association between then is provided by $\sigma_{ij} = \text{Cov}(x_i,x_j)$, as discussed early. All this information can be summarized in matrix notation. The mean of each element of $\mathbf{x}$ is contained in the mean vector

$$\boldsymbol{\mu} = E(\mathbf{x}) = (\mu_1, \ldots, \mu_D)' = (\mu_i)_{i=1,\ldots,D}. \tag{3.9}$$

The D variances σ_{ii} and the $D(D-1)/2$ distinct covariances σ_{ij} $(i < j)$ are contained in the symmetric covariance matrix

$$\boldsymbol{\Sigma} = \text{Cov}(\mathbf{x}) = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})'] = (\sigma_{ij})_{i,j=1,\ldots,D}. \tag{3.10}$$

The D random variables $x_1, \ldots, x_D$ are mutually statistically independent if and only if their joint pdf can be factored as

$$p(\mathbf{x}) = p(x_1) \cdot \ldots \cdot p(x_D) \tag{3.11}$$

for all $(x_1,\ldots,x_D)' \in \mathbb{R}^D$. Statistical independence has an important implication for covariance:

$$p(\mathbf{x}) = p(x_1)\cdot\ldots\cdot p(x_D) \Rightarrow \mathbf{\Sigma} = \text{Diag}(\sigma_{11},\ldots,\sigma_{DD}). \tag{3.12}$$

The converse of Eq. (3.12) is not true in general.

Example 3.1. This example presents a list of densities that will be used in this work:

1. A random variable x has a uniform distribution in (a,b), denoted by $x \sim \text{Unif}(a,b)$, if its pdf is given by

$$\text{Unif}(x|a,b) = (b-a)^{-1} \tag{3.13}$$

for all $-\infty < a < x < b < \infty$. This distribution has mean $(a+b)/2$ and variance $(b-a)^2/12$.

2. A random variable x has a gamma distribution with parameters a and b, denoted by $x \sim Ga(a,b)$, if its pdf is given by

$$Ga(x|a,b) = \frac{b^a}{\Gamma(a)} x^{a-1}\exp(-bx) \tag{3.14}$$

for all $x > 0$ and $a,b > 0$, where $\Gamma(\cdot)$ is the well-known gamma function. This distribution has mean a/b and variance a/b^2.

3. A random vector $\mathbf{x}$ has a multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$, denoted by $\mathbf{x} \sim \mathbf{N}(\boldsymbol{\mu},\mathbf{\Sigma})$, if its joint pdf is given by

$$\mathbf{N}(\mathbf{x}|\boldsymbol{\mu},\mathbf{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}\mathbf{\Delta}^2(\mathbf{x}|\boldsymbol{\mu},\mathbf{\Sigma})\right] \tag{3.15}$$

where $\mathbf{x} \in \mathbb{R}^D$, $\boldsymbol{\mu} \in \mathbb{R}^D$, $\mathbf{\Sigma}$ is a $D \times D$ symmetric positive definite matrix, and Δ^2 is the squared Mahalanobis distance defined by $\mathbf{\Delta}^2(\mathbf{x}|\boldsymbol{\mu},\mathbf{\Sigma}) = (\mathbf{x}-\boldsymbol{\mu})'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$. When $D = 1$, x has a univariate normal distribution with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 > 0$, denoted by $x \sim N(\mu,\sigma^2)$. When $\mu = 0$ and $\sigma^2 = 1$, $N(0,1)$ is referred to as standard normal distribution.

4. A random vector $\mathbf{x}$ has a multivariate t-distribution with location $\boldsymbol{\mu}$, scale $\mathbf{C}$, and ν degrees of freedom, denoted by $\mathbf{x} \sim \mathbf{t}(\nu,\boldsymbol{\mu},\mathbf{C})$, if its joint pdf is given by

$$\mathbf{t}(\mathbf{x}|\nu,\boldsymbol{\mu},\mathbf{C}) = \frac{\Gamma((\nu+D)/2)}{\Gamma(\nu/2)(\nu\pi)^{D/2}|\mathbf{C}|^{1/2}} \left[1+\frac{1}{\nu}\mathbf{\Delta}^2(\mathbf{x}|\boldsymbol{\mu},\mathbf{C})\right]^{-\frac{\nu+D}{2}} \tag{3.16}$$

where $\mathbf{x} \in \mathbb{R}^D$, $\boldsymbol{\mu} \in \mathbb{R}^D$, $\mathbf{C}$ is a $D \times D$ symmetric positive definite matrix, and $\nu > 0$. This distribution has mean $\boldsymbol{\mu}$, for $\nu > 1$, and covariance matrix $[\nu/(\nu-2)]\mathbf{C}$, for $\nu > 2$. When $D = 1$, x has a univariate t-distribution with location $\mu \in \mathbb{R}$, scale $c > 0$, and $\nu > 0$ degrees of freedom, denoted by $x \sim t(\nu,\mu,c^2)$. In the univariate case, this distribution has mean μ, for $\nu > 1$, and variance $[\nu/(\nu-2)]c$, for $\nu > 2$. When $\mu = 0$ and $c = 1$, $t(\nu,0,1)$ is referred to as standard t-distribution.

This section ends discussing types of convergence of sequences of random variables to some limit random variable. A sequence $(x_n)_{n\geq 1}$ of random variables converges in rth mean ($r \geq 1$) to the random variable x if $\text{E}(x_n^r) < \infty$ for all $n \geq 1$, $\text{E}(x) < \infty$, and

$$\lim_{n\to\infty} \text{E}(|x_n - x|^r) \to 0.$$

This type of convergence is often denoted by $x_n \xrightarrow{r} x$. The most important cases of convergence in rth mean are:

1. $x_n \xrightarrow{1} x$ that means x_n converges in mean to x.
2. $x_n \xrightarrow{2} x$ that means x_n converges in mean square to x.

Convergence in rth mean (for $r \geq 1$) implies convergence in probability (by Markov's inequality), which in turn means that

$$\lim_{n\to\infty} \Pr(|x_n - x| \geq \varepsilon) = 0$$

for all $\varepsilon > 0$. Convergence in probability is often denoted by $x_n \xrightarrow{\text{Pr}} x$. In addition, if $r > s \geq 1$, then convergence in rth mean implies convergence in sth mean (by Lyapunov's inequality). Hence, convergence in mean square implies convergence in mean. Finally, x_n converges in mean square to a constant c if and only if $\text{E}(x_n) \to c$ and $\text{Var}(x_n) \to 0$.

3.2 Scale mixtures of normal distributions

Scale mixtures of normal distributions (SMN distributions) or scale mixtures of Gaussians (Andrews & Mallows, 1974; Choy & Chan, 2008) are derived by mixing a normally distributed random vector $\mathbf{y}$ with a non-negative random variable λ as follows:

$$\mathbf{x}|\lambda = \boldsymbol{\mu} + \phi^{1/2}(\lambda)\mathbf{y} \tag{3.17}$$

where $\boldsymbol{\mu}$ is a location parameter, $\phi(\cdot)$ is a positive function, and $\mathbf{y} \sim \mathbf{N}(\mathbf{0}, \boldsymbol{\Sigma})$. The random variable λ is independent of $\mathbf{y}$ and has a pdf $h(\lambda|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a parameter vector indexing the distribution of λ. Given λ, $\mathbf{x}|\lambda \sim \mathbf{N}(\boldsymbol{\mu}, \phi(\lambda)\boldsymbol{\Sigma})$ and the pdf of $\mathbf{x}$ is given by

$$p(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_0^\infty \mathbf{N}(\mathbf{x}|\boldsymbol{\mu}, \phi(\lambda)\boldsymbol{\Sigma}) h(\lambda|\boldsymbol{\theta}) d\lambda. \tag{3.18}$$

The pdf h is referred to as the mixing density of the SMN representation of the distribution of $\mathbf{x}$, which in turn can be expressed hierarchically as

$$\mathbf{x}|\lambda \sim \mathbf{N}(\boldsymbol{\mu}, \phi(\lambda)\boldsymbol{\Sigma}) \qquad \lambda \sim h(\boldsymbol{\theta}). \tag{3.19}$$

From a suitable choice of the density h, a rich class of continuous, symmetric, and unimodal distributions can be described by $p(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ that can readily accommodate a thicker-than-normal process. The normal distribution can be retrieved when $\lambda = 1$ almost surely.

Example 3.2. The multivariate t-distribution with location $\boldsymbol{\mu}$, scale $\boldsymbol{\Sigma}$, and ν degrees of freedom is an important heavy-tailed distribution. It is an example of scale mixture of Gaussians which can be described as

$$\mathbf{t}(\mathbf{x}|\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_0^\infty \mathbf{N}(\mathbf{x}|\boldsymbol{\mu}, \lambda^{-1}\boldsymbol{\Sigma}) Ga(\lambda|\nu/2, \nu/2) d\lambda. \tag{3.20}$$

As a result, the distribution of $\mathbf{x}$ can be expressed hierarchically as

$$\mathbf{x}|\lambda \sim \mathbf{N}(\boldsymbol{\mu}, \lambda^{-1}\boldsymbol{\Sigma}) \qquad \lambda \sim Ga(\nu/2, \nu/2). \tag{3.21}$$

Note that the Cauchy distribution can be retrieved when $\nu = 1$ and the normal distribution when $\nu \to \infty$.

Example 3.3. The univariate case $x \sim t(\nu, \mu, \sigma^2)$ is similar. The distribution of x can be expressed hierarchically as

$$x|\lambda \sim N(\mu, \lambda^{-1}\sigma^2) \qquad \lambda \sim Ga(\nu/2, \nu/2). \tag{3.22}$$

and

$$t(x|\nu, \mu, \sigma^2) = \int_0^\infty N(x|\mu, \lambda^{-1}\sigma^2) Ga(\lambda|\nu/2, \nu/2) d\lambda. \tag{3.23}$$

See Andrews & Mallows (1974); Choy & Chan (2008); Madan & Seneta (1990); Abanto-Valle et al. (2010) for further examples of SMN distributions.

3.3 The Shannon's entropy

Entropy is a fundamental quantity in physics, statistics, and computational intelligence with a large number of applications (Shannon, 1948; Renyi, 1961; Tsallis, 1998; Chao & Shen, 2003; Hausser & Strimmer, 2009; Petalas, Parsopoulos & Vrahatis, 2007; Liu, Mernik & Bryant, 2007, 2009; Kaleli, 2014). The entropy of a random variable is a measure of its uncertainty. Since the amount of information obtained from observing the outcome of a random experiment can be considered numerically equal to the amount of uncertainty associated with the outcome of the experiment before carrying it out, then the entropy of a random variable can also be viewed as a measure of information required to describe it. Let x be a discrete random variable with a finite alphabet containing M possible states. Each state has an associated probability π_m such that $0 \leq \pi_m \leq 1$ and $\sum_{m=1}^M \pi_m = 1$. The Shannon's entropy (Shannon, 1948) of $x \sim \boldsymbol{\pi} = (\pi_1, \ldots, \pi_M)'$ is defined by

$$\mathscr{H}_b(x) \overset{\text{or}}{=} \mathscr{H}_b(\boldsymbol{\pi}) = \mathrm{E}_\pi(-\log_b \pi) = -\sum_{m=1}^M \pi_m \log_b \pi_m \qquad \text{for all } 0 < b \neq 1. \tag{3.24}$$

The convention that $0 \log_b 0 = 0$ is used and it is easily justified by continuity since $\pi \log_b \pi \to 0$ as $\pi \to 0$. Thus, adding terms of zero probability does not change the entropy. If $b = 2$, then the entropy is measured in bits. If $b = e$, then the entropy is measured in nats. Note that the entropy is a functional of the distribution $\boldsymbol{\pi}$. It does not depend of the actual values taken by x, but only of the probabilities.

3.4 Entropy properties

Result 3.1. *Properties of the Shannon's Entropy (Cover & Thomas, 2006).*

1. $\mathscr{H}_b(x) \geq 0$ *with equality if and only if* x *assumes a single value with probability 1.*
2. $\mathscr{H}_b(x) = (\log_b a) \cdot \mathscr{H}_a(x)$.
3. *For any positive integer* M, $\mathscr{H}_b(\pi_1, \ldots, \pi_M) = \mathscr{H}_b(\pi_{\tau_1}, \ldots, \pi_{\tau_M})$ *where* $(\tau_1, \ldots, \tau_M)$ *represents a particular permutation of* $(1, \ldots, M)$.
4. *For any positive integer* M, $\mathscr{H}_b$ *satisfies*

$$\mathscr{H}_b^{\min} = 0 = \mathscr{H}_b(1, \ldots, 0) \leq \mathscr{H}_b(\pi_1, \ldots, \pi_M) \leq \mathscr{H}_b\left(\frac{1}{M}, \ldots, \frac{1}{M}\right) = \log_b M = \mathscr{H}_b^{\max}. \tag{3.25}$$

5. $\mathscr{H}_b(\boldsymbol{\pi})$ *is a continuous function of* $\boldsymbol{\pi}$.
6. $\mathscr{H}_b(\boldsymbol{\pi})$ *is a concave function of* $\boldsymbol{\pi}$.
7. *For any positive integer* M, $\mathscr{H}_b$ *satisfies*

$$\mathscr{H}_b\left(\frac{1}{M}, \ldots, \frac{1}{M}\right) < \mathscr{H}_b\left(\frac{1}{M+1}, \ldots, \frac{1}{M+1}\right). \tag{3.26}$$

8. *For positive integers* $N_1, \ldots, N_K$ *where* $N_1 + \cdots + N_K = M$, $\mathscr{H}_b$ *satisfies*

$$\mathscr{H}_b\left(\frac{1}{M}, \ldots, \frac{1}{M}\right) = \mathscr{H}_b\left(\frac{N_1}{M}, \ldots, \frac{N_K}{M}\right) + \sum_{k=1}^{K} \frac{N_k}{M} \mathscr{H}_b\left(\frac{1}{N_k}, \ldots, \frac{1}{N_k}\right). \tag{3.27}$$

9. $\mathscr{H}_b(x,y) \leq \mathscr{H}_b(x) + \mathscr{H}_b(y)$ *with equality if and only if the random variables* x *and* y *are independent.*
10. $\mathscr{H}_b(y|x) \leq \mathscr{H}_b(y)$ *with equality if and only if the random variables* x *and* y *are independent.*
11. $\mathscr{H}_b(x,y) = \mathscr{H}_b(x) + \mathscr{H}_b(y|x)$ *where* $\mathscr{H}_b(y|x) = -\mathrm{E}_{\pi(x,y)}(\log \pi(y|x))$.

3.5 Diversity and entropy

In biological studies, the diversity of the population is intended as a quantitative measure which reflects how many different types (such as species) exist in the population (Ricotta & Szeidl, 2006; Izsák, 2007; Butturi-Gomes et al., 2014). Consider a population composed of M different types. Let π_m be the relative abundance of the mth type, such that $0 \leq \pi_m \leq 1$ and $\sum_{m=1}^{M} \pi_m = 1$. A widely used measure of diversity is defined by

$$\mathscr{D}_q(\boldsymbol{\pi}) = \begin{cases} \left(\sum_{m=1}^{M} \pi_m^q\right)^{1/(1-q)} & \text{if } q \neq 1 \\ 1 / \prod_{m=1}^{M} \pi_m^{\pi_m} & \text{if } q = 1 \end{cases} \tag{3.28}$$

where q is a real number representing the order of the measure. From Eq. (3.28), it is possible to see that the order of $\mathscr{D}_q$ indicates its sensitivity to common and rare species. $\mathscr{D}_0$ is a measure of diversity completely insensitive to species abundances, it simply quantifies how many different types the population contains (the effective number of species). Each $\mathscr{D}_q$ with value of q greater than unity provides a diversity that disproportionately favor the most common species (the weight given to abundant species is exaggerated), while $\mathscr{D}_q$ with value of q less than unity provides a diversity that disproportionately favor the rare species.

The critical point that weighs all species by their relative abundances without favoring either common or rare species occurs when $q = 1$. In addition, note that $\mathscr{D}_1$ is the limit of $\mathscr{D}_q$ when $q \to 1$. In fact, it can be observed that

$$\sum_{m=1}^{M} \pi_m^{1+\varepsilon} = \sum_{m=1}^{M} \pi_m e^{\varepsilon \cdot \ln \pi_m} \approx 1 + \varepsilon \sum_{m=1}^{M} \pi_m \ln \pi_m. \tag{3.29}$$

Hence,

$$\lim_{\varepsilon\to 0} \mathscr{D}_{1+\varepsilon}(\boldsymbol{\pi}) = \lim_{\varepsilon\to 0} \left(1+\varepsilon \sum_{m=1}^{M} \pi_m \ln \pi_m\right)^{-1/\varepsilon} = e^{\mathscr{H}(\boldsymbol{\pi})} = \mathscr{D}_1(\boldsymbol{\pi}) \tag{3.30}$$

where $\mathscr{H}(\boldsymbol{\pi}) = -\sum_m \pi_m \ln \pi_m$ is the Shannon's entropy associated with the distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_M)$ of relative abundances. It follows from Eq. (3.30) that the Shannon's entropy is a monotonic function of $\mathscr{D}_1$. In fact,

$$\mathscr{H}_b(\boldsymbol{\pi}) = (\log_b e) \cdot \ln \mathscr{D}_1(\boldsymbol{\pi}) \qquad \text{for all } 0 < b \neq 1. \tag{3.31}$$

Therefore, $\mathscr{H}_b$ can be considered as an index of diversity. The Shannon's entropy takes into account simultaneously the number of different types that exist in the population and how the individuals within the population are distributed among these types. It increases both when the number of types increases and also when the uniformity of the distribution of individuals increases as well. For a given number of types, the value of the Shannon's entropy is maximized when all types are equally abundant.

3.6 Entropy estimation

In practice, $\boldsymbol{\pi}$ (the abundances of the different types) is unknown, then $\mathscr{H}$ and π_m (for all m) need to be estimated from observed cell counts $y_m \geq 0$ (observed data). Therefore, this section focuses on the entropy estimation problem, with special emphasis on the small-sample case. A particularly simple and widely used estimator of entropy is the maximum likelihood (ML) estimator

$$\hat{\mathscr{H}}^{ML} = -\sum_{m=1}^{M} \hat{\pi}_m^{ML} \log_b \hat{\pi}_m^{ML} \tag{3.32}$$

constructed by plugging the ML frequency estimates

$$\hat{\pi}_m^{ML} = \frac{y_m}{n} \tag{3.33}$$

into Eq. (3.24), with $n = \sum_m y_m$ being the total number of counts. The connection between y_m and π_m is given by the multinomial distribution:

$$L(\boldsymbol{\pi}|\mathbf{y}) = \Pr(\mathbf{y}|\boldsymbol{\pi}) = \frac{n!}{y_1! \cdot \ldots \cdot y_M!} \prod_{m=1}^{M} \pi_m^{y_m}. \tag{3.34}$$

The ML estimator of π_m maximizes $L(\boldsymbol{\pi}|\mathbf{y})$ (the likelihood function) for y_m fixed, leading to the observed frequencies defined in Eq. (3.33) with

$$\mathrm{E}(\hat{\pi}_m^{ML}) = \pi_m \qquad \mathrm{Var}(\hat{\pi}_m^{ML}) = (n-1)^{-1}\pi_m(1-\pi_m) \to 0 \tag{3.35}$$

when $n \to \infty$. Fig. 3.1 shows a graphical representation of the ML estimator (in red) of $\pi_m = p(m|M,p) = C_{M,m}p^m(1-p)^{M-m}$, $m = 0,1,\ldots,M$ (in black), with $M = 100$, $p = 0.3$, and $C_{M,m} = M!/[m!(M-m)!]$, while the sample size varies from $n = 4$ to $n = 1024$.

In general, when $n \gg M$, it is easy to infer the entropy reliably and it is well-known that in this case the ML estimator is optimal. However, when $n \ll M$ or $n \approx M$ (especially in high-dimensional problems), it becomes extremely challenging to estimate the entropy. In the last case, the ML estimator performs very poorly and severely underestimates the true entropy. This drawback has lead to new approaches to the small-sample entropy estimation problem. Nemenman, Shafee & Bialek (2002) introduced a Bayesian entropy estimator (NSB estimator) by using a Dirichlet mixture prior with infinite number of components. This approach resulted in an estimator nearly unbiased and with remarkably good statistical properties. However, the NSB estimator is computationally expensive and sometimes slow for practical applications. Another proposed estimator is due to Chao & Shen (2003). This approach combines the Horvitz-Thompson entropy estimator (Horvitz & Thompson, 1952) with the Good-Turing correction (Good, 1953; Orlitsky, Santhanam & Zhang, 2003) of the empirical cell frequencies. The Chao-Shen entropy estimator is nearly unbiased and statistically very efficient with small mean squared error (MSE) regardless of sample size. Hausser & Strimmer (2009) introduced an entropy estimator that employs James-Stein-type shrinkage at the level of cell frequencies. This leads to an entropy estimator that is highly effective, both in terms of statistical accuracy and computational complexity. James-Stein-type shrinkage is a simple analytic method

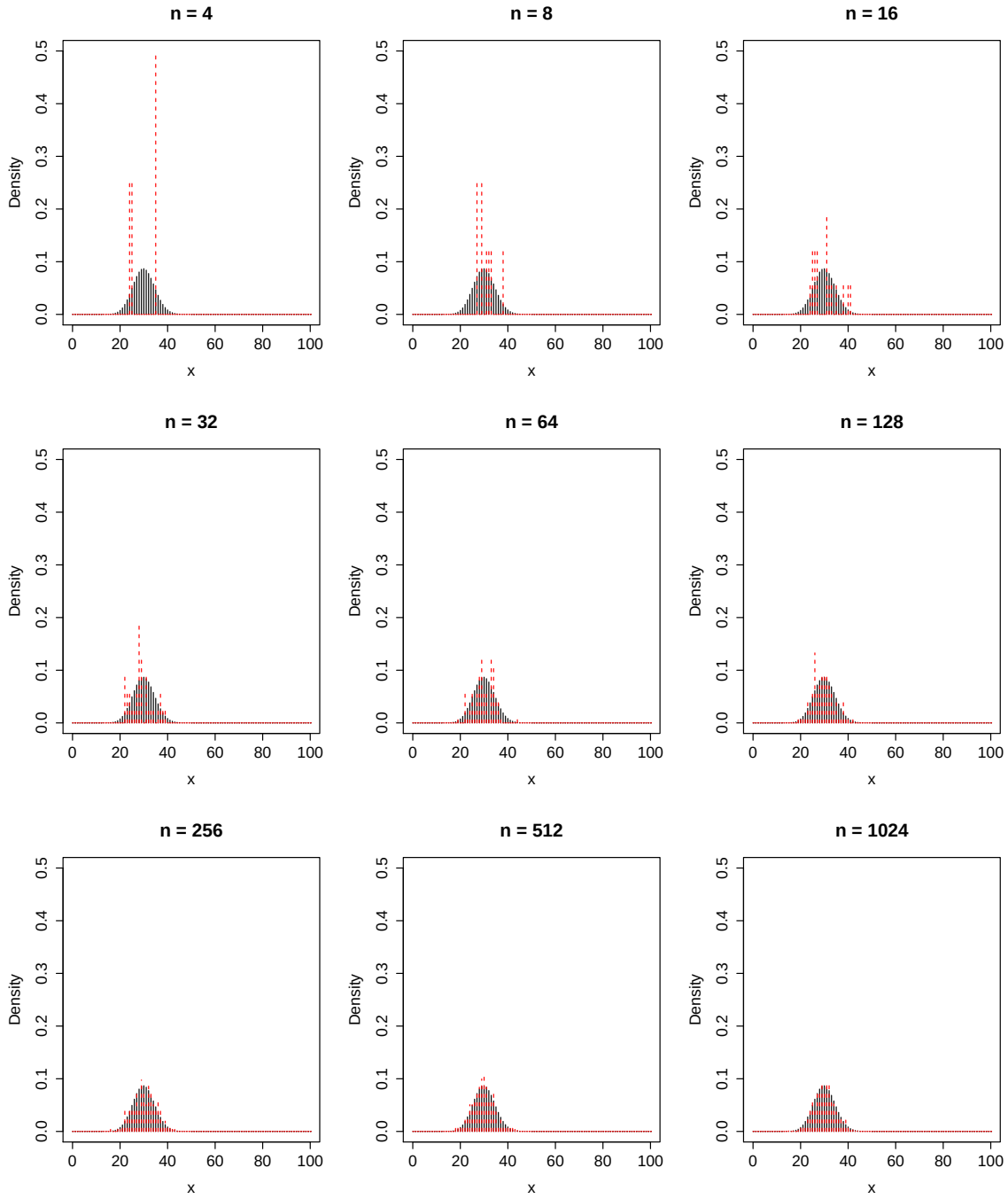

Figure 3.1: MLE (red) of $\boldsymbol{\pi} \sim p(m|100, 0.3)$ (black) while n varies from 4 to 1024.

to perform regularized high-dimensional inference. It is based on averaging two very different models: a high-dimensional model with low bias and high variance and a lower-dimensional model with larger bias but smaller variance. The intensity of the regularization is determined by the relative weighting of the two models. A general description for constructing shrinkage estimators is given by Hausser & Strimmer (2009, Appendix A). Following this description, the problem of estimating cell frequencies is addressed by considering the following convex combination:

$$\hat{\pi}_m^S = \xi \hat{\beta}_m + (1 - \xi)\hat{\pi}_m^{ML} \tag{3.36}$$

where $\xi \in [0, 1]$ is the shrinkage intensity and $\hat{\beta}_m$ is the shrinkage target. A convenient choice of $\hat{\beta}_m$ is the uniform

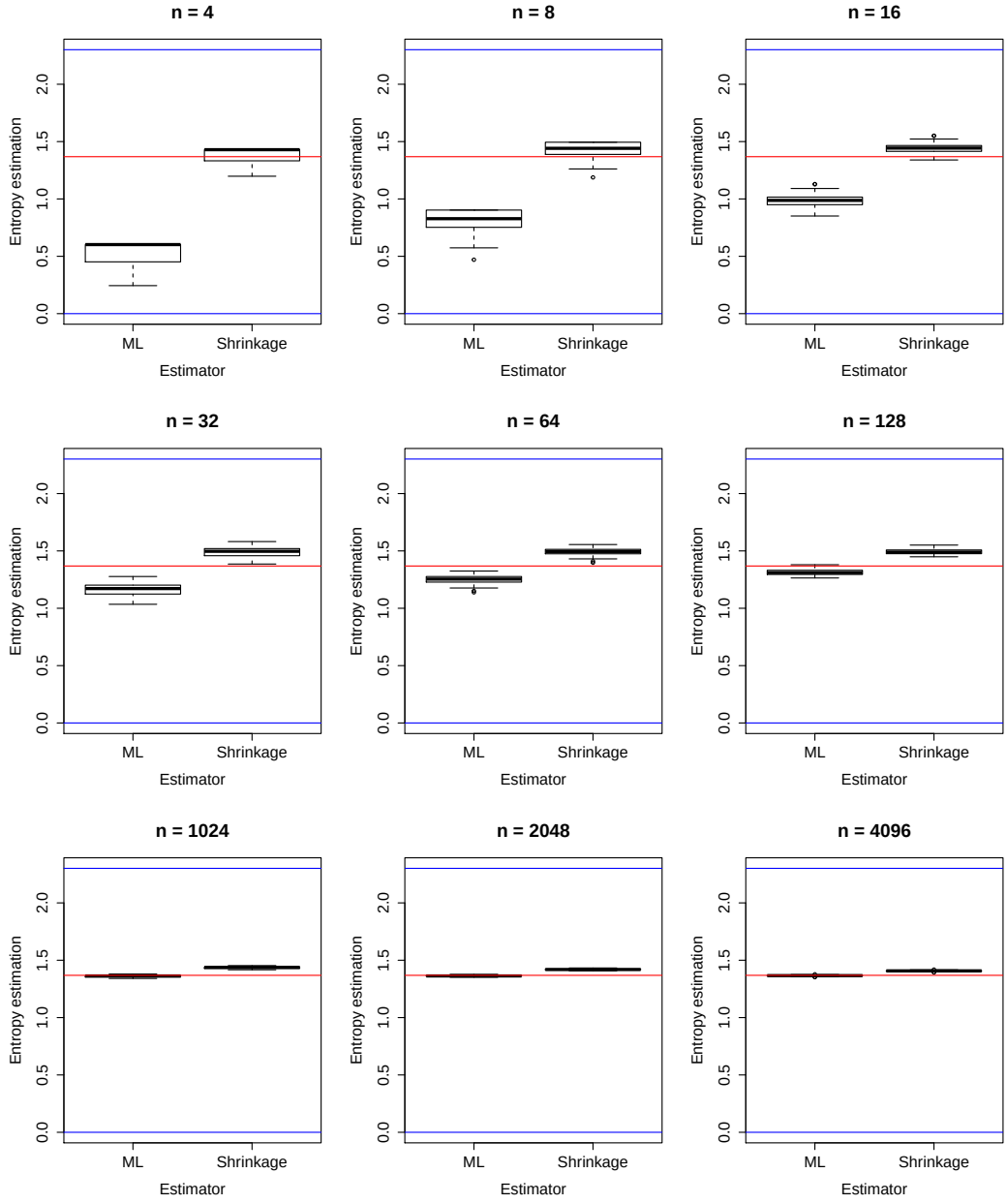

Figure 3.2: $\hat{\mathscr{H}}^{ML}$ versus $\hat{\mathscr{H}}^{S}$ for $\boldsymbol{\pi} \sim p(m|200, 0.2)$ while n varies from 4 to 4096.

distribution, $\hat{\beta}_m = M^{-1}$ for all m. This is the distribution with maximum entropy. Considering the statistical properties of $\hat{\pi}_m^{ML}$, the estimated shrinkage intensity is given by

$$\xi = \frac{\sum_{m=1}^{M} \mathrm{Var}(\hat{\pi}_m^{ML})}{\sum_{m=1}^{M} \mathrm{E}[(\hat{\pi}_m^{ML} - \hat{\beta}_m)^2]} = \frac{1 - \sum_{m=1}^{M} \hat{\pi}_m^{ML}}{(n-1)\sum_{m=1}^{M} (\hat{\beta}_m - \hat{\pi}_m^{ML})^2} \tag{3.37}$$

and the resulting plugin shrinkage entropy estimator is given by

$$\hat{\mathscr{H}}^{S} = -\sum_{m=1}^{M} \hat{\pi}_m^{S} \log_b \hat{\pi}_m^{S}. \tag{3.38}$$

Fig. 3.2 shows a comparison between $\hat{\mathscr{H}}^{ML}$ and $\hat{\mathscr{H}}^{S}$ for $\boldsymbol{\pi} \sim \pi_m = p(m|M,p) = C_{M,m} p^m (1-p)^{M-m}$, $m = 0, 1, \ldots, M$, with $M = 200$ and $p = 0.2$, while the sample size varies from 4 to 4096. Similar results are also

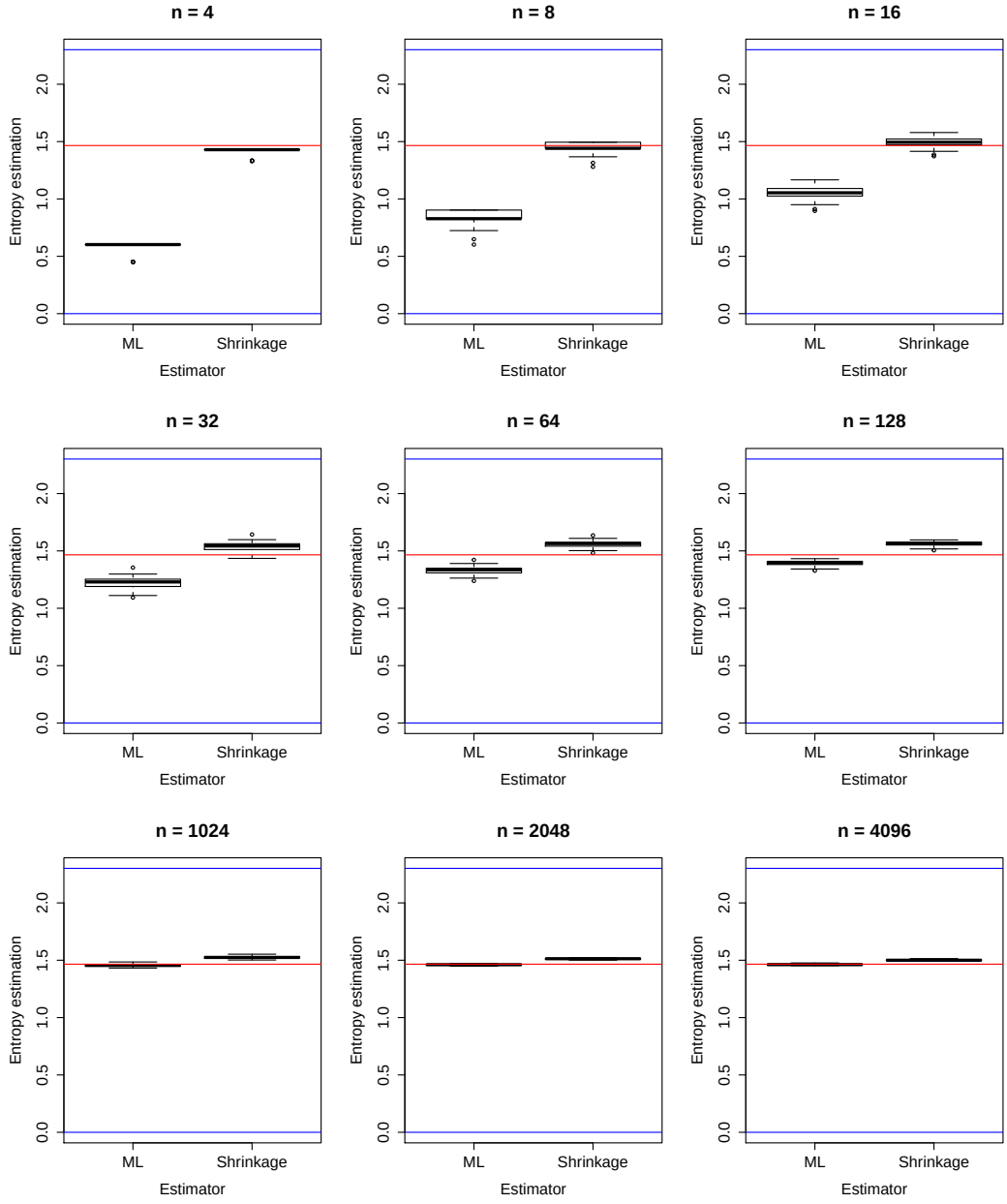

Figure 3.3: $\hat{\mathscr{H}}^{ML}$ versus $\hat{\mathscr{H}}^{S}$ for $\boldsymbol{\pi} \sim p(m|200, 0.5)$ while n varies from 4 to 4096.

presented in Figs. 3.3 and 3.4, for $p = 0.5$ and $p = 0.8$ respectively. The red line represents the exact value of the entropy of $\boldsymbol{\pi}$ (the target value) given by:

$$\mathscr{H}(\boldsymbol{\pi}) = -\sum_{m=1}^{M} C_{M,m} p^m (1-p)^{M-m} \log_{10}\left[C_{M,m} p^m (1-p)^{M-m}\right] = \frac{\log_{10}[(2e\pi)Mp(1-p)]}{2} + O(1/M). \quad (3.39)$$

When $p = 0.2$ or $p = 0.8$, then $\mathscr{H}(\boldsymbol{\pi}) = 1.36840$. When $p = 0.5$, then $\mathscr{H}(\boldsymbol{\pi}) = 1.46572$. The blue lines represent the values $\mathscr{H}^{\min} = 0$ and $\mathscr{H}^{\max} = \log_{10} M = \log_{10} 200 = 2.30103$.

The NSB, Chao-Shen, and shrinkage estimators are statistically very efficient with small MSEs regardless of sample size. However, in terms of versatility, the shrinkage estimator has two distinct advantages over the NSB and Chao-Shen estimators. First, in addition to estimating the entropy, it also provides the underlying multinomial frequencies for use with the Shannon entropy. This is useful in the context of using mutual information to quantify

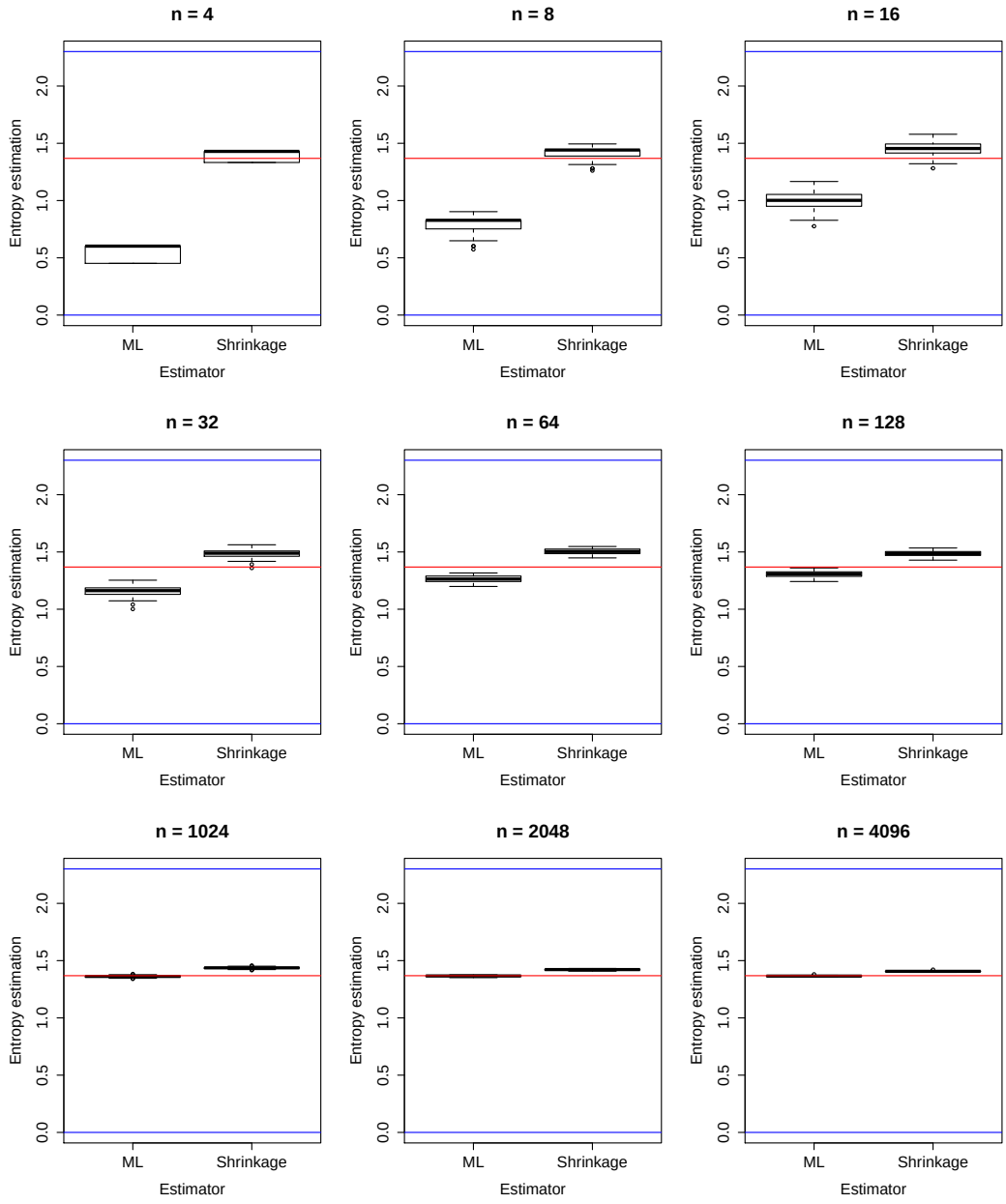

Figure 3.4: $\hat{\mathscr{H}}^{ML}$ versus $\hat{\mathscr{H}}^{S}$ for $\boldsymbol{\pi} \sim p(m|200, 0.8)$ while n varies from 4 to 4096.

nonlinear pairwise dependencies for instance. Second, unlike the NSB estimator, the shrinkage estimator is a full analytic estimator. Also note that the relationship between diversity and entropy will be explored in Chapter 4 by defining the Shannon's entropy as a phenotypic diversity index associated with a population of particles during the optimization process. The estimation procedure developed here will be used to estimate the Shannon's index of diversity that will be used to monitor the population diversity, having an important role in the mechanism to maintain or introduce diversity during the search process.

3.7 Ranking decision alternatives

A decision matrix is a multivariate data set with m rows and n columns, where rows represent decision alternatives to be selected and columns represent evaluation criteria of interest to be considered in this decision problem of

finding the best alternative (from the set of decision alternatives) based on evaluation criteria. Each row of a decision matrix records the performance rating under different evaluation criteria for a given alternative and each column records the performance rating for different alternatives under a given criterion. Formally, a decision matrix can be expressed as follows:

$$\mathbf{D} = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{cccc} \langle C_1, w_1 \rangle & \langle C_2, w_2 \rangle & \cdots & \langle C_n, w_n \rangle \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}_{m \times n} \end{array} \tag{3.40}$$

where $A_1, A_2, \ldots, A_m$ are feasible alternatives, $C_1, C_2, \ldots, C_n$ are evaluation criteria, x_{ij} is the performance rating of alternative A_i under criterion C_j, and w_j is the weight of criterion C_j, satisfying $\sum_j w_j = 1$. Evaluation criteria can be classified into two types: benefit and cost. A benefit criterion means that a larger value is more valuable, while a cost criterion is just the reverse.

Given a decision matrix, the main objective is to find the best alternative from the set of feasible alternatives, taking into account the set of evaluation criteria of this decision problem. This problem is commonly referred to as a multi-criteria decision making (MCDM) problem. Technique for order preference by similarity to ideal solution (TOPSIS) was first developed by Hwang & Yoon (1981) as an effective ranking method for solving MCDM problems. This method uses two ideal solutions as reference. One is the positive ideal solution (PIS), which minimizes the cost criteria and maximizes the benefit criteria simultaneously. The other is the negative ideal solution (NIS), which maximizes the cost criteria and minimizes the benefit criteria simultaneously. TOPSIS is based upon the principle that the chosen alternative should have the shortest distance from the PIS and the farthest distance from the NIS. It can be described as follows. The data of a decision matrix come from different sources so, in general, it is necessary to normalize them in order to obtain a dimensionless matrix. The normalized value r_{ij} can be calculated as

$$r_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}} \qquad i = 1, \ldots, m; j = 1, \ldots, n \tag{3.41}$$

or

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \qquad i = 1, \ldots, m; j = 1, \ldots, n. \tag{3.42}$$

The normalized decision matrix $\mathbf{D}_n = (r_{ij})_{m \times n}$ represents the relative performance of the alternatives. After normalization, the weighted normalized decision matrix $\mathbf{D}_w = (p_{ij})$ is calculated, where the weighted normalized value p_{ij} is given as

$$p_{ij} = w_j r_{ij} \qquad i = 1, \ldots, m; j = 1, \ldots, n. \tag{3.43}$$

Define the positive ideal solution $\mathbf{p}^+$ and the negative ideal solution $\mathbf{p}^-$ as

$$\text{PIS} = \mathbf{p}^+ = (p_1^+, \ldots, p_n^+)' \qquad p_j^+ = \begin{cases} \min\{p_{1j}, \ldots, p_{mj}\} & \text{if } C_j \text{ is a cost criterion} \\ \max\{p_{1j}, \ldots, p_{mj}\} & \text{if } C_j \text{ is a benefit criterion} \end{cases} \tag{3.44}$$

and

$$\text{NIS} = \mathbf{p}^- = (p_1^-, \ldots, p_n^-)' \qquad p_j^- = \begin{cases} \max\{p_{1j}, \ldots, p_{mj}\} & \text{if } C_j \text{ is a cost criterion} \\ \min\{p_{1j}, \ldots, p_{mj}\} & \text{if } C_j \text{ is a benefit criterion.} \end{cases} \tag{3.45}$$

Calculate the distances from $\mathbf{p}^+$ and $\mathbf{p}^-$ of each alternative, respectively as

$$d_i^+ = \sqrt{\sum_{j=1}^{n} (p_j^+ - p_{ij})^2} \qquad i = 1, \ldots, m \tag{3.46}$$

and

$$d_i^- = \sqrt{\sum_{j=1}^{n} (p_j^- - p_{ij})^2} \qquad i = 1, \ldots, m. \tag{3.47}$$

Calculate the relative closeness of each alternative with respect to the positive ideal solution as

$$RC_i = \frac{d_i^-}{d_i^- + d_i^+} \qquad i = 1, \ldots, m. \tag{3.48}$$

Rank the alternatives according to the relative closeness. The best alternatives are those that have a higher value RC_i and therefore should be chosen because they are closer to the positive ideal solution. That is

$$A_{(m)} \prec \cdots \prec A_{(1)} \text{ since } RC_{(m)} \leq \cdots \leq RC_{(1)} \tag{3.49}$$

where $RC_{(m)} \leq \cdots \leq RC_{(1)}$ represents an increasing ordering of $RC_1, \cdots, RC_m$ and $A_{(i)} \prec A_{(i')}$ informs that $A_{(i')}$ is a better alternative than $A_{(i)}$.

This section ends by describing a strategy to define the weight vector based on the amount of information associated with each evaluation criterion (see Huang, 2008). This strategy can be established when

$$\sum_{i=1}^{m} r_{ij} = 1 \tag{3.50}$$

which is the case when r_{ij} is calculated, for example, by Eq. (3.41). In this case, $(r_{1j}, \ldots, r_{mj})'$ represents a probability distribution when j is fixed. The amount of uncertainty contained in the criterion C_j can be measured by the entropy associated with this criterion, which in turn can be calculated as

$$\mathscr{H}_j = -\frac{1}{\ln(m)} \sum_{i=1}^{m} r_{ij} \ln(r_{ij}) \qquad j = 1, \ldots, n. \tag{3.51}$$

Therefore, the amount of decision information contained in criterion C_j can be calculated as

$$\mathscr{I}_j = 1 - \mathscr{H}_j \qquad j = 1, \ldots, n \tag{3.52}$$

and the weight for each criterion is given by

$$w_j = \frac{\mathscr{I}_j}{\sum_{j=1}^{n} \mathscr{I}_j} \qquad j = 1, \ldots, n. \tag{3.53}$$

The criteria associated with higher weights influence more in the decision making process.

Chapter 4

Entropy-Based Bare Bones Particle Swarm for Dynamic Constrained Optimization

Dynamic constrained optimization problems (DCOPs) were discussed in detail in Chapter 2. This chapter is devoted to presenting a new algorithmic methodology for solving DCOPs. The proposed algorithm explores the principles of swarm computation jointly with the concepts and methods described in Chapter 3 to design an approach for dealing with DCOPs. The approach is endowed with different mechanisms that are described in the context of dynamic optimization. Results of experiments designed to evaluate the performance of the proposed algorithm are presented and discussed in Chapter 5 along with experimental results found by other algorithms for comparison purposes.

4.1 Introduction

This chapter presents an algorithmic methodology for solving DCOPs. The proposed algorithm is named as entropy-based bare bone particle swarm (EBBPSO-T to be short) and it is endowed with the following mechanisms:

1. A mechanism to monitor the population diversity that was defined as the entropy of a distribution of fitness values.
2. An entropy-based dynamic rule for governing the search of each particle. This rule works as a scheme to maintain or introduce diversity during the search process.
3. A neighborhood structure for information exchange between particles.
4. A constraint-handling method that deal with the objective function and the constraints separately using a ranking method with selection based on TOPSIS.
5. A mechanism to detect changes in the environment based on a fixed set of detectors uniformly distributed in the search space. The detectors differs fully from particles that constitute the swarm.
6. A mechanism to update particles' memories based on re-evaluation of fitness values and replacement using a ranking method with selection based on TOPSIS.
7. Finally, a scheme of random immigrants that acts only on the first iteration of a changed environment.

These mechanisms operate related to each other to tackle the inherent difficulties involved in optimization problems in dynamic environments. The combined effect of these mechanisms provides an algorithm capable of maintaining a proper balance between exploration and exploitation at any stage of the search process without losing traceability to search an optimal solution that is changing its position in the search space over time.

The mechanism that monitors the population diversity operates continuously throughout the optimization process. The Shannon's entropy is established as a phenotypic diversity index and the proposed algorithm uses the Shannon's index to aggregate two variants of the bare bones particle swarm optimizer (BBPSO), namely: the global-best and local-best BBPSO. The idea of mixture of search directions is used considering the index of diversity as a factor to balance the influence of the global-best and local-best search directions. High diversity promotes the search guided by global-best solution with a normal distribution for exploitation. Low diversity promotes the search guided by local-best solution with a heavy-tailed distribution for exploration. This strategy works as a scheme to maintain or introduce diversity during the search process that is continuously affected by the mechanism that monitors the diversity.

The proposed constraint-handling strategy uses a ranking method combined with TOPSIS to obtain the best solution within a population of candidate solutions $\mathbf{P}'$. The particles (or candidate solutions) represent the alternatives and all the criteria are cost criteria to be minimized that are related to numerical values of the objective function and the degree of constraint violation. When a new environment is established, the search is divided into two phases. The first phase aims to find feasible solutions, regardless of the objective function value. After an appropriate number of feasible solutions has been found, the second phase starts with the goal of optimizing the objective function. To this aim, the parameters of TOPSIS are defined depending on whether or not there is at least one feasible solution in $\mathbf{P}'$. Note that in the initial iterations of a changed environment, it is likely that TOPSIS has to manipulate a population of particles composed entirely of infeasible solutions.

When a change in the environment is detected, this information is used to trigger actions in order to react to the change. These actions occur sequentially as follows: (1) the random immigrants mechanism operates and a subset of particles of the swarm is replaced by particles randomly generated (this mechanism acts only on the first iteration of the changed environment); (2) the numerical properties (such as $f(\mathbf{x},t)$ and $g_i(\mathbf{x},t)$ for $i \in I$) of each particle are updated according to the new environment; and (3) the mechanism to update particles' memories acts using a ranking method combined with TOPSIS to select the best particle (or solution) within a population of particles associated with each specific case.

It is important to emphasize that significant diversity loss can be generated when constraint-handling strategies treat the objective function and the constraints separately as different criteria in the selection mechanism of the best particle within a specific population (Mezura-Montes & Coello, 2011). In this case, it is important to continuously monitor the population diversity and use this information to trigger mechanisms to maintain and introduce diversity in order to combat this problem. The following sections discuss in detail the proposed algorithm.

4.2 Swarm structure

The structure of the EBBPSO-T is equivalent to the structure of the canonical BBPSO. BBPSO and its variants are presented in Section 1.1 and Appendix B for solving unconstrained optimization problems (UOPs) in static environments. EBBPSO-T has a swarm $\mathbf{P}$ with K particles and a neighborhood system $\mathscr{N}$ defined by a ring topology with $|\mathscr{N}_k|$ neighbors to each particle. Each particle is characterized by a vector $(\mathbf{x}_k, \mathbf{p}_k, \mathbf{n}_k, \mathbf{g})'$, where $\mathbf{x}_k = (x_{k1}, \ldots, x_{kD})'$ denotes the position of the particle, $\mathbf{p}_k$ and $\mathbf{n}_k$ denote respectively the personal-best and local-best positions (pbest and lbest), and $\mathbf{g}$ denotes the global-best position (gbest). In addition, each position $\mathbf{x}_k$ is associated with the following numerical properties:

1. $f(\mathbf{x}_k,t)$: objective function value of $\mathbf{x}_k$ at time t.
2. $g_i(\mathbf{x}_k,t)$: constraint function value of $\mathbf{x}_k$ at time t.
3. $s(\mathbf{x}_k,t) = \sum_{i=1}^{I} [\max\{0, g_i(\mathbf{x}_k,t)\}]^2$: degree of constraint violation related to $\mathbf{x}_k$ at time t.
4. $u(\mathbf{x}_k,t) = |\{i : g_i(\mathbf{x}_k,t) > 0\}|$: number of constraints violated by $\mathbf{x}_k$ at time t.

Since the memories $\mathbf{p}_k, \mathbf{n}_k$, and $\mathbf{g}$ represent positions such as $\mathbf{x}_k$, it is obvious that each of these memories also has the properties (f, g_i, s, u).

4.3 Index of diversity

The Shannon's entropy was discussed in Chapter 3 (see Section 3.5) as an index of diversity. This index can be applied in monitoring the diversity of a population of particles during an optimization process. Consider a swarm $\mathbf{P}$

with K particles along with the set of variables and numerical properties associated with each particle. The entropy estimation is performed based on a set of fitness values, for example, the set of values related to $f(\mathbf{x},t)$ and $f(\mathbf{p},t)$. The approach is described in the following steps:

Step 1. The data related to $f(\mathbf{x},t)$ must be discretized, with each measurement assuming one of M_1 states. An analogous procedure must be done with the data related to $f(\mathbf{p},t)$, with each measurement assuming one of M_2 states.

Step 2. $M = M_1 \times M_2$ cell frequencies are estimated for the pair of variables $f(\mathbf{x},t)$ and $f(\mathbf{p},t)$, using the shrinkage approach defined by Eq. (3.36) (see Chapter 3). Note that typically K (here representing the sample size) is much smaller than M, thus simple approaches such as ML estimation should be avoided.

Step 3. Finally, the Shannon's index of diversity is estimated using the plugin shrinkage entropy estimator $\hat{\mathscr{H}}^S$, defined in Eq. (3.38).

Large values of $\hat{\mathscr{H}}^S$ are related with small values of $\hat{\pi}_m^S$ for nearly every state, i.e., the fitness values associated with the particles are evenly distributed over practically all states. On the other hand, small values of $\hat{\mathscr{H}}^S$ are related with large values of $\hat{\pi}_m^S$ for a few states, i.e., the fitness values associated with the swarm of particles are concentrated in few states. Therefore, high entropy indicates high diversity in the swarm and low entropy indicates low diversity in the swarm. The essential steps of the shrinkage entropy estimation procedure can be summarized as the pseudo code shown in Algorithm 2.

Algorithm 2 ShrinkageEntropyEstimation() procedure.

Input: $f(\mathbf{x}_k,t), f(\mathbf{p}_k,t)$ for all k, M_1, and M_2.

1: Discretize the observations related to $f(\mathbf{x},t)$ and $f(\mathbf{p},t)$ into $M = M_1 \times M_2$ bins
2: Establish the $M_1 \times M_2$ table of counts
3: Calculate $\hat{\pi}_m^S$, $m = 1,\ldots,M$, according to Eq. (3.36)
4: Calculate $\hat{\mathscr{H}}^S$ according to Eq. (3.38)
5: **return** The shrinkage entropy estimate $\hat{\mathscr{H}}^S$.

The Shannon's index of diversity differs from the traditional approach used in swarm computation, where the diversity of a population with K individuals is measured by:

$$\text{Diversity} = \frac{1}{nf \cdot K} \sum_{k=1}^{K} \left(\sum_{d=1}^{D} (x_{kd} - \bar{x}_d)^2 \right)^{1/2} \tag{4.1}$$

where $\bar{\mathbf{x}} = (\bar{x}_1,\ldots,\bar{x}_D) = K^{-1} \sum_{k=1}^{K} \mathbf{x}_k$ and nf is a normalization factor (the maximum distance between opposite corners of the landscape or the diameter of the swarm are examples of factors that can be used for normalization). The measure given by Eq. (4.1) is a genotypic diversity measure, because it monitors the diversity of solutions. The Shannon's index is a phenotypic diversity measure, because it monitors the diversity of fitness values, i.e., the diversity of solution responses. The reader is referred to Olorunda & Engelbrecht (2008) and Corriveau et al. (2012) that provide an overview of genotypic diversity measures for real-coded representations.

4.4 Dynamic rule

EBBPSO-T explores three concepts to define the dynamic rule to update the position of a particle: (1) the neighborhood system to exchange information between particles; (2) heavy-tailed distributions to encourage exploration in the search space and diversify the solutions; and (3) the Shannon's entropy as a phenotypic diversity index.

In swarm optimization, it is well-known that the global-best model converges faster than the local-best model, since all particles are attracted by the same best position. Therefore, the global-best model is distinguished for its exploitation ability. On the other hand, the local-best model has better exploration properties, since the information regarding the best position of each neighborhood is gradually communicated to the rest of the particles through their neighborhood system. Therefore, the attraction toward a specific point is weaker, thus preventing the swarm to get trapped in suboptimal solutions.

The main idea is to apply the Shannon's index of diversity as an unification factor to aggregate the global-best and local-best BBPSO variants, combining their exploitation and exploration properties. Low diversity encourages

the search guided by the local-best solution, using a heavy-tailed distribution for exploration. High diversity encourages the search guided by the global-best solution, using a normal distribution for exploitation. The Shannon's index balances the influence of these search directions in a single dynamic rule to update the position of a particle. The proposed dynamic rule combines the diversification and intensification properties and works as a mechanism for introducing or maintaining diversity.

The first strategy defines the following search direction:

$$\mathbf{x}_k^{(1)} = \boldsymbol{\eta}_k + \boldsymbol{\rho}_k \odot \mathbf{z}_1 \tag{4.2}$$

where

- $\boldsymbol{\eta}_k = (\eta_{k1}, \ldots, \eta_{kD})' = \frac{1}{2}(p_{k1} + g_1, \ldots, p_{kD} + g_D)'$
- $\boldsymbol{\rho}_k = (\rho_{k1}, \ldots, \rho_{kD})' = (|p_{k1} - g_1|, \ldots, |p_{kD} - g_D|)'$
- $\mathbf{z}_1 = (z_{11}, \ldots, z_{1D})' \sim \left(N(0,1), \overset{iid}{\ldots}, N(0,1)\right)'$.

In Eq. (4.2), $\odot$ represents the componentwise vector multiplication. The symbol $\sim$ indicates that z_1 is a vector of independent and identically distributed random variables, where each component has a normal distribution with mean 0 and variance 1. The pdf of $x_{kd}^{(1)}$ is given by

$$x_{kd}^{(1)} \sim N(x_{kd}^{(1)} | \eta_{kd}, \rho_{kd}^2) \tag{4.3}$$

for all $d = 1, \ldots, D$, i.e., $x_{kd}^{(1)}$ has a normal distribution with mean η_{kd} and variance ρ_{kd}^2.

The second strategy defines the following search direction:

$$\mathbf{x}_k^{(2)} = \boldsymbol{\mu}_k + \boldsymbol{\lambda}^{-1/2} \odot \boldsymbol{\sigma}_k \odot \mathbf{z}_2 \qquad \boldsymbol{\lambda} \sim \left(Ga\left(\frac{\nu}{2}, \frac{\nu}{2}\right), \overset{iid}{\ldots}, Ga\left(\frac{\nu}{2}, \frac{\nu}{2}\right)\right)' \tag{4.4}$$

where

- $\boldsymbol{\mu}_k = (\mu_{k1}, \ldots, \mu_{kD})' = \frac{1}{2}(p_{k1} + n_{k1}, \ldots, p_{kD} + n_{kD})'$
- $\boldsymbol{\sigma}_k = (\sigma_{k1}, \ldots, \sigma_{kD})' = (|p_{k1} - n_{k1}|, \ldots, |p_{kD} - n_{kD}|)'$
- $\mathbf{z}_2 = (z_{21}, \ldots, z_{2D})' \sim \left((N(0,1), \overset{idd}{\ldots}, N(0,1)\right)'$
- $\boldsymbol{\lambda}^{-1/2} = (\lambda_1^{-1/2}, \ldots, \lambda_D^{-1/2})'$.

Each $\lambda_d \sim Ga(\frac{\nu}{2}, \frac{\nu}{2})$ is independent of $z_{2d} \sim N(0,1)$. The distribution of $x_{kd}^{(2)}$ can be expressed hierarchically as

$$x_{kd}^{(2)} | \lambda_d \sim N(x_{kd}^{(2)} | \mu_{kd}, \lambda_d^{-1} \sigma_{kd}^2) \qquad \lambda_d \sim Ga(\lambda_d | \nu/2, \nu/2) \tag{4.5}$$

and its pdf is given by

$$x_{kd}^{(2)} \sim t(x_{kd}^{(2)} | \nu, \mu_{kd}, \sigma_{kd}^2) \tag{4.6}$$

for all $d = 1, \ldots, D$, i.e., $x_{kd}^{(2)}$ has a t-distribution with location μ_{kd}, scale σ_{kd}, and ν degrees of freedom. The parameter ν can be used to control the heaviness of the tails of the distribution of $x_{kd}^{(2)}$.

The aggregation of the global-best and local-best search directions defines the new position of a particle as follows:

$$\mathbf{x}_k = \frac{\hat{\mathscr{H}}^S}{\mathscr{H}_{\max}} \cdot \mathbf{x}_k^{(1)} + \left(1 - \frac{\hat{\mathscr{H}}^S}{\mathscr{H}_{\max}}\right) \cdot \mathbf{x}_k^{(2)} \tag{4.7}$$

where $\hat{\mathscr{H}}^S$ is the Shannon's index of diversity and $\mathscr{H}_{\max}$ is the maximum entropy. Thus, the global-best search strategy is realized with normal distributions and the local-best search strategy is realized with heavy-tailed distributions in their hierarchical forms, as scale mixtures of normal distributions (see Section 3.2 in Chapter 3 for more details). Small values of $\hat{\mathscr{H}}^S$ favor the local-best search for exploration. Large values of $\hat{\mathscr{H}}^S$ favor the global-best search for exploitation.

It is important to emphasize that entropy has been used in swarm and evolutionary optimization as an index of diversity (see Petalas, Parsopoulos & Vrahatis, 2007; Liu, Mernik & Bryant, 2007, 2009). Petalas, Parsopoulos & Vrahatis (2007) introduced a memetic PSO that combines the PSO with a local search method for computing periodic orbits of nonlinear mappings. Liu, Mernik & Bryant (2007, 2009) introduced an entropy-driven evolutionary approach for solving UOPs in static environments. Our approach differs from the approaches used by Petalas, Parsopoulos & Vrahatis (2007) and Liu, Mernik & Bryant (2007, 2009) in the following aspects: (1) the approach used by Petalas, Parsopoulos & Vrahatis (2007) employs the Shannon's entropy for deciding when the local search method is applied to exploit the best position of each particle. This decision depends on two user-defined threshold values. The approach used by Liu, Mernik & Bryant (2007, 2009) applies an user-defined threshold value to switch (whenever necessary) from a low mutation rate to a high mutation rate and vice versa depending on whether or not there is a degradation of the Shannon's entropy (index of diversity). Our approach does not require any threshold value to switch the search strategy between two regimes, for example, from global search to local search. Instead, our approach uses the idea of mixture of search directions, having the Shannon's index as a factor to balance the influence of these search directions; (2) the approaches used by Petalas, Parsopoulos & Vrahatis (2007) and Liu, Mernik & Bryant (2007, 2009) do not make clear how the entropy estimation problem was resolved. It is not clear the estimator that was effectively used in that problem. Instead, our approach suggests an entropy estimator and justifies its application as an index of diversity in swarm optimization.

4.5 Constraint handling

This section proposes a constraint-handling strategy that uses a ranking method with selection based on TOPSIS to obtain the best solution within a population of candidate solutions. The solutions represents the alternatives and all the criteria are cost criteria to be minimized that are related to numerical properties (f,s,u). The idea is divide the search in two phases. The first phase aims to find feasible solutions, regardless of the objective function value. After an appropriate number of feasible solutions has been found, the second phase starts with the goal of optimizing the objective function. The parameters of TOPSIS are defined depending on whether or not there is at least one feasible solution in the population of candidate solutions. The proposed approach is described as follows. Since the numerical properties f,s, and u are of different orders of magnitude, a ranking method is used to deal with the proper scaling of these properties. Consider a population of particles $\mathbf{P}'$. The particles in $\mathbf{P}'$ are ranked with respect to each property (f,s,u) independently. This produces three new terms, denoted by R_1,R_2, and R_3, respectively. Clearly this terms are all of the same order of magnitude. This information can be summarized in a decision matrix

$$\mathbf{D} = \begin{array}{c} \\ \text{particle}_1 \\ \\ \text{particle}_m \end{array} \begin{array}{c} \begin{array}{ccc} R_1 & R_2 & R_3 \end{array} \\ \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ \vdots & \vdots & \vdots \\ R_{m1} & R_{m2} & R_{m3} \end{pmatrix} \end{array} \tag{4.8}$$

where the particles are alternatives, R_1,R_2, and R_3 are criteria, and R_{ij} indicates the rank achieved by the i-th particle $(i = 1,\ldots,m)$ according to criteria R_j $(j = 1,2,3)$. The selection strategy is based on TOPSIS, which is described in the following steps:

Step 1. Calculate the normalized decision matrix $\mathbf{D}_n = (r_{ij})$ from $\mathbf{D}$, where the normalized value r_{ij} is usually calculated as:

$$r_{ij} = \frac{R_{ij}}{\sqrt{\sum_{i=1}^{m} R_{ij}^2}} \qquad i = 1,\ldots,m; j = 1,2,3. \tag{4.9}$$

Step 2. After normalization, calculate the weighted normalized decision matrix $\mathbf{D}_w = (p_{ij})$, where the weighted normalized value p_{ij} is calculated as:

$$p_{ij} = w_j r_{ij} \qquad i = 1,\ldots,m; j = 1,2,3. \tag{4.10}$$

The weight vector $w = (w_1,w_2,w_3)'$ satisfies $w_1 + w_2 + w_3 = 1$.

Step 3. Define the positive ideal solution p^+ and the negative ideal solution p^- as:

$$\mathbf{p}^+ = (p_1^+, p_2^+, p_3^+)' \qquad p_j^+ = \min\{p_{1j},\ldots,p_{mj}\} \tag{4.11}$$

and

$$\mathbf{p}^- = (p_1^-, p_2^-, p_3^-)' \qquad p_j^- = \max\{p_{1j}, \ldots, p_{mj}\}. \tag{4.12}$$

Step 4. Calculate the distances from p^+ and p^- of each alternative, respectively as:

$$d_i^+ = \sqrt{\sum_j (p_j^+ - p_{ij})^2} \qquad i = 1, \ldots, m \tag{4.13}$$

and

$$d_i^- = \sqrt{\sum_j (p_j^- - p_{ij})^2} \qquad i = 1, \ldots, m. \tag{4.14}$$

Step 5. Calculate the relative closeness of each alternative with respect to the positive ideal solution as:

$$RC_i = \frac{d_i^-}{d_i^- + d_i^+} \qquad i = 1, \ldots, m. \tag{4.15}$$

Step 6. Rank the alternatives according to the relative closeness. The best alternatives are those that have higher value RC_i and therefore should be chosen because they are closer to the positive ideal solution.

Algorithm 3 RankingTOPSIS() procedure.

Input: $\mathbf{P}'$, a population of solutions.

1: Evaluate each solution with respect to (f, s, u)
2: Calculate the decision matrix $\mathbf{D}$ and define w
3: Calculate the normalized decision matrix $\mathbf{D}_n$
4: Calculate the weighted normalized decision matrix $\mathbf{D}_w$
5: Define $\mathbf{p}^+$ and $\mathbf{p}^-$
6: Calculate d_i^+ and d_i^- of each solution
7: Calculate the relative closeness of each solution
8: Rank the solutions according to the relative closeness
9: **return** The best solution (particle) in $\mathbf{P}'$.

The essential steps of the RankingTOPSIS procedure can be summarized as the pseudo code shown in Algorithm 3. When all particles in $\mathbf{P}'$ are infeasible, our aim is to seek the first feasible solution from the search space. The information from f then becomes unimportant and hence, only R_2 and R_3 are used. In this case, the weight vector w is set to $(0, 1/2, 1/2)$. When feasible individuals exist in $\mathbf{P}'$, the algorithm should explore the search space in order to find an optimum solution and hence, the information from f becomes relevant. In the last case, R_2 and R_3 serve as terms to penalize infeasible solutions and w is set to $(1/3, 1/3, 1/3)$, since R_1 is also used along with R_2 and R_3. In addition, R_1 enables us to relate the infeasible individuals to the feasible individuals based on the numerical property f. The consideration of this term allows us to retain for the next iteration only those infeasible solutions with a small degree of constraint violation and a small objective function value. This strategy is necessary to maintain the diversity of the population in order to explore into infeasible regions of the search space as well.

4.6 Change detection

Swarm algorithms operating in dynamic environments need a specific mechanism to detect changes in the environment. There are two types of changes that can affect the search process: (1) changes in the objective function and (2) changes in the constraint functions. The change detection problem is addressed based on the use of information extracted from the objective and constraint functions by a set of points in the search space. This extraction of information can be done in two ways. One is to use the own swarm particles as points of information collection (population-based detection) and the other is to use an additional population of sensors (or detectors) as points of information collection (sensor-based detection) (Richter, 2009; Richter & Dietel, 2010).

This work addresses the change detection problem by using a fixed set of detectors. The detectors are established in the search space and then their objective values and constraint violations are evaluated after each iteration.

If the present and past objective values and constraint violations are different, then, indeed, the environment has changed. The essential steps of the mechanism to detect changes in the environment can be summarized as the pseudo code shown in Algorithm 4.

The set of detectors differs fully from particles that constitute the swarm. Therefore, our mechanism is not based on performance drop. The detectors are distributed in the search space using a low-discrepancy (quasi-random) sequence that can cover the search space more evenly, since it can achieve a measure of discrepancy much smaller when compared with the measure of discrepancy of a sequence of pseudo-random numbers (Nguyen et al., 2007; Pant et al., 2008).

Algorithm 4 ChangeDetection() procedure.

Input: Υ (the set of detectors), $\mathbf{f}_o = f(\Upsilon, t-1|\Theta_{t-1})$, $\mathbf{s}_o = s(\Upsilon, t-1|\Theta_{t-1})$ (objective values and constraint violations at time $t-1$), $\mathbf{f}_c = f(\Upsilon, t|\Theta_t)$, and $\mathbf{s}_c = s(\Upsilon, t|\Theta_t)$ (objective values and constraint violations at time t).

```
1: Flag ← false
2: if at least one value among the objective values and the constraint violations is not the same as those of the last evaluation of the detectors then
3:     Flag ← true
4: end if
5: return Flag.
```

When a change in the environment is detected, a random-immigrants scheme acts to introduce population diversity. Part of the swarm is replaced by particles randomly generated and this strategy acts only on the first iteration of the changed environment. In addition, a mechanism to update memories associated with the particles also acts on the first iteration of a changed environment. This mechanism is based on re-evaluation of fitness values and replacement using the RankingTOPSIS procedure such as described in Section 4.5.

Now that all components of the new approach have been described, the pseudo code of EBBPSO-T for DCOPs can be presented in Algorithm 5, followed by its flowchart shown in Figure 4.1.

4.7 Convergence analysis

Theoretical analyses of the PSO algorithm have been offered in the literature. A detailed review of these studies is presented in Section B.5 (see Appendix B). Some studies are developed using results of the theory of stochastic processes. Jiang, Luo & Yang (2007) were able to present a convergence analysis for the standard PSO by considering the positions of a particle as a collection of random variables indexed by time. Thus, PSO was studied using results of stochastic convergence of random variables to formalize the idea that the swarm of particles can be expected to settle in a pattern. The convergence condition of the system and guidelines for parameter selection were derived. In that study, the only assumption made by Jiang and co-authors was stagnation. Poli (2009) introduced a method that allows one to exactly determine the moments of the sampling distribution produced by PSO and explain how they change over the generations. Again, the only assumption made by Poli was stagnation. Zhang et al. (2014a) introduced a new BBPSO with adaptive disturbance and developed a stochastic convergence analysis for the proposed algorithm by using the same approach adopted by Jiang, Luo & Yang (2007).

Under the same assumptions adopted by Jiang, Luo & Yang (2007) and Zhang et al. (2014a), a convergence analysis for EBBPSO-T is developed here. First, it is important to observe two points: (1) a dynamic optimization problem can be seen as a sequence of static optimization problems over time, which are solved online by an algorithm as time goes by. The overall algorithm performance depends on its performance in each environment. For convenience, our analysis is carried out considering a unique environment, but its results can be applied to any other environment of the dynamic problem; (2) when a swarm algorithm operates on an optimization problem, the values of pbest, lbest, and gbest are continually updated. For convenience, stagnation is initially assumed as a simplifying assumption, but subsequently this hypothesis will be partially relaxed. Now, our analysis can be presented in steps:

Step 1. If EBBPSO-T is in a stagnation phase, each particle behaves independently. In addition, during stagnation, each dimension is independent. So, the behaviour of each particle can be analysed in isolation, considering a one-particle swarm in one dimension. Given p (pbest), n (lbest), and g (gbest), the dynamic rule can be

Algorithm 5 EBBPSO-T for DCOPs.

Input: $K, \mathscr{N}, \nu, w, f(\cdot|\Theta_t)$, and $g_i(\cdot|\Theta_t), i \in I$.

$\tau \leftarrow 1$; $t \leftarrow 0$; Flag $\leftarrow$ false
$\Theta_t = (p(t), q(t), r(t), s(t))'$
Establish the set of detectors Υ in $\mathbb{S}$
for $k \in \{1, \ldots, K\}$ **do**
  Initialize $\mathbf{x}_k$
  Set $\mathbf{p}_k = \mathbf{x}_k$
end for
for $k \in \{1, \ldots, K\}$ **do**
  $\mathbf{n}_k \leftarrow$ RankingTOPSIS$(\mathbf{p}_l | l \in \mathscr{N}_k)$
end for
$\mathbf{g} \leftarrow$ RankingTOPSIS$(\mathbf{n}_k | k = 1, \ldots, K)$
$\mathbf{x}^* \leftarrow \emptyset$; $\mathbf{x}^* \leftarrow \mathbf{x}^* \cup \mathbf{g}$
$\mathbf{f}^* \leftarrow \emptyset$; $\mathbf{f}^* \leftarrow \mathbf{f}^* \cup f(\mathbf{g}, t|\Theta_t)$
repeat
  $\tau \leftarrow \tau + 1$
  if Flag = true **then**
    Call RandomImmigrants()
    for $k \in \{1, \ldots, K\}$ **do**
      Call RankingTOPSIS() to update $\mathbf{p}_k$ and $\mathbf{n}_k$
    end for
    $\mathbf{g} \leftarrow$ RankingTOPSIS$(\mathbf{n}_k | k = 1, \ldots, K)$
    Flag $\leftarrow$ false
  end if
  $\hat{\mathscr{H}}^S \leftarrow$ ShrinkageEntropyEstimation()
  for $k \in \{1, \ldots, K\}$ **do**
    Change the position according to Eq. (4.7)
    Call RankingTOPSIS() to update $\mathbf{p}_k$ and $\mathbf{n}_k$
  end for
  $\mathbf{g} \leftarrow$ RankingTOPSIS$(\mathbf{n}_k | k = 1, \ldots, K)$
  $\mathbf{x}^* \leftarrow \mathbf{x}^* \cup \mathbf{g}$
  $\mathbf{f}^* \leftarrow \mathbf{f}^* \cup f(\mathbf{g}, t|\Theta_t)$
  Evaluate $\mathbf{f}_o = f(\Upsilon, t|\Theta_t)$ and $\mathbf{s}_o = s(\Upsilon, t|\Theta_t)$
  if $\tau \bmod \Delta = 0$ **then**
    $t \leftarrow t + 1$; Update Θ_t
  end if
  Evaluate $\mathbf{f}_c = f(\Upsilon, t|\Theta_t)$ and $\mathbf{s}_c = s(\Upsilon, t|\Theta_t)$
  Flag $\leftarrow$ ChangeDetection$(\mathbf{f}_o, \mathbf{s}_o, \mathbf{f}_c, \mathbf{s}_c)$
until some termination condition is met
return $\mathbf{x}^*$ and $[1 : \tau_{\max}] \times \mathbf{f}^*$ (convergence plot).

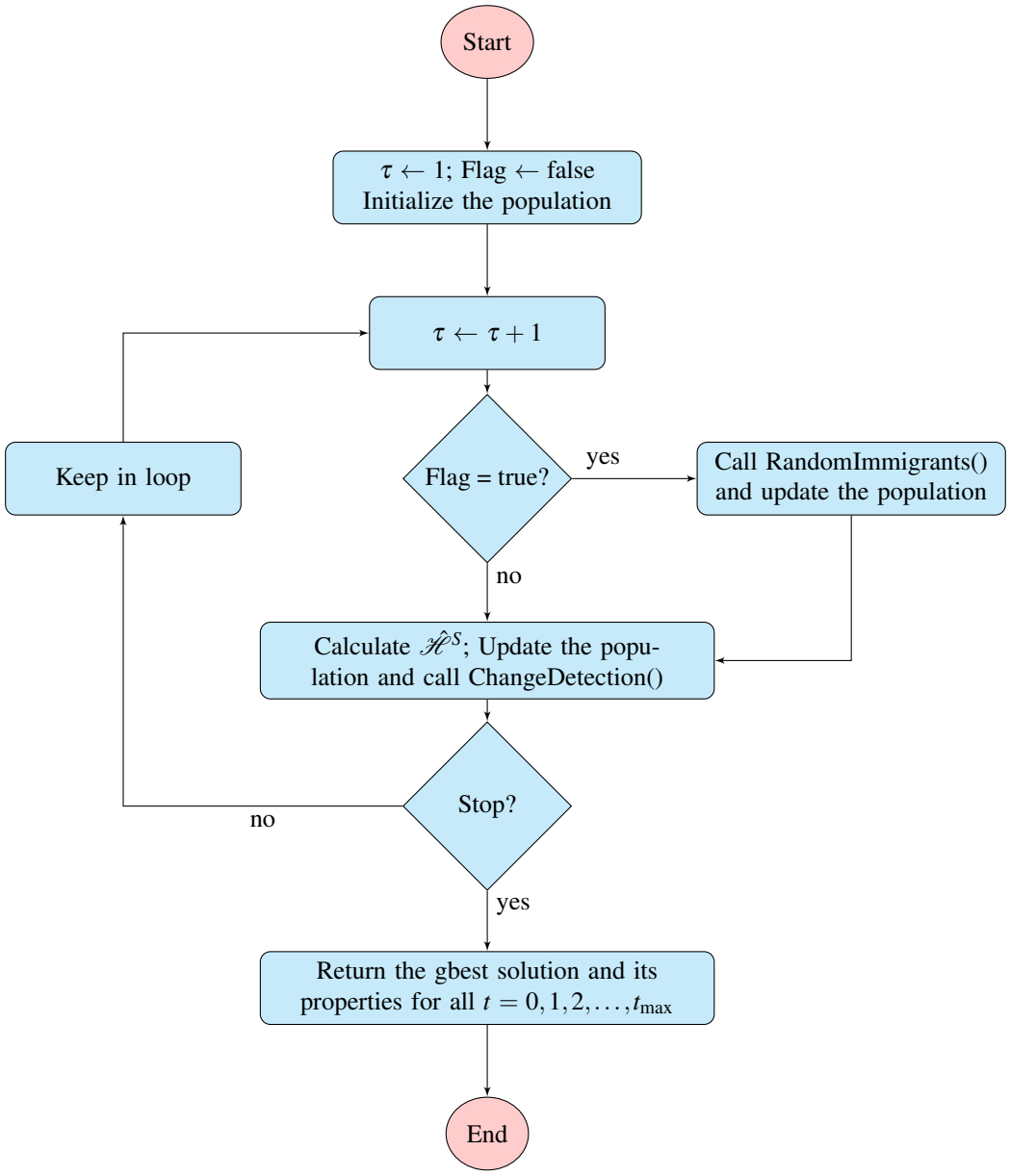

Figure 4.1: The flowchart of EBBPSO-T

written as

$$x_{\tau+1} = \hat{\mathscr{H}}_\tau \left[\frac{p+g}{2} + |p-g| \cdot z \right] + \hat{\mathscr{W}}_\tau \left[\frac{p+n}{2} + |p-n| \cdot t \right] \tag{4.16}$$

where $\hat{\mathscr{H}}_\tau = \hat{\mathscr{H}}_S^\tau / \mathscr{H}_{\max}$, $\hat{\mathscr{W}}_\tau = 1 - \hat{\mathscr{H}}_\tau$, $z \sim N(0,1)$, $t \sim t(\nu,0,1)$, and τ represents the iteration counter. Note that, for convenience in notation, τ is used as a subscript index rather than superscript as done in Appendix B. From Eq. (4.16), it follows that for $\nu > 1$

$$\mathrm{E}(x_{\tau+1}) = \frac{1}{2}\left(p + \hat{\mathscr{H}}_\tau g + \hat{\mathscr{W}}_\tau n\right) \tag{4.17}$$

for all $\tau \geq 0$ and for $\nu > 2$

$$\mathrm{Var}(x_{\tau+1}) = \hat{\mathscr{H}}_\tau^2 (p-g)^2 + \hat{\mathscr{W}}_\tau^2 \frac{\nu(p-n)^2}{\nu-2} \tag{4.18}$$

and

$$\mathrm{E}(x_{\tau+1}^2) = \hat{\mathscr{H}}_\tau^2 \left[\frac{(p+g)^2}{4} + (p-g)^2\right] + \frac{\hat{\mathscr{H}}_\tau \hat{\mathscr{W}}_\tau}{2}(p+g)(p+n) + \hat{\mathscr{W}}_\tau^2 \left[\frac{(p+n)^2}{4} + \frac{\nu(p-n)^2}{\nu-2}\right] \tag{4.19}$$

for all $\tau \geq 0$.

Step 2. Now, the following situation can be considered: the values of pbest and lbest are constantly updated during the search process, but the value of gbest is kept constant as the best position previously found by the swarm. This means that the stagnation assumption is now partially relaxed. Search for the global-best solution allowing changes in their memories (pbest and lbest) implies that the spatial extension of the search for a particle decreases over time. In fact, Blackwell (2005) suggested that the maximum spatial extension of a swarm along any axis decreases exponentially with time. Hence, p_τ and n_τ converge almost surely to g. As a result, $\mathrm{E}(x_\tau) \to g$, $\mathrm{Var}(x_\tau) \to 0$, and x_τ converges in mean square to g if $\nu > 2$.

Step 3. As each dimension of the position of a particle is updated independently of the others, it follows that for $\nu > 1$

$$\mathrm{E}(\mathbf{x}_k^{\tau+1}) = \frac{1}{2}\left(\mathbf{p}_k^\tau + \hat{\mathscr{H}}_\tau \mathbf{g} + \hat{\mathscr{W}}_\tau \mathbf{n}_k^\tau\right) \to \mathbf{g} \tag{4.20}$$

and for $\nu > 2$

$$\mathrm{Var}(\mathbf{x}_k^{\tau+1}) = \hat{\mathscr{H}}_\tau^2 (\mathbf{p}_k^\tau - \mathbf{g})^2 + \hat{\mathscr{W}}_\tau^2 \frac{\nu(\mathbf{p}_k^\tau - \mathbf{n}_k^\tau)^2}{\nu-2} \to \mathbf{0} \tag{4.21}$$

since $\mathrm{dist}(\mathbf{p}_k^\tau, \mathbf{g}) \to \mathbf{0}$ and $\mathrm{dist}(\mathbf{n}_k^\tau, \mathbf{g}) \to \mathbf{0}$ almost surely when $\tau \to \infty$. Hence, $\mathbf{x}_k^\tau$ converges in mean square to $\mathbf{g}$ if $\nu > 2$. Finally, this conclusion applies to each particle, thus the swarm as a whole will converge to $\mathbf{g}$ when τ tends to infinity.

In summary, three special cases can be highlighted:

1. For all $\nu > 2$, the random process that generates the position of a particle has mean and variance both finite for all iterations and the swarm converges in mean square to $\mathbf{g}$ when $\tau \to \infty$.
2. For all $1 < \nu \leq 2$, the random process that generates the position of a particle has finite mean and infinite variance for all iterations and the swarm converges in mean to $\mathbf{g}$ when $\tau \to \infty$.
3. For all $\nu \leq 1$, the random process that generates the position of a particle has undefined mean and also undefined variance for all iterations, and therefore, no type of convergence can be established.

It is also important to note that this section provides an analysis to establish that the particles converge to a stable point. The stable point is shown to be the global-best position. This is not a proof of convergence to a global minimum, but only states that the swarm will reach an equilibrium point.

To complete this section, two examples are provided to empirically illustrate the convergence of EBBPSO-T when it is applied to solve DCOPs.

Example 4.1. Consider the G24-3 problem that was discussed in Example 2.2 (see Chapter 2). This DCOP was simulated considering $\delta = 4$ and $S = 20$. The window Δ, where the dynamic problem remains constant, was defined as 40 iterations. This problem was solved using EBBPSO-T with $K = 25$ particles, a neighborhood system $\mathscr{N}$ defined by a ring topology, and $\nu = 2.1$. Table 4.1 shows the obtained results for each environment ($t = 0, 1, 2, \ldots$): $\mathbf{x}^* = (x_1^*, x_2^*)$ is the best solution obtained by EBBPSO-T and $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)$ is the global minimum. In addition, Table 4.1 shows $f^* = f(\mathbf{x}^*, t)$ (the objective value of $\mathbf{x}^*$ at time t), $\bar{f} = f(\bar{\mathbf{x}}, t)$ (the objective value of $\bar{\mathbf{x}}$ at time t), and the absolute error $|f^* - \bar{f}|$ for each environment. Figure 4.2 shows the convergence plot of EBBPSO-T and Figure 4.3 shows the Shannon's index of diversity for each iteration. For each environment of the problem, EBBPSO-T converges to an approximate solution close to the global minimum, thus obtaining small absolute errors. Note that all obtained solutions are feasible solutions. As discussed in Example 2.2 the size and the shape of the feasible regions change over time and a new global minimum is revealed whenever the environment is changed. In addition, the old minimum value is not changed, since the objective function is static. In this scenario, EBBPSO-T was able to track the global minimum whenever it is revealed. Finally, considering the performance measures discussed in Section 2.9 of the Chapter 2, the solution shown in Figure 4.2 has an offline error of 0.029 and a best-error-before-change of 0.00023.

Table 4.1: Best solution using EBBPSO-T for the G24-3 problem (see Example 4.1).

t	$\bar{x}_1$	$\bar{x}_2$	x_1^*	x_2^*	f^*	$\bar{f}$	$\lvert f^* - \bar{f}\rvert$	g_1^*	g_2^*
0	2.32952	1.17849	2.32965	1.17784	-3.50749	-3.50801	5.21E-04	-1.72E-03	-4.34E-05
1	2.32952	1.37849	2.32963	1.37792	-3.70755	-3.70801	4.58E-04	-1.46E-03	-5.68E-05
2	2.32952	1.57849	2.32954	1.57832	-3.90786	-3.90801	1.52E-04	-2.91E-04	-1.01E-04
3	2.32952	1.77849	2.32945	1.77784	-4.10765	-4.10801	3.62E-04	-6.56E-05	-9.95E-04
4	2.32952	1.97849	2.32958	1.97821	-4.30779	-4.30801	2.23E-04	-7.79E-04	-5.56E-06
5	2.32952	2.17849	2.32964	2.17785	-4.50757	-4.50801	4.36E-04	-1.63E-03	-7.75E-05
6	2.32952	2.37849	2.32959	2.37814	-4.70784	-4.70801	1.67E-04	-9.41E-04	-1.89E-05
7	2.32952	2.57849	2.32957	2.57818	-4.90796	-4.90801	4.53E-05	-7.42E-04	-6.18E-05
8	2.32952	2.77849	2.32956	2.77828	-5.10796	-5.10801	4.75E-05	-5.56E-04	-9.54E-06
9	2.32952	2.97849	2.32954	2.97835	-5.30794	-5.30801	6.93E-05	-2.94E-04	-5.74E-05
10	2.32952	3.17849	2.32953	3.17844	-5.50798	-5.50801	2.55E-05	-1.24E-04	-1.49E-05

$\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)$, $\mathbf{x}^* = (x_1^*, x_2^*)$, $\bar{f} = f(\bar{\mathbf{x}}, t)$, $f^* = f(\mathbf{x}^*, t)$, and $g_i^* = g_i(\mathbf{x}^*, t)$

Example 4.2. Consider the G24-4 problem that was discussed in Example 2.3 (see Chapter 2). This DCOP was simulated considering $\kappa = 1/2$, $\delta = 4$, and $S = 20$. The window Δ, where the dynamic problem remains constant, was defined as 40 iterations. This problem was solved using EBBPSO-T with $K = 25$ particles, a neighborhood system $\mathscr{N}$ defined by a ring topology, and $\nu = 2.1$. Table 4.2 shows the obtained results for each environment ($t = 0, 1, 2, \ldots$): $\mathbf{x}^* = (x_1^*, x_2^*)$ is the best solution obtained by EBBPSO-T and $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)$ is the global minimum. In addition, Table 4.2 shows $f^* = f(\mathbf{x}^*, t)$, $\bar{f} = f(\bar{\mathbf{x}}, t)$, and the absolute error $|f^* - \bar{f}|$ for each environment. Figure 4.4 shows the convergence plot of EBBPSO-T and Figure 4.5 shows the Shannon's index of diversity for each iteration. For each environment of the problem, EBBPSO-T converges to an approximate solution close to the global minimum, thus obtaining small absolute errors. Note that all obtained solutions are feasible solutions. As discussed in Example 2.3 the objective function and the constraints change over time. As a result, the location of the global minimum is switched between two disconnected feasible regions. In addition, the size and the shape of the feasible regions also change over time. In this scenario, EBBPSO-T was able to track the global minimum whenever it is switched between feasible regions. Finally, considering the performance measures discussed in Section 2.9 of the Chapter 2, the solution shown in Figure 4.4 has an offline error of 0.117 and a best-error-before-change of 0.00094.

Table 4.2: Best solution using EBBPSO-T for the G24-4 problem (see Example 4.2).

t	$\bar{x}_1$	$\bar{x}_2$	x_1^*	x_2^*	f^*	$\bar{f}$	$\lvert f^* - \bar{f}\rvert$	g_1^*	g_2^*
0	2.32952	3.17849	2.33016	3.17475	-5.50490	-5.50801	3.11E-03	-8.95E-03	-7.54E-04
1	0.61160	3.24210	0.61133	3.24107	-3.24107	-3.24209	1.02E-03	-3.10E-04	-6.65E-03
2	0.61160	3.04210	0.61105	3.04037	-2.42932	-2.43049	1.17E-03	-2.65E-04	-1.32E-02
3	0.61160	2.84210	0.61137	2.84127	-2.84159	-2.84210	5.12E-04	-2.29E-04	-5.60E-03
4	2.32952	2.37849	2.33004	2.37561	-4.70596	-4.70801	2.05E-03	-7.16E-03	-4.18E-04
5	0.61160	2.44210	0.61128	2.44106	-2.44141	-2.44209	6.75E-04	-1.78E-04	-7.81E-03
6	0.61160	2.24210	0.61138	2.24123	-1.63013	-1.63049	3.55E-04	-2.91E-04	-5.47E-03
7	0.61160	2.04210	0.61138	2.04090	-2.04181	-2.04210	2.92E-04	-6.22E-04	-5.75E-03
8	2.32952	1.57849	2.32974	1.57734	-3.90708	-3.90801	9.32E-04	-2.93E-03	-1.31E-04
9	0.61160	1.64210	0.61153	1.64183	-1.64193	-1.64209	1.56E-04	-9.47E-05	-1.69E-03
10	0.61160	1.44210	0.61156	1.44197	-0.83042	-0.83049	6.76E-05	-2.99E-05	-9.62E-04

$\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)$, $\mathbf{x}^* = (x_1^*, x_2^*)$, $\bar{f} = f(\bar{\mathbf{x}}, t)$, $f^* = f(\mathbf{x}^*, t)$, and $g_i^* = g_i(\mathbf{x}^*, t)$

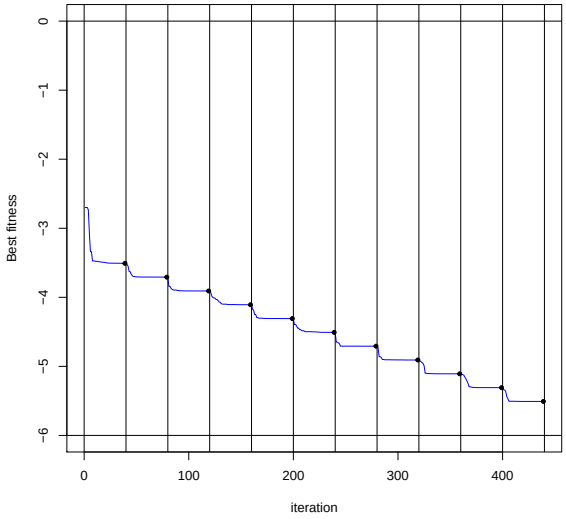

Figure 4.2: Convergence plot for the G24-3 problem.

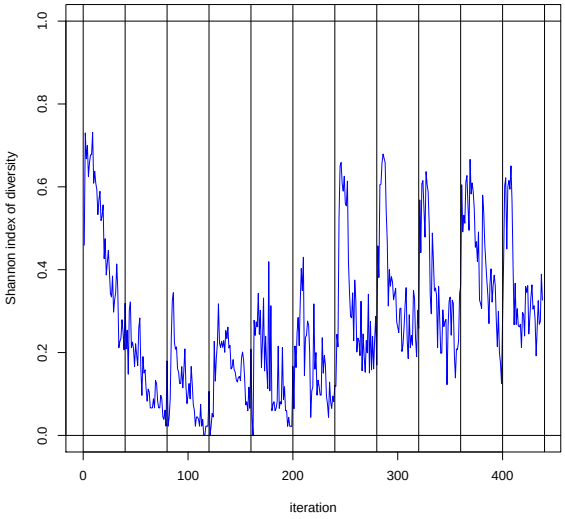

Figure 4.3: Shannon's index of diversity.

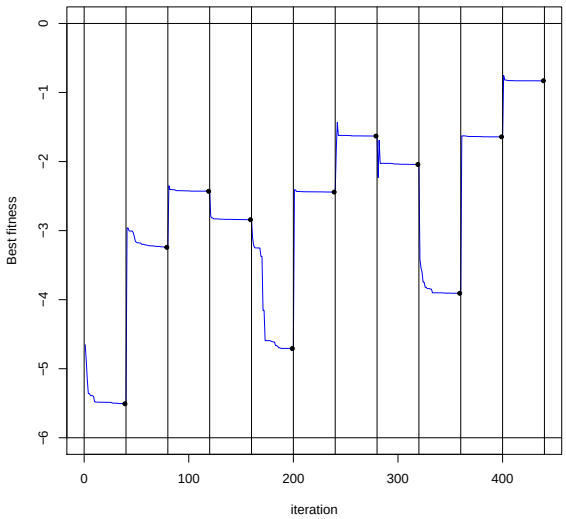

Figure 4.4: Convergence plot for the G24-4 problem.

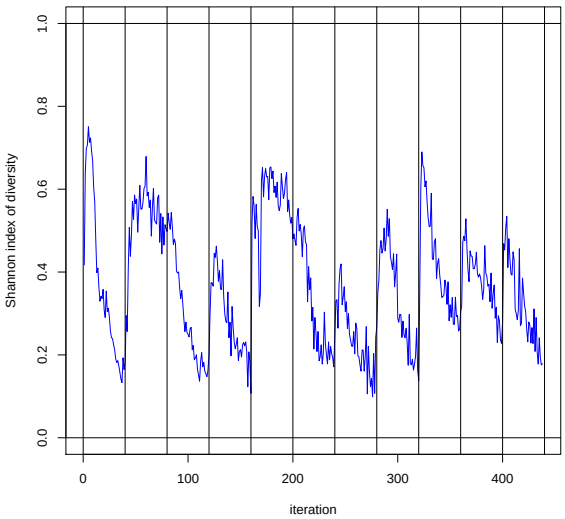

Figure 4.5: Shannon's index of diversity.

Part III

Results, Discussion, and Conclusion

Chapter 5

Experimental Results

This chapter presents an experimental analysis conducted to investigate the performance of the EBBPSO-T to solve DCOPs. Results of experiments designed to evaluate the performance of the proposed algorithm are reported and discussed along with experimental results obtained by other algorithms for comparison purposes. An application is also developed, where EBBPSO-T is applied to solve the dynamic economic dispatch problem in power systems.

5.1 Benchmark problems

An experimental analysis was conducted to investigate empirically the performance of the EBBPSO-T to solve DCOPs. The G24 set of DCOPs (Nguyen & Yao, 2009; Nguyen, 2010; Nguyen & Yao, 2012) was used in this experimental study. These problems have characteristics that are representative of real-world DCOPs. Chapter 2 provides a description of all problems in this set, including the parameters used to simulate each problem. In particular, some detailed examples of problems in this set are presented in Section 2.6. As discussed in Chapter 2, these problems can be simulated considering the following parameters: κ, S, and Δ, where κ determines the severity of objective function changes, S determines the severity of constraint changes, and Δ determines the window where the dynamic problem remains constant. The basic principle used to select numerical values for these parameters was to ensure that our experimental results could be fairly compared with the experimental results obtained by other algorithms also tested on the problems of the G24 set. The values chosen are: $\kappa = 1/2$, $S = 20$, and $\Delta = 40$ iterations, which is equivalent to define a change frequency equal to 1000 function evaluations if the algorithm uses a swarm (population) with $K = 25$ particles. Furthermore, the constants $C_1 = 1.470561702$, $C_2 = 3.442094786232$, and $R_a = 0.858958496$ are necessary to simulate the problems G24-8a and G24-8b.

5.2 Experimental setup

Four swarm algorithms were investigated in our experiments. These algorithms and their characteristics are described as follows:

LBBPSO-SR. Lbest BBPSO-SR has two parameters: K (the number of particles) and $\mathcal{N}$ (the neighborhood system). LBBPSO-SR was tested with $K = 25$ particles and $\mathcal{N}$ defined by a ring topology. Its dynamic rule is defined by

$$x_{kd}^{\tau+1} = \mu_{kd}^{\tau} + \sigma_{kd}^{\tau} z_d \tag{5.1}$$

for all k, d and $\tau \geq 0$, where $\mu_{kd}^{\tau} = 0.5(p_{kd}^{\tau} + n_{kd}^{\tau})$, $\sigma_{kd}^{\tau} = |p_{kd}^{\tau} - n_{kd}^{\tau}|$, and $z_d \sim N(0,1)$. The constraint-handling strategy is based on sum of ranks (Ho & Shimizu, 2007).

GBBPSO-SR. Gbest BBPSO-SR has two parameters: K and $\mathcal{N}$. GBBPSO-SR was tested with $K = 25$ particles and $\mathcal{N}$ defined by a fully connected topology. Its dynamic rule is defined by

$$x_{kd}^{\tau+1} = \eta_{kd}^{\tau} + \rho_{kd}^{\tau} z_d \tag{5.2}$$

for all k, d and $\tau \geq 0$, where $\eta_{kd}^{\tau} = 0.5(p_{kd}^{\tau} + g_d^{\tau})$, $\rho_{kd}^{\tau} = |p_{kd}^{\tau} - g_d^{\tau}|$, and $z_d \sim N(0,1)$. The constraint-handling strategy is based on sum of ranks (Ho & Shimizu, 2007).

EBBPSO-SR. Entropy-based BBPSO-SR has three parameters: K, $\mathscr{N}$, and ν. EBBPSO-SR was tested with $K = 25$ particles, $\mathscr{N}$ defined by a ring topology, $\nu = 1.0$ and 2.1, and a mechanism to monitor the population diversity that use the Shannon's index such as discussed in Subsection 4.3. EBBPSO-SR has an entropy-based dynamic rule such as discussed in Subsection 4.4, i.e., the position of a particle is updated as follows

$$x_{kd}^{\tau+1} = \hat{\mathscr{H}}^{\tau} \cdot (\eta_{kd}^{\tau} + \rho_{kd}^{\tau} z_{1d}) + \hat{\mathscr{W}}^{\tau} \cdot (\mu_{kd}^{\tau} + \lambda_d^{-1/2} \sigma_{kd}^{\tau} z_{2d}) \tag{5.3}$$

for all k, d and $\tau \geq 0$, where $\hat{\mathscr{H}}^{\tau} = \hat{\mathscr{H}}_S^{\tau} / \mathscr{H}_{\max}$, $\hat{\mathscr{W}}^{\tau} = 1 - \hat{\mathscr{H}}^{\tau}$, $\eta_{kd}^{\tau} = 0.5(p_{kd}^{\tau} + g_d^{\tau})$, $\mu_{kd}^{\tau} = 0.5(p_{kd}^{\tau} + n_{kd}^{\tau})$, $\rho_{kd}^{\tau} = |p_{kd}^{\tau} - g_d^{\tau}|$, $\sigma_{kd}^{\tau} = |p_{kd}^{\tau} - n_{kd}^{\tau}|$, $z_{1d}, z_{2d} \sim N(0,1)$, and $\lambda_d \sim Ga(\nu/2, \nu/2)$. The constraint-handling strategy is based on sum of ranks (Ho & Shimizu, 2007).

EBBPSO-T. Entropy-based BBPSO-T has three parameters: K, $\mathscr{N}$, and ν. EBBPSO-T was tested with $K = 25$ particles, $\mathscr{N}$ defined by a ring topology, $\nu = 1.0$ and 2.1, and a mechanism to monitor the population diversity that use the Shannon's index such as discussed in Subsection 4.3. EBBPSO-T has an entropy-based dynamic rule such as discussed in Subsection 4.4, i.e, each particle uses Eq. (5.3) to update its position. The constraint-handling strategy uses a ranking method with selection based on TOPSIS such as discussed in Subsection 4.5.

The values chosen for the parameter ν represent two distinct characteristic cases. For $\nu = 1.0$ (in fact, for all $\nu \leq 1$), the random process that generates the position of a particle has undefined mean and also undefined variance for all iterations. For $\nu = 2.1$ (in fact, for all $\nu > 2$), the random process that generates the position of a particle has mean and variance both finite for all iterations. Table 5.1 summarizes this information. In addition, all these algorithms are endowed with a mechanism to detect changes in the environment and a mechanism to update particles' memories. The mechanism to detect changes in the environment uses a set Υ with $\min\{\lceil 0.3 \cdot K \rceil, 10\}$ detectors uniformly distributed in the search space. When a change in the environment is detected, a scheme of random immigrants operates on the first iteration of the changed environment, such that $\lceil 0.3 \cdot K \rceil$ particles are replaced by particles randomly generated in the search space.

Table 5.1: Characteristics of the swarm algorithms investigated in the G24 set.

Algorithm	K	$\mathscr{N}$	Dynamic rule	Constraint handling	ν
LBBPSO-SR	25	R	Lbest driven	Ho & Shimizu (2007)	-
GBBPSO-SR	25	FC	Gbest driven	Ho & Shimizu (2007)	-
EBBPSO-SR	25	R	Subsection 4.4*	Ho & Shimizu (2007)	1.0 or 2.1
EBBPSO-T	25	R	Subsection 4.4*	Subsection 4.5*	1.0 or 2.1

R: ring topology; FC: fully connected topology.
* Dynamic rule proposed in this work.
* Constraint-handling strategy proposed in this work.

5.3 Results and discussion

The experimental study consists of a set of experiments. In each experiment, a pair problem/algorithm was considered and sequences of offline errors and execution times were obtained from 50 independent runs, where in each run the tested algorithm was applied to solve the benchmark problem. Mean and standard deviation were calculated for the sequences of errors and execution times. All experiments were executed on a computer with Intel Core(TM) i5-2450M(2.5GHz) processor, 4GB RAM, and MS Windows 8.1 operating system. The algorithms were implemented with the R programming language[1] using RStudio[2] as an integrated development environment for R. All results presented in this section were derived considering the experimental setup described in Section 5.2.

5.3.1 Offline error

Table 5.2 shows the mean and the standard deviation of the offline errors obtained by each algorithm tested on 18 benchmark problems of the G24 set, considering EBBPSO-SR and EBBPSO-T, both with $\nu = 1.0$. The rank

[1] Version 2.15.3, `https://www.r-project.org/`
[2] Version 0.98.1087, `https://www.rstudio.com/`

Table 5.2: Mean and standard deviation of offline errors for four algorithms tested on 18 problems of the G24 set.

Algorithm	LBBPSO-SR	GBBPSO-SR	EBBPSO-SR ($v = 1.0$)	EBBPSO-T ($v = 1.0$)
Problem	Mean (SD) [Rank]	Mean (SD) [Rank]	Mean (SD) [Rank]	Mean (SD) [Rank]
G24-u	0.105 (0.038) [4]	0.072 (0.029) [3]	0.057 (0.005) [2]	**0.056** (0.005) [1]
G24-1	0.212 (0.211) [4]	0.116 (0.128) [3]	0.090 (0.051) [2]	**0.084** (0.041) [1]
G24-f	0.027 (0.009) [4]	0.016 (0.005) [2]	0.017 (0.006) [3]	**0.014** (0.005) [1]
G24-uf	0.014 (0.004) [4]	**0.007** (0.002) [1]	0.010 (0.003) [3]	0.009 (0.003) [2]
G24-2	0.206 (0.054) [4]	0.150 (0.068) [3]	0.149 (0.026) [2]	**0.136** (0.013) [1]
G24-2u	0.051 (0.003) [4]	0.048 (0.003) [3]	0.047 (0.002) [2]	**0.045** (0.001) [1]
G24-3	0.063 (0.015) [4]	0.038 (0.008) [2]	0.039 (0.008) [3]	**0.032** (0.005) [1]
G24-3b	0.190 (0.114) [4]	0.138 (0.089) [3]	0.113 (0.033) [2]	**0.104** (0.015) [1]
G24-3f	0.030 (0.019) [4]	0.014 (0.007) [2]	0.016 (0.007) [3]	**0.012** (0.005) [1]
G24-4	0.212 (0.061) [4]	**0.128** (0.051) [1]	0.146 (0.035) [3]	0.138 (0.022) [2]
G24-5	0.279 (0.142) [4]	0.177 (0.113) [3]	0.137 (0.020) [2]	**0.126** (0.019) [1]
G24-6a	0.173 (0.222) [4]	0.170 (0.258) [3]	0.124 (0.150) [2]	**0.116** (0.099) [1]
G24-6b	0.279 (0.053) [4]	**0.182** (0.042) [1]	0.199 (0.046) [3]	0.184 (0.043) [2]
G24-6c	0.351 (0.059) [4]	0.255 (0.059) [2]	0.263 (0.053) [3]	**0.251** (0.061) [1]
G24-6d	0.581 (0.232) [4]	0.346 (0.267) [2]	0.347 (0.214) [3]	**0.312** (0.203) [1]
G24-7	0.081 (0.016) [3]	0.084 (0.035) [4]	0.046 (0.009) [2]	**0.045** (0.009) [1]
G24-8a	0.383 (0.065) [4]	0.259 (0.067) [3]	0.215 (0.026) [2]	**0.194** (0.022) [1]
G24-8b	0.535 (0.119) [4]	0.314 (0.115) [2]	0.369 (0.124) [3]	**0.312** (0.086) [1]
Minimum	0.014	0.007	0.010	0.009
1st quartile	0.068	0.054	0.046	0.045
Median	0.198	0.133	0.118	0.110
Mean	0.210	0.140	0.132	0.121
3rd quartile	0.279	0.181	0.186	0.173
Maximum	0.581	0.346	0.369	0.312
SD	0.169	0.103	0.110	0.097
Wilcoxon signed-ranks test for pairwise comparisons of algorithms				
Algorithm	LBBPSO-SR	GBBPSO-SR	EBBPSO-SR	-
Mean rank	3.94	2.39	2.50	1.17
$W/L/T$	18/0/0	15/3/0	18/0/0	-
p-value	7.629e-05	1.045e-03	7.629e-05	-

obtained by each algorithm in each problem is also shown and the best results are shown in boldface. Comparisons between EBBPSO-T and its algorithms competitors are summarized in the last rows of Table 5.2, which shows summary statistics of the offline errors, considering the complete set of problems. In addition, $W/L/T$ counts are also shown, which means that EBBPSO-T with $v = 1.0$ wins in W problems, loses in L problems, and ties in T problems.

A multi-problem analysis was developed based on the data presented in Table 5.2. The Wilcoxon signed-ranks test (Demsar, 2006; Derrac et al., 2011) was used for pairwise comparisons of algorithms, when both algorithms are applied to a common set of problems. The results of this analysis are summarized in the last rows of Table 5.2. EBBPSO-T with $v = 1.0$ shows a significant improvement over LBBPSO-SR, GBBPSO-SR, and EBBPSO-SR with $v = 1.0$. The results of the Wilcoxon signed-ranks test are in agreement with the results of the Sign test (Demsar, 2006; Derrac et al., 2011). In our experimental setup, the critical value for a two-tailed Sign test for paired comparisons of algorithms is 13 wins in 18 test problems for a significance level of 10%. This means that an algorithm is significantly better than another if it performs better on at least 13 out of 18 problems.

Table 5.3 shows the mean and the standard deviation of the offline errors obtained by each algorithm tested on 18 benchmark problems of the G24 set, considering EBBPSO-SR and EBBPSO-T, both with $v = 2.1$. Comparisons between EBBPSO-T and its algorithms competitors are summarized in Table 5.3. A multi-problem analysis was also developed based on the data presented in Table 5.3. Taking in account the complete set of benchmark problems, the results obtained by EBBPSO-T with $v = 2.1$ are in agreement with the results obtained by EBBPSO-T when $v = 1.0$. EBBPSO-T with $v = 2.1$ shows a significant improvement over LBBPSO-SR, GBBPSO-SR, and

Table 5.3: Mean and standard deviation of offline errors for four algorithms tested on 18 problems of the G24 set.

Algorithm	LBBPSO-SR	GBBPSO-SR	EBBPSO-SR ($\nu = 2.1$)	EBBPSO-T ($\nu = 2.1$)
Problem	Mean (SD) [Rank]	Mean (SD) [Rank]	Mean (SD) [Rank]	Mean (SD) [Rank]
G24-u	0.105 (0.038) [4]	0.072(0.029) [3]	0.057(0.005) [2]	**0.053**(0.005) [1]
G24-1	0.212 (0.211) [4]	0.116(0.128) [2]	0.118(0.089) [3]	**0.072**(0.015) [1]
G24-f	0.027 (0.009) [4]	0.016(0.005) [2]	0.019(0.006) [3]	**0.015**(0.005) [1]
G24-uf	0.014 (0.004) [4]	**0.007**(0.002) [1.5]	0.008(0.003) [3]	**0.007**(0.002) [1.5]
G24-2	0.206 (0.054) [4]	0.150(0.068) [3]	0.144(0.023) [2]	**0.133**(0.020) [1]
G24-2u	0.051 (0.003) [4]	0.048(0.003) [3]	**0.046**(0.002) [1.5]	**0.046**(0.002) [1.5]
G24-3	0.063 (0.015) [4]	0.039(0.008) [3]	0.037(0.008) [2]	**0.031**(0.005) [1]
G24-3b	0.190 (0.114) [4]	0.138(0.089) [3]	0.118(0.036) [2]	**0.105**(0.018) [1]
G24-3f	0.030 (0.019) [4]	0.014(0.007) [2]	0.016(0.008) [3]	**0.011**(0.005) [1]
G24-4	0.212 (0.061) [4]	**0.128**(0.051) [1]	0.156(0.045) [3]	0.147(0.027) [2]
G24-5	0.279 (0.142) [4]	0.177(0.113) [3]	0.133(0.017) [2]	**0.129**(0.020) [1]
G24-6a	0.173 (0.222) [4]	0.170(0.258) [3]	**0.107**(0.067) [1]	0.118(0.015) [2]
G24-6b	0.279 (0.053) [4]	**0.182**(0.042) [1]	0.216(0.053) [3]	0.185(0.048) [2]
G24-6c	0.351 (0.059) [4]	0.255(0.059) [2]	0.281(0.054) [3]	**0.247**(0.051) [1]
G24-6d	0.581 (0.232) [4]	0.346(0.267) [2]	0.363(0.261) [3]	**0.278**(0.174) [1]
G24-7	0.081 (0.016) [3]	0.084(0.035) [4]	0.044(0.010) [2]	**0.043**(0.009) [1]
G24-8a	0.383 (0.065) [4]	0.259(0.067) [3]	0.203(0.024) [2]	**0.188**(0.022) [1]
G24-8b	0.535 (0.119) [4]	0.314(0.115) [2]	0.378(0.109) [3]	**0.290**(0.076) [1]
Minimum	0.014	0.007	0.008	0.007
1st quartile	0.068	0.054	0.044	0.044
Median	0.198	0.133	0.118	0.111
Mean	0.210	0.140	0.136	0.117
3rd quartile	0.279	0.181	0.191	0.176
Maximum	0.581	0.346	0.378	0.290
SD	0.169	0.103	0.114	0.091
Wilcoxon signed-ranks test for pairwise comparisons of algorithms				
Algorithm	LBBPSO-SR	GBBPSO-SR	EBBPSO-SR	-
Mean rank	3.94	2.42	2.42	1.22
$W/L/T$	18/0/0	15/2/1	16/1/1	-
p-value	7.629e-05	1.045e-03	4.196e-04	-

EBBPSO-SR with $\nu = 2.1$.

These results suggest that EBBPSO-T performed globally well for most of the benchmark problems considered in our experiments, especially when compared with traditional approaches such as LBBPSO-SR and GBBPSO-SR. Empirically, it seems that the strategy of mixture of search directions weighted by the Shannon's index of diversity performed better than strategies using individually the global-best or local-best search directions. This indicates that a mixture of search directions maintains the proper balance between diversification and intensification throughout the search process without loss of traceability of the global optimum, which is dynamic for most of the problems investigated. It is also apparent that the constraint-handling strategy combining the ranking of solutions based on the numerical properties (f,s,u) with TOPSIS-based selection performed better than the constraint-handling strategy based on sum of ranks. Finally, it is also important to note that empirical convergence was observed for EBBPSO-T with $\nu = 1.0$ in all the problems investigated, although in this case, there is no theoretical result establishing convergence for a stable fixed point, as is the case for EBBPSO-T with $\nu = 2.1$.

5.3.2 Comparisons with other algorithms

It is important to compare the performance of EBBPSO-T with the performances of other algorithms established in the literature on dynamic constrained optimization. Appendix D presents the full set of results obtained by Nguyen (2010, p. 181) for 18 algorithms tested on the G24 set along with the results obtained by the algorithm GSARepair proposed by Pal et al. (2013).

Table 5.4: Comparison of the algorithm EBBPSO-T with respect to 9 other algorithms on 18 problems of the G24 set.

Algorithm	EBBPSO-T										
	$\nu = 1.0$	$\nu = 2.1$	A19	A18	A14	A17	A13	A12	A9	A11	A10
Problem	Mean offline error [Rank]										
G24-u	0.056[3]	0.053[2]	**0.049**[1]	0.091[4]	0.120[5]	0.123[6]	0.175[9]	0.152[7]	0.254[10]	0.156[8]	0.319[11]
G24-1	0.084[4]	**0.072**[1]	0.132[11]	0.085[5]	0.099[8]	0.103[9]	0.091[6]	0.078[2]	0.082[3]	0.104[10]	0.093[7]
G24-f	**0.014**[1]	0.015[2]	0.029[7.5]	0.021[4]	0.020[3]	0.022[5]	0.043[10]	0.029[7.5]	0.028[6]	0.041[9]	0.045[11]
G24-uf	0.009[2]	**0.007**[1]	0.047[6]	0.030[4.5]	0.030[4.5]	0.029[3]	0.249[11]	0.151[7]	0.194[8]	0.248[10]	0.218[9]
G24-2	0.136[3]	0.133[2]	0.182[10]	**0.099**[1]	0.177[9]	0.138[4]	0.161[5]	0.171[7.5]	0.162[6]	0.196[11]	0.171[7.5]
G24-2u	**0.045**[1]	0.046[2]	0.196[10.5]	0.060[3]	0.120[7]	0.123[8]	0.096[6]	0.082[4]	0.187[9]	0.084[5]	0.196[10.5]
G24-3	0.032[8]	0.031[7]	0.028[3.5]	0.028[3.5]	0.099[10]	0.103[11]	**0.026**[1]	0.029[5.5]	0.029[5.5]	0.035[9]	0.027[2]
G24-3b	0.104[10]	0.105[11]	0.076[9]	0.068[5]	**0.020**[1]	0.022[2]	0.074[7]	0.059[4]	0.058[3]	0.075[8]	0.071[6]
G24-3f	0.012[4]	0.011[3]	0.009[2]	**0.005**[1]	0.030[11]	0.029[10]	0.027[9]	0.014[5.5]	0.014[5.5]	0.026[8]	0.025[7]
G24-4	0.138[4.5]	0.147[9]	0.073[3]	0.140[6.5]	0.177[11]	0.138[4.5]	0.062[2]	0.143[8]	0.140[6.5]	0.164[10]	**0.059**[1]
G24-5	0.126[4]	0.129[5]	0.153[9]	0.114[3]	**0.059**[1]	0.074[2]	0.131[6.5]	0.154[10]	0.152[8]	0.177[11]	0.131[6.5]
G24-6a	0.116[4]	0.118[5]	**0.033**[1]	0.315[6]	0.041[3]	0.036[2]	0.339[7]	0.361[9]	0.366[10]	0.395[11]	0.358[8]
G24-6b	0.184[2]	0.185[3]	**0.047**[1]	0.334[5]	0.407[11]	0.319[4]	0.342[7]	0.352[9]	0.346[8]	0.391[10]	0.341[6]
G24-6c	0.251[3]	0.247[2]	**0.045**[1]	0.263[4]	0.296[6]	0.280[5]	0.330[9]	0.350[10]	0.323[7]	0.386[11]	0.326[8]
G24-6d	0.312[9]	0.278[3]	**0.037**[1]	0.242[2]	0.281[4.5]	0.291[7]	0.281[4.5]	0.302[8]	0.315[10]	0.352[11]	0.286[6]
G24-7	0.045[3]	0.043[2]	**0.018**[1]	0.192[10]	0.230[11]	0.171[8]	0.068[5]	0.153[6]	0.154[7]	0.179[9]	0.067[4]
G24-8a	0.194[2]	**0.188**[1]	0.202[3]	0.415[7]	0.408[5]	0.427[9]	0.397[4]	0.449[11]	0.448[10]	0.422[8]	0.413[6]
G24-8b	0.312[5]	0.290[4]	**0.192**[1]	0.416[8]	0.446[9]	0.447[10]	0.242[2]	0.339[6]	0.341[7]	0.449[11]	0.257[3]
Minimum	0.009	0.007	0.009	0.005	0.020	0.022	0.026	0.014	0.014	0.026	0.025
1st quartile	0.045	0.044	0.034	0.062	0.046	0.046	0.070	0.079	0.096	0.089	0.068
Median	0.110	0.111	0.048	0.107	0.120	0.123	0.146	0.152	0.174	0.178	0.184
Mean	0.121	0.117	0.086	0.162	0.170	0.160	0.174	0.187	0.200	0.216	0.189
3rd quartile	0.173	0.176	0.148	0.258	0.268	0.253	0.273	0.330	0.321	0.377	0.311
Maximum	0.312	0.290	0.202	0.416	0.446	0.447	0.397	0.449	0.448	0.449	0.413
SD	0.097	0.091	0.069	0.136	0.143	0.136	0.124	0.136	0.133	0.147	0.129
Friedman test for comparisons of algorithms (F=47.69, p-value=7.068e-07)											
Mean rank	4.03	3.61	4.53	4.58	6.67	6.08	6.17	7.06	7.19	9.44	6.64
$W/L/T$	-	7/11/0	7/11/0	12/6/0	14/4/0	13/4/1	13/5/0	14/4/0	15/3/0	17/1/0	13/5/0
p-value	-	1.000000	1.000000	1.000000	0.089168	0.265137	0.265137	0.049342	0.037607	0.000010	0.089168

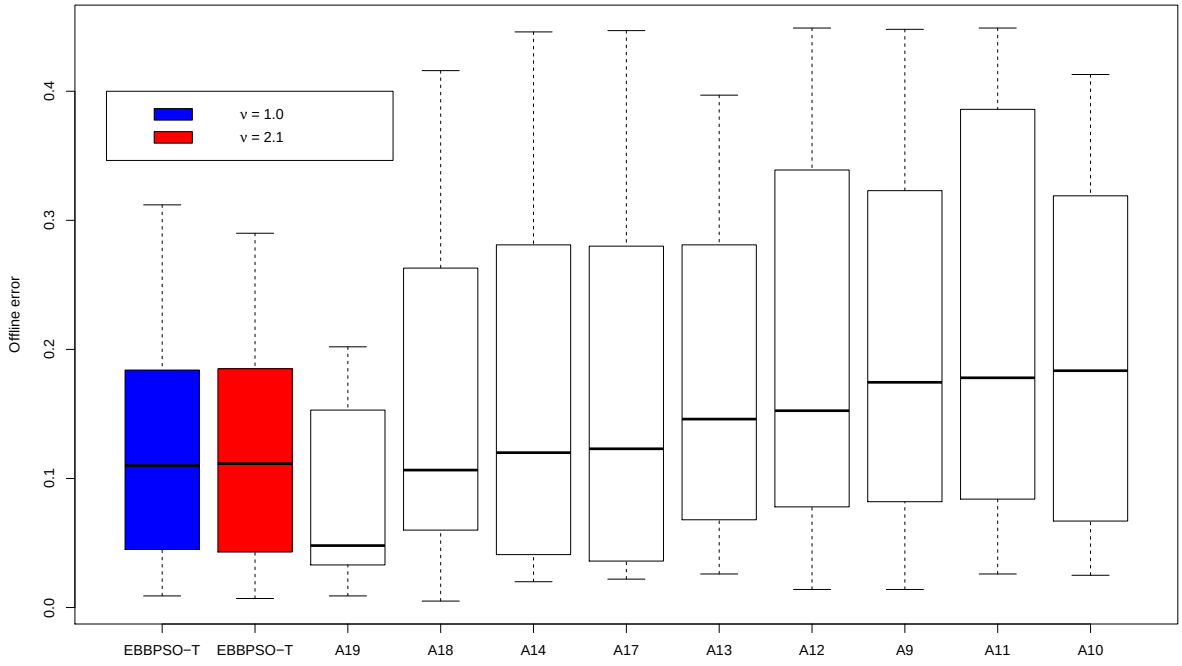

Figure 5.1: Graphical representation of the results presented in Table 5.4. Box-plots of the offline errors for each investigated algorithm: EBBPSO-T (with $v = 1.0$ and 2.1), GSARepair (A19), dGenocop (A18), Genocop (A14), GenocopwUPCwNRR (A17), dRepairHyperM-OOR (A13), dRepairRIGA-OOR (A12), dRepairRIGA (A9), dRepairGA-OOR (A11), dRepairHyperM (A10).

For comparison purposes, Table 5.4 shows the experimental results of 9 algorithms (8 algorithms investigated by Nguyen (2010, p. 181) and the algorithm GSARepair investigated by Pal et al. (2013)) along with the results obtained by EBBPSO-T, all results obtained on the G24 set of DCOPs. The rank obtained by each algorithm in each problem is also shown in Table 5.4 and the best results are shown in boldface. Comparisons between EBBPSO-T and its algorithms competitors are summarized in the last rows of Table 5.4, which shows summary statistics of the offline errors and $W/L/T$ counts. Figure 5.1 shows a graphical representation of the results presented in Table 5.4.

Once again, a multi-problem analysis was developed based on the data presented in Table 5.4 in order to investigate whether the performance obtained by EBBPSO-T is statistically different from those obtained by its competitors on the set of benchmark problems. The Friedman test (Demsar, 2006; Derrac et al., 2011) was used for multiple comparisons of algorithms. The results of this analysis are also presented in Table 5.4. EBBPSO-T with $v = 1.0$ is defined as the control method and a family of 10 hypotheses are established to be tested. The analysis shows a significant improvement of EBBPSO-T with $v = 1.0$ over dRepairGA-OOR (A11), dRepairRIGA (A9), Genocop (A14), dRepairRIGA-OOR (A12), and dRepairHyperM (A10). None of the remaining 5 hypotheses can be rejected, that means no significant difference between EBBPSO-T with $v = 1.0$ and the following algorithms: EBBPSO-T with $v = 2.1$, GenocopwUPCwNRR (A17), dRepairHyperM-OOR (A13), dGenocop (A18), and GSARepair (A19). In all comparisons, the level of significance considered was 10%. These results suggest that the two versions of EBBPSO-T (with $v = 1.0$ and $v = 2.1$) investigated in our experiments were both competitive compared to the best algorithms established in the literature for DCOPs, in particular they were very competitive against dGenocop (A18) and GSARepair (A19). Considering all test problems, the mean ranks of the two versions of EBBPSO-T are slightly better than the mean ranks of dGenocop (A18) and GSARepair (A19).

This subsection is concluded with an analysis of the performance obtained by EBBPSO-T in each group of problems of the G24 set, namely: (1) group-(fF,dC), (2) group-(fF,fC), (3) group-(dF,fC), (4) group-(dF,dC), (5) group-(dF,noC), and (6) group-(fF,noC). Table 5.5 summarizes the main results obtained from this analysis.

1. In the group of DCOPs with fixed objective function and dynamic constraints (fF,dC), EBBPSO-T presented a good performance when applied to solve the G24-7 problem, obtaining good ranks. In this problem, the size and the shape of the feasible regions change over time and a new global optimum is revealed whenever the environment is changed. The old optimum value is not changed because the objective function is static. However, the old optimum becomes infeasible because the size of the feasible regions decreases over time. On the other hand, although the ranks obtained by EBBPSO-T are poor when the algorithm was applied to solve the G24-3 problem, the offline errors obtained in this particular problem do not differ considerably from the offline errors obtained by the best algorithms for this problem. For the G24-3 problem, the size and shape of the feasible regions change over time and a new global optimum is revealed whenever the environment is changed. The old optimum value is not changed because the objective function is static, but the old optimum remains feasible because the size of the feasible regions increases over time. These results suggest that the proposed constraint-handling strategy performs considerably well with the moving of feasible regions. In addition, note that the two best algorithms (dRepairHyperM-OOR (A13) and dRepairHyperM (A10)) for this particular problem have poor mean ranks when compared to the mean ranks of EBBPSO-T, demonstrating that these algorithms have poor performance for most of the other problems investigated.

2. In the group of static constrained optimization problems (fF,fC), EBBPSO-T presented a good performance, obtaining considerable ranks for the problems G24-f and G24-3f. In this case, the mechanism to detect changes in the environment and the scheme of random immigrants are not triggered during de search and do not affect the performance of our approach in solving problems in this group.

3. In the group of DCOPs with dynamic objective function and fixed constraints (dF,fC), EBBPSO-T presented a moderate performance when applied to solve the problems in this group, obtaining moderate ranks. For the G24-1 problem, EBBPSO-T performs better than GSARepair (A19), while presenting a performance comparable to the performance of dGenocop (A18). For the problem G24-2, EBBPSO-T performs better than GSARepair (A19), but performs worse than dGenocop (A18). For the G24-6a,b,c, and d problems, both EBBPSO-T and dGenocop (A18) perform worse than GSARepair (A19). In general, the G24-6a,b,c, and d problems are characterized by islands of feasible regions disconnected by regions of infeasible solutions, which may be large infeasible regions depending on the problem at hand. The global optimum is switched from a feasible region to another over time. The moderate performance of the EBBPSO-T when tested on the G24-6a,b,c, and d problems suggests that our approach finds difficulties to travel easily through the infeasible regions separating disconnected feasible regions, especially when these infeasible regions are large. However, note that the problems G24-1 and G24-2 are also characterized by having two feasible regions disconnected by a region of infeasible solutions, but on these problems it is apparently easier to track the global optimum, which in turn is switched from one region to another over time. However, GSARepair (A19) does not perform as significantly when tested on these problems.

4. In the group of DCOPs with dynamic objective function and dynamic constraints (dF,dC), EBBPSO-T presented a moderate performance when applied to solve the problems in this group, obtaining in general moderate ranks. For the G24-4 problem, EBBPSO-T performs worse than GSARepair (A19), while presenting a performance comparable to dGenocop (A18). For the G24-5 problem, EBBPSO-T performs better than GSARepair (A19), while presenting a performance comparable to dGenocop (A18). Note that the best algorithm for the G24-4 problem (see dRepairHyperM (A10)) has a poor mean rank when compared to the mean ranks obtained by EBBPSO-T, dGenocop (A18), and GSARepair (A19), demonstrating that this algorithm presents a poor performance for most of the other problems investigated. An analogous observation can be made for the Genocop (A14) algorithm, which presented the best result for the G24-5 problem, but its mean rank shows a poor performance for most other of the problems. In fact, considering the complete set of test problems, the algorithms EBBPSO-T, dGenocop (A18), and GSARepair (A19) show a statistically significant improvement over dRepairHyperM (A10) and Genocop (A14). For the G24-3b problem, EBBPSO-T presents its worst performance.

5. For the group of dynamic unconstrained optimization problems (dF,noC), our approach preserves all proposed mechanisms except the constraint-handling strategy that is not applied for the problems in this group, since it is not necessary. In this group, our approach presented a considerable performance, presenting the best ranks for the problems G24-u, G24-2u, and G24-8a.

6. Our approach also presented a considerable performance when applied to solve the problem G24-uf, which is a static unconstrained optimization problem (fF, NoC). Once again, in this case, the mechanism to detect

Table 5.5: Overall performance of EBBPSO-T over the G24 set of DCOPs.

Group	G24 problem	Performance	
(fF,noC)	uf	Good	Mechanisms to deal with dynamic environments do not affect the performance.
(dF,noC)	u,2u,8a	Good	Good performance for the problems in this group. The strategy of mixture of search directions maintains the proper balance between diversification and intensification throughout the search process without loss of traceability of the global optimum.
Group	G24 problem	Performance	
(fF,fC)	f,3f	Good	Mechanisms to deal with dynamic environments do not affect the performance.
(dF,fC)	1,2,6abcd,8b	Moderate	Moderate performance for the problems in this group. Difficulties were observed to overcome infeasible regions between feasible regions, especially for the problems of the subgroup G24-6.
Group	G24 problem	Performance	
(fF,dC)	3,7	Good	Good performance for the problems in this group. The CH strategy performs considerably well with the moving of feasible regions. However, low convergence speed was observed for the G24-3 problem.
(dF,dC)	3b,4,5	Moderate	Moderate performance for the problems in this group. The worst performance was obtained for the G24-3b problem.

changes in the environment and the scheme of random immigrants are not triggered during de search and do not affect the performance of our approach in solving this problem.

5.3.3 Execution time

Table 5.6 shows the mean and the standard deviation of the execution times (in seconds) obtained by each swarm algorithm defined in Section 5.2, when these algorithms are tested on benchmark problems considered in our experimental study. The shortest times are shown in boldface. Comparisons between EBBPSO-T and its competitors are established in the last rows of Table 5.6, which shows summary statistics of the execution times of each algorithm, considering the complete set of problems. EBBPSO-T takes, on average, 13.54 seconds to solve a problem in the G24 set of DCOPs, which is approximately 1.23 seconds per environment, since in our experiments all problems are composed of 11 environments. The shortest times for all problems investigated were observed by GBBPSO-SR, since this algorithm uses a global topology. In general, the largest execution times were observed by the algorithm EBBPSO-T, since this algorithm uses a more complex dynamic rule to define the new position of a particle and also applies TOPSIS, after a ranking method, to select the solution that has the shortest distance from the positive ideal solution. Figure 5.2 shows a graphical representation of the results presented in Table 5.6.

5.4 Effect of varying the neighborhood topology

This section presents an analysis of the effect of the neighborhood topology on the performance of EBBPSO-T. Appendix B presents a discussion about different topologies commonly used in swarm algorithms. In this experimental study, four topologies were investigated: ring, clusters, Von Neumann, and random ring, where the last one topology is characterized by the fact that the neighbors of each particle are randomly selected at each iteration.

Table 5.7 shows the results of this analysis, considering EBBPSO-T with different neighborhood structures. The rank obtained in each problem by each EBBPSO-T with its respective topology is also shown and the best results are shown in boldface. Comparisons between topologies are summarized in the last rows of Table 5.7, which shows summary statistics of the offline errors considering the complete set of problems. The topology that presented the worst performance was the random ring, followed by the cluster topology. The best performances were observed by ring and Von Neumann topologies, with the Von Neumann topology apparently presenting

Table 5.6: Mean and standard deviation of execution times (in seconds) for four swarm algorithms tested on 18 problems of the G24 set.

Algorithm	LBBPSO-SR	GBBPSO-SR	EBBPSO-SR	EBBPSO-T ($\nu = 1.0$)
Problem	Mean (SD)	Mean (SD)	Mean (SD)	Mean (SD)
G24-u	2.789 (0.018)	**2.231** (0.016)	3.027 (0.028)	3.048 (0.021)
G24-1	12.831 (0.043)	**6.502** (0.029)	13.105 (0.037)	16.498 (0.043)
G24-f	12.133 (0.057)	**6.112** (0.028)	12.400 (0.040)	15.617 (0.043)
G24-uf	2.225 (0.042)	**1.447** (0.016)	1.669 (0.023)	1.779 (0.037)
G24-2	13.663 (0.048)	**6.954** (0.024)	14.686 (0.992)	17.266 (0.068)
G24-2u	3.778 (0.020)	**2.231** (0.016)	4.019 (0.020)	4.022 (0.020)
G24-3	13.325 (0.055)	**6.617** (0.027)	13.553 (0.054)	16.860 (0.057)
G24-3b	13.843 (0.042)	**6.869** (0.026)	13.990 (0.041)	17.320 (0.061)
G24-3f	12.491 (0.053)	**6.221** (0.030)	12.724 (0.073)	15.907 (0.082)
G24-4	13.605 (0.046)	**6.809** (0.033)	13.843 (0.040)	17.187 (0.047)
G24-5	14.327 (0.048)	**7.243** (0.036)	14.790 (0.062)	17.977 (0.061)
G24-6a	13.124 (0.096)	**6.573** (0.104)	13.312 (0.041)	16.668 (0.065)
G24-6b	9.049 (0.056)	**4.852** (0.019)	9.262 (0.023)	12.564 (0.049)
G24-6c	11.596 (0.035)	**5.910** (0.022)	11.803 (0.046)	15.323 (0.049)
G24-6d	14.001 (0.084)	**6.957** (0.025)	14.125 (0.054)	17.705 (0.059)
G24-7	13.182 (0.057)	**6.570** (0.064)	13.354 (0.048)	16.705 (0.054)
G24-8a	4.272 (0.029)	**2.216** (0.024)	3.974 (0.021)	4.002 (0.021)
G24-8b	13.681 (0.067)	**6.905** (0.028)	13.835 (0.061)	17.248 (0.101)
Minimum	2.225	1.447	1.669	1.779
1st quartile	9.686	5.117	9.897	13.254
Median	12.977	6.536	13.209	16.583
Mean	10.773	5.512	10.971	13.539
3rd quartile	13.649	6.854	13.841	17.233
Maximum	14.327	7.243	14.790	17.977
SD	4.310	1.992	4.485	5.820

significant improvements in some of the problems investigated. Figure 5.3 shows a graphical representation of the results presented in Table 5.7.

5.5 Effect of varying the population size

This section presents an analysis of the effect of the population size on the performance of EBBPSO-T. Table 5.8 shows the results of this analysis considering EBBPSO-T with different population sizes. The population size affects the performance of the algorithm. The offline errors and the best-errors-before-change decrease with increasing population size, but at the cost of execution times almost 5 times higher on average, when the population size increases from 25 to 100 particles.

5.6 Effect of varying the parameter ν

This section presents an analysis of the effect of the parameter ν on the performance of EBBPSO-T. Fig. 5.4 shows the results of this analysis when EBBPSO-T is applied to solve the G24-4 problem. As it can be seen, the choice of ν can influence significantly the performance of the algorithm. Indeed, ν can be fitted according to the problem at hand. For instance, considering the G24-4 problem, a satisfactory performance can be obtained with ν between 1.0 and 1.4. However, experimental results show that ν between 0.8 and 2.2 is an appropriate setting for many problems investigated, taking into consideration the fixed number of evaluations.

Table 5.7: Mean and standard deviation of offline errors for EBBPSO-T ($v = 1.0$) with different neighborhood topologies.

$\mathcal{N}$	Ring	Clusters	Von Neumann	Random Ring
Problem	Mean (SD) [Rank]	Mean (SD) [Rank]	Mean (SD) [Rank]	Mean (SD) [Rank]
G24-u	0.056 (0.005) [2]	0.061 (0.005) [4]	**0.054** (0.005) [1]	0.058 (0.006) [3]
G24-1	0.084 (0.041) [3]	0.083 (0.019) [2]	**0.077** (0.017) [1]	0.138 (0.164) [4]
G24-f	0.014 (0.005) [2.5]	0.014 (0.005) [2.5]	**0.013** (0.005) [1]	0.019 (0.008) [4]
G24-uf	0.009 (0.003) [3]	**0.007** (0.003) [1]	0.008 (0.003) [2]	0.010 (0.003) [4]
G24-2	**0.136** (0.013) [1]	0.148 (0.019) [3]	0.137 (0.015) [2]	0.156 (0.044) [4]
G24-2u	**0.045** (0.001) [1.5]	0.046 (0.002) [3]	**0.045** (0.001) [1.5]	0.047 (0.003) [4]
G24-3	**0.032** (0.005) [1.5]	0.043 (0.007) [4]	**0.032** (0.006) [1.5]	0.039 (0.009) [3]
G24-3b	0.104 (0.015) [2]	0.123 (0.025) [3]	**0.102** (0.019) [1]	0.136 (0.059) [4]
G24-3f	**0.012** (0.005) [1]	0.014 (0.007) [2.5]	0.014 (0.006) [2.5]	0.017 (0.008) [4]
G24-4	0.138 (0.022) [2]	0.141 (0.025) [3]	**0.134** (0.032) [1]	0.155 (0.051) [4]
G24-5	0.126 (0.019) [2]	0.139 (0.022) [3]	**0.123** (0.019) [1]	0.160 (0.043) [4]
G24-6a	0.116 (0.099) [3]	0.080 (0.018) [2]	**0.069** (0.011) [1]	0.155 (0.148) [4]
G24-6b	**0.184** (0.043) [1]	0.205 (0.055) [3]	0.189 (0.044) [2]	0.230 (0.066) [4]
G24-6c	0.251 (0.061) [2]	0.275 (0.069) [3]	**0.235** (0.058) [1]	0.298 (0.096) [4]
G24-6d	0.312 (0.203) [2]	0.373 (0.213) [3]	**0.256** (0.147) [1]	0.405 (0.215) [4]
G24-7	0.045 (0.009) [2]	0.049 (0.010) [3]	**0.043** (0.010) [1]	0.054 (0.018) [4]
G24-8a	0.194 (0.022) [2]	0.227 (0.034) [4]	**0.192** (0.022) [1]	0.214 (0.033) [3]
G24-8b	0.312 (0.086) [2]	0.341 (0.070) [3]	**0.286** (0.048) [1]	0.389 (0.072) [4]
Minimum	0.009	0.007	0.008	0.010
1st quartile	0.045	0.047	0.043	0.049
Median	0.110	0.103	0.090	0.147
Mean	0.121	0.132	0.112	0.149
3rd quartile	0.173	0.191	0.176	0.200
Maximum	0.312	0.373	0.286	0.405
SD	0.097	0.112	0.088	0.121
Mean rank	1.97	2.89	1.31	3.83
$W/L/T$	-	14/3/1	3/13/2	18/0/0

Table 5.8: Mean of offline errors, best-errors-before-change, and execution times for EBBPSOT ($v = 1.0$) with different population sizes.

Problem	Offline error			Best-error-before-change			Execution time (s)		
K	25	50	100	25	50	100	25	50	100
G24-1	0.084	0.071	0.059	5.260e-02	8.954e-03	1.255e-03	16.376	36.360	71.712
G24-3	0.032	0.029	0.027	4.570e-04	1.467e-04	5.047e-05	16.829	37.662	74.061
G24-4	0.138	0.107	0.087	1.118e-02	1.505e-03	3.954e-04	17.097	38.244	74.986
G24-6a	0.116	0.091	0.074	1.729e-03	6.687e-04	2.237e-04	16.632	36.962	72.199
G24-6c	0.251	0.163	0.096	2.188e-03	6.022e-04	1.672e-04	15.170	33.830	65.684
G24-7	0.045	0.039	0.036	1.985e-03	3.528e-04	9.863e-05	16.672	37.222	73.042

5.7 Application

This section presents a real-world application as a case study. EBBPSO-T is applied for solving the dynamic economic dispatch problem, which is one of the major optimization problems in dynamic environments in the field of power system operation.

5.7.1 The economic dispatch problem

The economic dispatch problem (ED problem) is used to determine the optimal combination of power outputs of all generating units to minimize the total fuel cost, while satisfying the load demand and also physical and operational

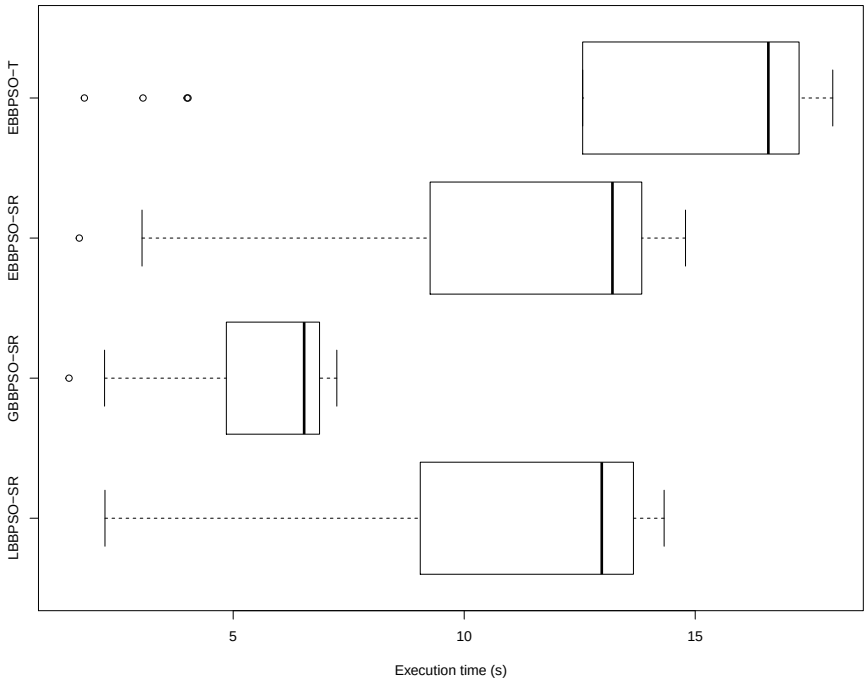

Figure 5.2: Execution time for four swarm algorithms tested on the G24 set.

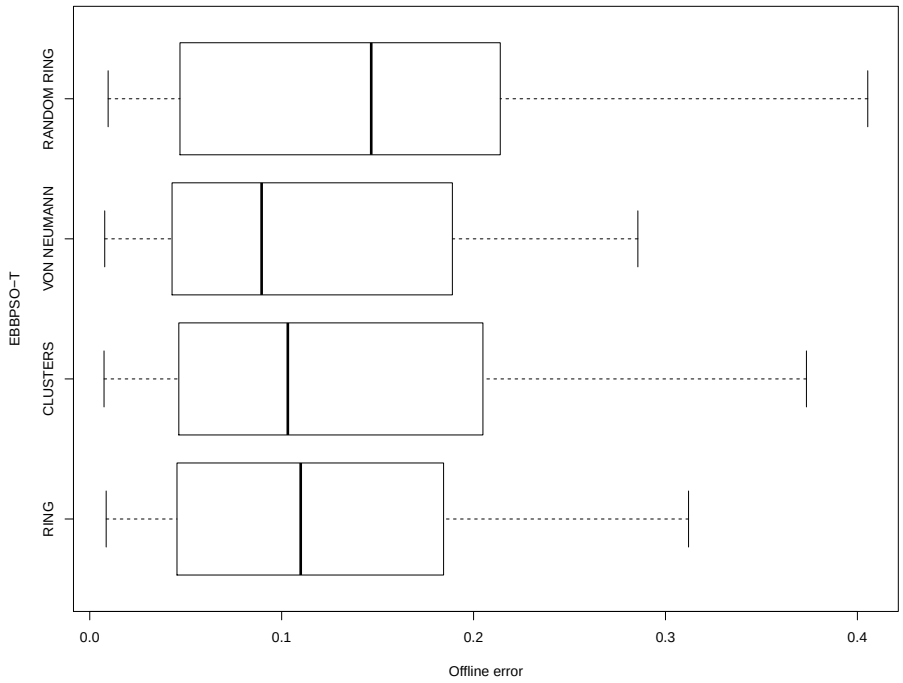

Figure 5.3: Offline error for EBBPSO-T with different neighborhood topologies tested on the G24 set.

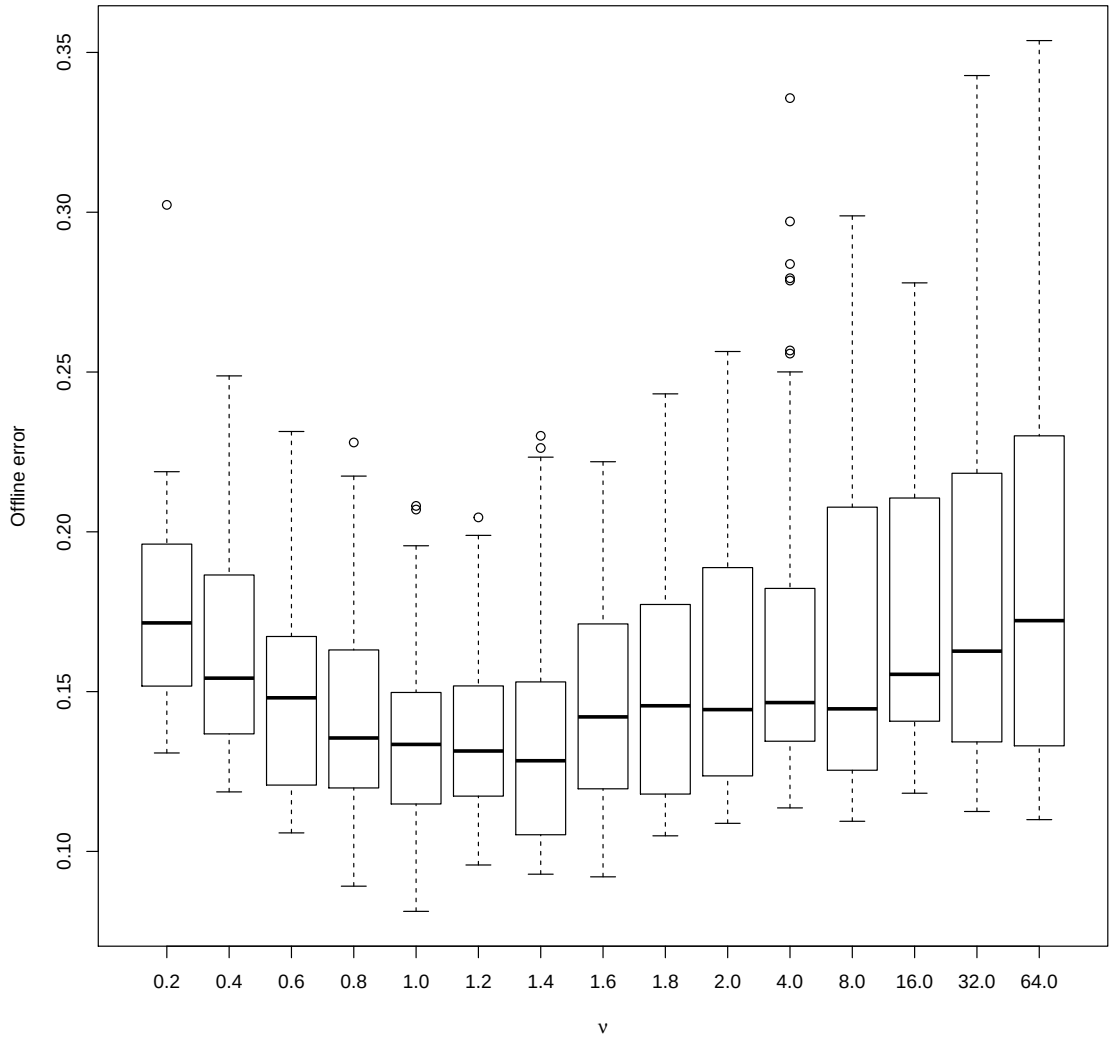

Figure 5.4: Effect of the parameter ν on the performance of EBBPSO-T when the algorithm is applied to solve the G24-4 problem.

constraints (Abouheaf, Lee & Lewis, 2013; Zhang et al., 2014b). The ED problem for a N-unit system is defined as:

$$\begin{array}{rll} \min & FC(P_1,\ldots,P_N) & \\ \text{subject to} & \sum_{i=1}^{N} P_i - P_L - P_D = 0 & \\ & P_i^{\min} \leq P_i \leq P_i^{\max} & i = 1,\ldots,N. \end{array} \tag{5.4}$$

Generally, the total fuel cost function FC to be minimized can be expressed as $FC(P_1,\ldots,P_N) = \sum_{i=1}^{N} FC_i(P_i)$, where $FC_i(P_i)$ (in $\$/h$) is the fuel cost corresponding to power output P_i of the generator i. This function is defined by

$$FC_i(P_i) = a_i P_i^2 + b_i P_i + c_i + \left| e_i \sin[f_i(P_i^{\min} - P_i)] \right| \tag{5.5}$$

where a_i, b_i, c_i, e_i, and f_i are cost coefficients of the generator i. The equality constraint represents the active power balance, i.e., the generated power should be equal to the sum of the load demand (P_D) with the transmission losses (P_L), given by $P_L = \sum_{i=1}^{N} \sum_{j=1}^{N} P_i B_{ij} P_j$, where B_{ij} are loss coefficients. Inequality constraints must be also satisfied. The generated power of the generator i should be laid between max-min limits ($P_i^{\max}$ and $P_i^{\min}$ respectively).

5.7.2 The dynamic economic dispatch problem

Abouheaf, Lee & Lewis (2013) introduced a dynamic formulation of the ED problem. This dynamic formulation was developed to optimally allocate the change in the total active load demand to the generating units. It is based on two assumptions:

Assumption 1. Assume that for a total active load demand $P_{D(t-1)}$ at time $t-1$, each generating unit i has a power generation $P_{i,(t-1)}$. Thus, the change in the generated power of each generator i, due to the change in the total active load demand $\delta P_{Dt} = P_{Dt} - P_{D(t-1)}$ at time t, is denoted by $\delta P_{it} = P_{it} - P_{i(t-1)}$. Therefore

$$\sum_{i=1}^{N} \delta P_{it} = (P_{Lt} - P_{L(t-1)}) + \delta P_{Dt} \tag{5.6}$$

for all $t = 1, 2, \ldots, T'$.

Assumption 2. One assumption that prevailed for simplifying the ED problem in many of the earlier research is that the adjustments of the power outputs are instantaneous. However, under practical circumstances, ramp rate limit restricts the operating range of all the online units for adjusting the operation between two periods. The generation may increase or decrease, but it is bounded by corresponding upper and lower ramp rate limits (UR_i and DR_i). Therefore

$$P_{it} - P_{i(t-1)} \leq UR_i \tag{5.7}$$

and

$$P_{i(t-1)} - P_{it} \leq DR_i \tag{5.8}$$

for all $i = 1, \ldots, N$ and $t = 1, \ldots, T'$.

Given a set of N generators with initial outputs P_{i0}, the dynamic economic dispatch problem (DED problem) can be defined as follows:

$$\begin{array}{rll} \min & \sum_{i=1}^{N} FC_i(P_{i(t-1)} + \delta P_{it}) & \\ \text{subject to} & \sum_{i=1}^{N} (P_{i(t-1)} + \delta P_{it}) - P_{Lt} - P_{Dt} = 0 & \text{for all } t \\ & \sum_{i=1}^{N} \delta P_{it} - (P_{Lt} - P_{L(t-1)}) - \delta P_{Dt} = 0 & \text{for all } t \\ & -DR_i \leq \delta P_{it} \leq UR_i & \text{for all } i,t \\ & P_i^{\min} - P_{i(t-1)} \leq \delta P_{it} \leq P_i^{\max} - P_{i(t-1)} & \text{for all } i,t \end{array} \tag{5.9}$$

where $t = 1, 2, \ldots, T'$,

$$P_{Lt} = \sum_{i=1}^{N} \sum_{j=1}^{N} (P_{i(t-1)} + \delta P_{it}) B_{ij} (P_{j(t-1)} + \delta P_{jt}) \tag{5.10}$$

and

$$P_{L(t-1)} = \sum_{i=1}^{N} \sum_{j=1}^{N} P_{i(t-1)} B_{ij} P_{j(t-1)}. \tag{5.11}$$

The DED problem can be seen as an optimization problem such as discussed in Sections 1.2 and 2.1. Both the objective function and the constraints are dynamic. Moreover, the boundary constraints of this problem are also dynamic. In our approach, equality constraints are transformed into inequality constraints, for instance,

$$\left| \sum_{i=1}^{N} (P_{i(t-1)} + \delta P_{it}) - P_{Lt} - P_{Dt} \right| \leq \varepsilon \tag{5.12}$$

where ε is a tolerance allowed. Example 5.1 illustrates the use of our approach.

Example 5.1. Consider a 5-unit system. The valve-point effects, transmission losses, ramp rate constraints, and generation limits are considered in this system. The prohibited operating zones are not considered in this test case. The data for this system are presented in Table 5.9, including the B-matrix coefficients. This problem is solved using EBBPSO-T with $K = 25$ particles, a neighborhood system $\mathcal{N}$ defined by a ring topology, and $\nu = 1.0$. The change frequency used was $\Delta = 100$ iterations (2500 function evaluations) per environment. Table 5.10 shows the hourly load demand P_{Dt} and the best schedule obtained by EBBPSO-T for this system. The solution shown in Table

5.10 has a total cost of $44491.45 and a total loss of 195.55 MW, which means 1.32% of the total generated power in the scheduled horizon. The execution time (CPU time) to obtain the solution was of 152.41 s (or 2.54 min). For comparison purposes, Zhang et al. (2014b) presented a hybrid BBPSO to solve DED problems and reported a average total cost of $43737 and a CPU time of 1.48 min for the 5-unit system.

Table 5.9: Generating units' characteristics for 5-unit system.

-	a	b	c	e	f	$P_{\min}$	$P_{\max}$	UR	DR	-
Unit	\$/MW^{2}h	\$/MWh	\$/h	\$/h	1/MW	MW	MW	MW/h	MW/h	B-matrix ($\times 10^{-5}$)
1	0.0080	2.0	25	100	0.042	10	75	30	30	4.9 1.4 1.5 1.5 2.0
2	0.0030	1.8	60	140	0.040	20	125	30	30	1.4 4.5 1.6 2.0 1.8
3	0.0012	2.1	100	160	0.038	30	175	40	40	1.5 1.6 3.9 1.0 1.2
4	0.0010	2.0	120	180	0.037	40	250	50	50	1.5 2.0 1.0 4.0 1.4
5	0.0015	1.8	40	200	0.035	50	300	50	50	2.0 1.8 1.2 1.4 3.5

Table 5.10: Best generator schedule using EBBPSO-T for 5-unit system.

Hour	P_{1t}	P_{2t}	P_{3t}	P_{4t}	P_{5t}	$\sum P_{it}$	Demand	Loss	Cost
t	MW	MW	MW	MW	MW	MW	MW	MW	\$
01	11.23	20.36	30.08	123.82	228.41	413.89	410	3.89	1245.58
02	17.64	27.04	34.99	129.94	229.82	439.42	435	4.42	1414.19
03	18.44	34.18	71.14	124.26	231.95	479.98	475	4.98	1658.59
04	24.87	36.35	95.36	146.58	232.87	536.04	530	6.04	1889.69
05	31.39	54.26	119.27	128.79	230.86	564.57	558	6.57	1849.64
06	37.17	79.08	121.60	142.91	235.06	615.81	608	7.81	2068.95
07	22.04	76.20	116.58	190.73	228.80	634.36	626	8.36	2019.94
08	13.25	97.63	112.18	210.23	229.97	663.26	654	9.26	1813.50
09	42.22	104.34	113.35	210.57	229.71	700.19	690	10.19	2016.01
10	64.21	97.48	112.48	210.61	229.78	714.56	704	10.56	2010.45
11	73.66	99.45	118.21	210.15	229.55	731.02	720	11.02	2052.22
12	74.54	124.50	112.62	209.44	230.63	751.73	740	11.73	2191.49
13	66.91	98.45	112.39	210.07	226.74	714.55	704	10.55	2013.21
14	51.59	97.98	111.40	209.74	229.46	700.17	690	10.17	1989.18
15	33.50	97.46	111.96	192.26	227.97	663.14	654	9.14	1998.40
16	11.44	91.09	111.98	147.38	225.38	587.26	580	7.26	1819.59
17	11.18	87.10	113.12	123.99	229.30	564.68	558	6.68	1632.96
18	10.54	98.07	110.83	167.43	229.08	615.96	608	7.96	1872.97
19	12.30	99.58	112.44	209.97	228.98	663.26	654	9.26	1807.66
20	41.43	119.71	112.55	211.52	229.46	714.67	704	10.67	2125.64
21	38.91	97.86	112.81	210.83	229.49	689.90	680	9.90	1955.48
22	10.83	98.55	109.83	165.64	228.03	612.88	605	7.88	1877.26
23	11.11	98.21	71.71	124.23	227.89	533.16	527	6.16	1665.07
24	11.63	72.50	32.90	123.48	227.60	468.11	463	5.11	1503.79
Total	742.01	2007.42	2381.77	4134.55	5506.81	14772.55	14577	195.55	\$44491.45

Chapter 6

Conclusion

6.1 Summary and conclusions

Dynamic constrained optimization problems (DCOPs) are difficult to solve because both the objective function and the constraints can vary over time. For these problems the goal is to find an optimal solution and track it as nearly as possible over time. Changes in the environment of the problem must be taken into account during the optimization process. The literature on swarm and evolutionary computation has documented the need to design algorithms containing specific mechanisms to deal with DCOPs. Swarm algorithms have shown potential to solve constrained optimization problems in static environments, but little attention has been paid to adapt these algorithms to deal with DCOPs.

This thesis introduced a new algorithmic methodology based on swarm computation for solving DCOPs. The proposed algorithm is named as entropy-based bare bone particle swarm (EBBPSO-T for short) and it applies the idea of mixture of search directions using the Shannon's index of diversity as a factor to balance the influence of the global-best and local-best search directions. This dynamic rule works as a mechanism to maintain and introduce diversity during the search process. A constraint-handling strategy is also proposed, which treats the objective function and the constraints separately using a ranking method with selection based on TOPSIS to obtain the best solution within a specific population of candidate solutions. This constraint-handling strategy balances the objective function against the degree of constraint violation in such a way that neither of them is dominant. Mechanisms to detect changes in the environment and to update the particles' memories are also implemented into the proposed algorithm. The mechanism to detect changes in the environment is based on a fixed set of detectors uniformly distributed in the search space. When a change in the environment is detected, a random-immigrants scheme acts to introduce diversity. Part of the swarm is replaced with randomly generated particles and this strategy acts only on the first iteration of the changed environment. Re-evaluation of fitness values and the ranking method combined with TOPSIS are also used to update the particles' memories. It is important to emphasize that these mechanisms do not act independently. They operate related to each other to tackle problems such as: loss of diversity due to convergence, outdated memories due to changes in the environment, and loss of the tracking ability to search a new optimal solution of a changed environment.

An experimental analysis was conducted to investigate empirically the performance of EBBPSO-T. The G24 set of DCOPs was used in this experimental study. These problems contain characteristics that are representative of real-world DCOPs. Experimental results show the suitability of the proposed algorithm in terms of effectiveness to find good solutions for most of the benchmark problems investigated. Non-parametric statistical tests indicated that EBBPSO-T shows a statistically significant improvement when compared with different GA-based algorithms (dRepairGA-OOR, dRepairRIGA, Genocop, dRepairRIGA-OOR, and dRepairHyperM), while EBBPSO-T shows a competitive performance when compared with the following algorithms: GenocopwUPCwNRR, dRepairHyperM-OOR, dGenocop, and GSARepair. These results suggest that EBBPSO-T is competitive compared to the best algorithms established in the literature for DCOPs, in particular it is very competitive against dGenocop and GSARepair.

The proposed constraint-handling strategy performed considerably well with the moving of feasible regions in DCOPs with fixed objective function and dynamic constraints, both when the size of the feasible regions decreases over time and makes the old optimum an infeasible solution or when the size of the feasible regions increases over time and reveals a new global optimum, keeping the old optimum value unchanged.

Our approach was tested on dynamic unconstrained optimization problems. In this case, the algorithm preserves all proposed mechanisms except the constraint-handling strategy that is not applied for these problems. Our approach presented a good performance for these problems, confirming that the strategy of mixture of search directions performs considerably well with the tracking of the optimal solution, maintaining diversity during the search process.

EBBPSO-T also presented a good performance in static constrained optimization problems. In this case, the mechanism to detect changes in the environment and the scheme of random immigrants are not triggered during the search and do not affect the performance of our approach in solving these problems.

The combined effect of the mechanisms implemented into EBBPSO-T provides an algorithm with ability to maintain a proper balance between exploration and exploitation at any stage of the search process, without losing the tracking ability to search an optimal solution that is changing over time. EBBPSO-T presented a moderate performance when applied to solve DCOPs with dynamic objective function and dynamic constraints. Moderate performance was also observed when EBBPSO-T was applied to solve DCOPs with dynamic objective function and fixed constraints.

Some limitations of our approach were observed in our experimental study. EBBPSO-T presented difficulties to travel between feasible regions in order to track the global optimum that is switched from one region to another, especially when these feasible regions are islands disconnected by large regions of infeasible solutions. Problems where this difficulty was sharply observed were: G24-3b, G24-4, and G24-6d. In addition, low convergence speed was observed when EBBPSO-T was applied to solve the G24-3 problem that expose a new global optimum without changing the objective value of the previous optimum.

The mechanisms implemented into EBBPSO-T can also be applied to other evolutionary algorithms to solve optimization problems in dynamic environments. Another feature of our approach is that it is also very flexible, allowing the inclusion of other distributions with heavy tails besides the t-distribution. From a suitable choice of the mixing density, a rich class of continuous, symmetric and unimodal distributions can be applied in our dynamic rule to update the position of a particle in order to maintain a good level of population diversity during the search process.

6.2 Future studies

The topics studied in this thesis open up possibilities for future studies.

- The proposed approach in this thesis to solve DCOPs can be extended to deal with dynamic multi-objective optimization problems (DMOOPs). These problems can be defined as:

$$\begin{array}{rll} \min & \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x},t),\ldots,f_K(\mathbf{x},t))' & \text{where } \mathbf{x} \in \mathbb{R}^D \text{ and } t \in T = \{0,1,2,\ldots\} \\ \text{subject to} & L_d \leq x_d \leq U_d & d = 1,\ldots,D \end{array} \tag{6.1}$$

  where $\mathbf{x}$ is the vector of the D decision variables and $\mathbf{f}(\mathbf{x})$ is the vector of K functions to be minimized over the feasible space $\mathbb{F}(t) = \mathbb{S} = [\mathbf{L},\mathbf{U}] = [L_1,U_1] \times \cdots \times [L_D,U_D]$. Two approaches can be followed:

  1. Reformulate the dynamic multi-objective optimization problem as a dynamic constrained optimization problem using the $\boldsymbol{\varepsilon}$-constraint method as described in Section C.4 (see Appendix C). This approach may be of potential use for DMOOPs with low-dimensional objective space, usually for $K = 2$ or 3.
  2. Alternatively, the dynamic multi-objective optimization problem can be addressed directly using Pareto dominance (see again Section C.4) to guide the search and return a set of nondominated solutions as result. This approach carry out some type of vector ranking scheme based on the dominance relationships between different solutions in the population. As a result, every solution has a ranking, which serves as an indicator for measuring how important each solution is. These ranking methods contribute to the selection of solutions with high ranking over solutions with low ranking, which determines the solutions that can be memorized in each given generation of a swarm algorithm or the solutions that can survive into the next generation of an evolutionary algorithm. Different vector ranking schemes can be investigated, including recent approaches such as the scheme proposed by Zhou, Chen & Zhang (2017).

  In both approaches, the strategy of mixture of search directions using the Shannon's index of diversity as a weighting factor may be of potential use to maintain and introduce diversity during the search process, which

is important to obtain a set of non-dominated solutions as diverse as possible. In addition, the estimation procedure of the diversity index can be applied in the objective space that is K-dimensional.

- Luchi & Krohling (2015) and Luchi (2016) introduced an estimation procedure of the Shannon's index of diversity that uses both the fitness values $f(\mathbf{x})$ and the satisfaction levels of the constraints $\mu(\mathbf{x})$ associated with solutions in the population through the product $f(\mathbf{x}) \cdot \mu(\mathbf{x})$. This product rule may be of potential use for DCOPs in order to incorporate information related with the degree of violation of constraints that change over time in dynamic environments.

Bibliography

Abanto-Valle, C.; Bandyopadhyay, D.; Lachos, V.; Enriquez, I. Robust Bayesian analysis of heavy-tailed stochastic volatility models using scale mixtures of normal distributions. **Computational Statistics and Data Analysis**, v. 54, pp. 2883-2898, 2010.

Abouheaf, M.; Lee, W.-J.; Lewis, F. Dynamic formulation and approximation methods to solve economic dispatch problems. **IET Generation, Transmission and Distribution**, 7(8), pp. 866-873, 2013.

Andrews, D.; Mallows, C. Scale mixtures of normal distributions. **Journal of the Royal Statistical Society. Series B (Methodological)**, v. 36(1), pp. 99-102, 1974.

Aragón, V.; Esquivel, S.; Coello Coello, C. Artificial immune system for solving dynamic constrained optimization problems. In Alba et al. (Eds.): **Metaheuristics for Dynamic Optimization**, SCI 433. Heidelberg: Springer-Verlag, 2013. pp. 225-263.

Back, T.; Schwefel, H.-P. An overview of evolutionary algorithms for parameter optimization. **Evolutionary Computation**, v. 1(1), pp. 1-23, 1993.

Bazaraa, M.; Sherall, H.; Shetty, C. **Nonlinear programming: theory and algorithms**, 2nd ed. New York: John Wiley and Sons, 1993.

Behzadian, M.; Otaghsara, S.K.; Yazdani, M.; Ignatius, J. A state of the art survey of TOPSIS applications. **Expert Systems with Applications**, v. 39, pp. 13051-13069, 2012.

Beyer, H.-G.; Schwefel, H.-P. Evolution strategies: a comprehensive introduction. **Natural Computing**, v. 1, pp. 3-52, 2002.

Beyer, H.-G.; Finck, S. On the design of constraint covariance matrix self-adaptation evolution strategies including a cardinality constraint. **IEEE Transactions on Evolutionary Computation**, v. 16(4), pp. 578-596, 2012.

Blackwell, T. Particle swarms and population diversity. **Soft Computing**, v. 9(11), pp. 793-802, 2005.

Blackwell, T.; Branke, J. Multiswarms, exclusion, and anti convergence in dynamic environments. **IEEE Transactions on Evolutionary Computation**, v. 10(4), pp. 459-472, 2006.

Blackwell, T. A study of collapse in bare bones particle swarm optimization. **IEEE Transactions on Evolutionary Computation**, v. 16(3), pp. 354-372, 2012.

Branke, J. Memory enhanced evolutionary algorithms for changing optimization problems. In **Proceedings of IEEE Congress on Evolutionary Computation**, 1999. pp. 1875-1882.

Bratton, D; Kennedy, J. Defining a standard for particle swarm optimization. In **Proceedings of IEEE Swarm Intelligence Symposium**, 2007. pp. 120-127.

Bu, C.; Luo, W.; Yue, L. Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies. **IEEE Transactions on Evolutionary Computation**, v. 21(1), pp. 14-33, 2017.

Butturi-Gomes, D.; Petrere Junior, M.; Giacomini, H.; De Marco Junior, P. Computer intensive methods for controlling bias in a generalized species diversity index. **Ecological Indicators**, v. 37, pp. 90-98, 2014.

Campos, M; Krohling, R.A. Hierarchical bare bones particle swarm for solving constrained optimization problems. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2013. pp. 805-812.

Campos, M.; Krohling, R.A.; Enriquez, I. Bare bones particle swarm optimization with scale matrix adaptation. **IEEE Transactions on Cybernetics**, v. 44(9), pp. 1567-1578, 2014.

Campos, M.; Krohling, R.A. Bare bones particle swarm with scale mixtures of Gaussians for dynamic constrained optimization problems. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2014. pp. 202-209.

Campos, M.; Krohling, R.A. Entropy-based bare bones particle swarm for dynamic constrained optimization. **Knowledge-Based Systems** v. 97, pp. 203-223, 2016.

Cerny, V. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. **Journal of Optimization Theory and Applications**, v. 45, pp. 41-51, 1985.

Chao, A.; Shen, T.-J. Nonparametric estimation of Shannon's index of diversity when there are unseen species in sample. **Environmental and Ecological Statistics**, v. 10, pp. 429-443, 2003.

Choy, S.; Chan, J. Scale mixtures of distributions in statistical modelling. **Australian and New Zealand Journal of Statistics**, v. 50(2), pp. 135-146, 2008.

Clerc, M.; Kennedy, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. **IEEE Transactions on Evolutionary Computation**, v. 6(1), pp. 58-73, 2002.

Coello, C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. **Computer Methods in Applied Mechanics and Engineering**, v. 191, pp. 1245?1287, 2002.

Corriveau, G.; Guilbautl, R.; Tahan, A.; Sabourin, R. Review and study of genotypic diversity measures for real-coded representations. **IEEE Transactions on Evolutionary Computation**, v. 16(5), pp. 695-710, 2012.

Cover, T; Thomas, J. **Elements of information theory, 2nd Edition**. New Jersey: Wiley, 2006.

Darwin, C. **On the origin of species by means of natural selection or the preservation of favoured races in the struggle for life**. London: John Murray (Albemarle Street), 1859.

Deb, K. An efficient constraint handling method for genetic algorithms. **Computer Methods in Applied Mechanics and Engineering**, v. 186, pp. 311-338, 2000.

Demsar, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, v. 7, pp. 1-30, 2006.

Derrac, J.; Garcia, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v. 1, pp. 3-18, 2011.

Dorigo, M. Optimization, learning and natural algorithms. **Ph.D. thesis**, Politecnico di Milano, Italy, 1992.

Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: optimization by a colony cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics**, v. 26(1), pp. 29-41, 1996.

Dorigo, M.; Birattari, M; Stützle, T. Ant colony optimization. **IEEE Computational Intelligence Magazine**, November, pp. 28-39, 2006.

Du, W.; Lin, B. Multi-strategy ensemble particle swarm optimization for dynamic optimization. **Information Sciences**, v. 178, pp. 3096-3109, 2008.

Eberhart, R.; Shi, Y. Tracking and optimizing dynamic systems with particle swarms. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2001, pp. 94-97.

Engelbrecht, A. **Computational intelligence: an introduction**. New Jersey: Wiley, 2007.

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

Fu, H.; Lewis, P.; Sendhoff, B.; Tang, K.; Yao, X. What are dynamic optimization problems?. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2014, pp. 1550-1557.

Good, I. The population frequencies of species and the estimation of population parameters. **Biometrika**, v. 40, pp. 237-264, 1953.

Hadj-Alouane, A. B.; Bean, J. C. A genetic algorithm for the multiple-choice integer program. **Operations Research**, v. 45, pp. 92-101, 1997.

Hastings, W. Monte Carlo sampling methods using Markov chains and their applications. **Biometrika**, v. 57, pp. pp. 97-109, 1970.

Hausser, J.; Strimmer, K. Entropy inference and the James-Stein estimator with application to nonlinear gene association networks. **Journal of Machine Learning Research**, v. 10, pp. 1469-1484, 2009.

He, Q.; Wang, L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. **Applied Mathematics and Computation**, v. 186(2), pp. 1407-1422, 2007.

Hinderer, K.; Rieder, U.; Stieglitz, M. **Dynamic optimization: deterministic and stochastic models**. Switzerland: Springer International Publishing, 2016.

Ho, P.; Shimizu, K. Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. **Information Sciences**, v. 177, pp. 2985-3004, 2007.

Hoffmeister, F.; Sprave, J. Problem-independent handling of constraints by use of metric penalty functions. In **Proceedings of the Fifth Annual Conference on Evolutionary Programming**, 1996. pp. 289-294.

Holland, J. **Adaptation in natural and artificial systems**. Ann Arbor: University of Michigan Press, 1975.

Homaifar, A.; Lai, S; Qi, X. Constrained optimization via genetic algorithms. **Simulation**, v. 62(4), pp. 242-254, 1994.

Horvitz, D.; Thompson, D. A generalization of sampling without replacement from a finite universe. **Journal of the American Statistical Association**, v. 47, pp. 663-685, 1952.

Hsieh, H.-I.; Lee, T.-S. A modified algorithm of bare bones particle swarm optimization. **International Journal of Computer Science**, v. 7(6), pp. 12-17, 2010.

Huang, J. Combining Entropy Weight and TOPSIS Method for Information System Selection. In **Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems, CIC2008**, 2008. pp. 1281-1284.

Hwang, C.L.; Yoon, K. **Multiple Attribute Decision Making: Methods and applications**. New York: Springer-Verlag, 1981.

Izsák, J. Parameter dependence of correlation between the Shannon index and members of parametric diversity index family. **Ecological Indicators**, v. 7, pp. 181-194, 2007.

Jiang, M.; Luo, Y.; Yang, S. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. **Information Processing Letters**, 102, pp. 8-16, 2007.

Jin, Y.; Branke, J. Evolutionary optimization in uncertain environments a survey. **IEEE Transactions on Evolutionary Computation**, v. 9(3), pp. 303-317, 2005.

Joines, J.; Houck, C. On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GAs. In **Proceedings of IEEE Congress on Evolutionary Computation**, 1994. pp. 579-584.

Jordehi, A. A review on constraint handling strategies in particle swarm optimisation. **Neural Computing and Applications**, v. 26(6), pp. 1265-1275, 2015.

Kaleli, C. An entropy-based neighbor selection approach for collaborative filtering. **Knowledge-Based Systems**, v. 56, pp. 273-280, 2014.

Kennedy, J. Bare bones particle swarms. In **Proceedings of IEEE Swarm Intelligence Symposium**, 2003. pp. 80-87.

Kennedy, J.; Eberhart, R. Particle swarm optimization. In **Proceedings of IEEE International Conference on Neural Networks**, 1995. pp. 1941-1948.

Kennedy, J.; Mendes, R. Neighborhood topologies in fully Informed and best-of-neighborhood particle swarms. **IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews**, v. 36(4), pp. 515-519, 2006.

Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. **Science**, v. 220(4598), pp. 671-680, 1983.

Krohling, R.A. Private communication, **PPGI/UFES**, 2012-2017.

Krohling, R.A.; Coelho, L.S. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. **IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics**, v. 36(6), pp. 1407-1416, 2006.

Krohling, R.A.; Mendel, E. Bare bones particle swarm optimization with Gaussian or Cauchy jumps. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2009. pp. 3285-3291.

Krohling, R.A.; Mendel, E.; Campos, M. Swarm algorithms with chaotic jumps for optimization of multimodal functions. **Engineering Optimization**, v. 43, pp. 1243-1261, 2011.

Lai, Y-J.; Liu, T-Y; Hwang, C-L. TOPSIS for MODM. **European Journal of Operational Research**, v. 76, pp. 486-500, 1994.

Lee, C.-Y.; Yao, X. Evolutionary programming using mutations based on the Lévy probability distribution. **IEEE Transactions on Evolutionary Computation**, v. 8(1), pp. 1-13, 2004.

Lemonge, A.; Barbosa, H. An adaptive penalty scheme for genetic algorithms in structural optimization. **International Journal for Numerical Methods in Engineering**, v. 54, pp. 703-736, 2004.

Li, C.; Yang, S. A general framework of multipopulation methods with clustering in undetectable dynamics environments. **IEEE Transactions on Evolutionary Computation**, v. 16(4), pp. 556-577, 2012.

Li, C.; Yang, S. A comparative study on particle swarm optimization in dynamic environments. In S. Yang and X. Yao (Eds.): **Evolutionary Computation for DOPs**, SCI 490. Heidelberg: Springer-Verlag, 2013. pp. 109-136.

Liang, J.; Qin, A.; Suganthan, P.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. **IEEE Transactions on Evolutionary Computation**, v. 10(3), pp. 281-295, 2006.

Liu, C.-A. New dynamic constrained optimization PSO algorithm. In **Proceedings of the Fourth International Conference on Natural Computation**, 2008. pp. 650-653.

Liu, H.; Cai, Z.; Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. **Applied Soft Computing**, v. 10, pp. 629-640, 2010.

Liu, H.; Ding, G.; Wang, B. Bare-bones particle swarm optimization with disruption operator. **Applied Mathematics and Computation**, v. 238, pp. 106-122, 2014.

Liu, S.-H.; Mernik, M.; Bryant, B. Entropy-driven parameter control for evolutionary algorithms. **Informatica**, v. 31, pp. 41-50, 2007.

Liu, S.-H.; Mernik, M.; Bryant, B. To explore or to exploit: an entropy-driven approach for evolutionary algorithms. **International Journal of Knowledge-based and Intelligent Engineering Systems**, v. 13, pp. 185-206, 2009.

Liu, L.; Yang, S.; Wang, D. Particle swarm optimization with composite particles in dynamic environments. **IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics**, v. 40(6), pp. 1634-1648, 2010.

Luchi, F. Um algoritmo híbrido entre evolução diferencial e Nelder-Mead usando entropia para problemas de otimização não-linear inteira mista. **Dissertação de Mestrado (em Português)**, Universidade Federal do Espírito Santo, Vitória ES, 2016.

Luchi, F.; Krohling, R.A. Differential Evolution and Nelder-Mead for constrained non-linear integer optimization problems. **Procedia Computer Science**, v. 55, pp. 668-677, 2015.

Luenberger, D.; Ye, Y. **Linear and Nonlinear Programming, 3rd ed**. New York: Springer, 2008.

Lung, R.; Dumitrescu, D. A collaborative model for tracking optima in dynamic environments. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2007, pp. 564-567.

Lung, R.; Dumitrescu, D. Evolutionary swarm cooperative optimization in dynamic environments. **Natural Computing**, v. 9, pp. 83-94, 2010.

Madan, D.; Seneta, E. The variance gamma (V.G.) model for share market returns. **The Journal of Business**, v. 63(4), pp. 511-524, 1990.

Mantegna, R. Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. **Physical Review E**, v. 49(5), pp. 4677-4683, 1994.

Mavrovouniotis, M.; Li, C.; Yang, S. A survey of swarm intelligence for dynamic optimization: algorithms and applications. **Swarm and Evolutionary Computation**, v. 33, pp. 1-17, 2017.

Mendel, E.; Krohling, R.A.; Campos, M. Swarm algorithms with chaotic jumps applied to noisy optimization problems. **Information Sciences**, v. 181, pp. 4494-4514, 2011.

Mendes, R.; Kennedy, J.; Neves, J. The fully informed particle swarm: simpler, maybe better. **IEEE Transactions on Evolutionary Computation**, v. 8(3), pp. 204-210, 2004.

Mezura-Montes, E.; Coello, C. Constraint-handling in nature-inspired numerical optimization: past, present, and future. **Swarm and Evolutionary Computation**, v. 1, pp. 173-194, 2011.

Morales, A.; Quezada, C. A univesal eclectic genetic algorithm for constrained optimization. In **Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing**, 1998, pp. 518-522.

Morrison, R. **Designing evolutionary algorithms for dynamic environments**. Heidelberg: Springer, 2004.

Nemenman, I.; Shafee, F.; Bialek, W. Entropy and inference, revisited. In T. Dietterich, S. Becker, and Z. Gharamani (Eds.): **Advances in Neural Information Processing Systems** 14. Cambridge (Massachusetts): MIT Press, 2002. pp. 471-478.

Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. Equations of state calculations by fast computing machines. **Journal of Chemical Physics**, v. 21, pp. 1087-1092, 1953.

Nguyen, Q.; Nguyen, X.; Mckay, R.; Tuan, P. Initializing PSO with randomised low-discrepancy sequences: the comparative results. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2007. pp. 1985-1992.

Nguyen, T.; Yao, X. Benchmarking and solving dynamic constrained optimization. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2009. pp. 690-697.

Nguyen, T. Continuous dynamic optimisation using evolutionary algorithms. **Ph.D. thesis**, The University of Birmingham, Birmingham UK, 2010.

Nguyen, T.; Yao, X. Continuous dynamic constrained optimization the challenges. **IEEE Transactions on Evolutionary Computation**, v. 16(6), pp. 769-786, 2012.

Nguyen, T.; Yang, S.; Branke, J. Evolutionary dynamic optimization: a survey of the state of the art. **Swarm and Evolutionary Computation**, v. 6, pp. 1-24, 2012.

Olorunda, O.; Engelbrecht, P. Measuring exploration/exploitation in particle swarms using swarm diversity. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2008. pp. 1128-1134.

Orlitsky, A.; Santhanam, N.; Zhang, J. Always Good Turing: asymptotically optimal probability estimation. **Science**, v. 302, pp. 427-431, 2003.

Pal, K.; Saha, C.; Das, S.; Coello Coello, C. Dynamic constrained optimization with offspring repair based gravitational search algorithm. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2013. pp. 2414-2421.

Pant, M.; Thangaraj, R.; Grosan, C.; Abraham, A. Improved particle swarm optimization with low-discrepancy sequences. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2008. pp. 3011-3018.

Parsopoulos, K.; Vrahatis, M. Parameter selection and adaptation in unified particle swarm optimization. **Mathematical and Computer Modelling**, v. 46, pp. 198-213, 2007.

Parsopoulos, K.; Vrahatis, M. **Particle swarm optimization and intelligence: advances and applications**. New York: Information Science Reference, 2010.

Petalas, Y.; Parsopoulos, K.; Vrahatis, M. Entropy-based memetic particle swarm optimization for computing periodic orbits of nonlinear mappings. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2007. pp. 2040-2047.

Poli, R. Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. **IEEE Transactions on Evolutionary Computation**, v. 13(4), pp. 712-721, 2009.

Renyi, A. On measures of entropy and information. In **Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics, and Probability**, 1961. pp. 547-561.

Richer, T.; Blackwell, T. The Lévy particle swarm. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2006. pp. 3150-3157.

Richter, H. Detecting change in dynamic fitness landscapes. In **Proceedings of IEEE Congress on Evolutionary Computation**, 2009. pp. 1613-1620.

Richter, H.; Dietel, F. Change detection in dynamic fitness landscapes with time-dependent constraints. In **Proceedings of IEEE Second World Congress on Nature and Biologically Inspired Computing**, 2010. pp. 580-585.

Richter, H.; Yang, S. Dynamic optimization using analytic and evolutionary approaches: a comparative review. In I. Zelinka et al. (Eds.): **Handbook of Optimization**, Intelligence Systems Reference Library 38. Heidelberg: Springer-Verlag, 2012. pp. 1-28.

Ricotta, C.; Szeidl, L. Towards a unifying approach to diversity measures: bridging the gap between the Shannon entropy and Rao's quadratic index. **Theoretical Population Biology**, v. 70, pp. 237-243, 2006.

Runarsson, T.; Yao, X. Stochastic ranking for constrained evolutionary optimization. **IEEE Transactions on Evolutionary Computation**, v. 4(3), pp. 284-294, 2000.

Schneider, E.; Krohling, R.A. A hybrid approach using TOPSIS, differential evolution, and tabu search to find multiple solutions of constrained non-linear integer optimization problems. **Knowledge-Based Systems**, v. 62, pp. 47-56, 2014.

Shannon, C. A mathematical theory of communication. **The Bell System Technical Journal**, v. 27(3), pp. 379-423, 1948.

Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In **Proceedings of IEEE Congress on Evolutionary Computation**, 1998. pp. 69-73.

Simon, D. **Evolutionary optimization algorithms**. New Jersey: Wiley, 2013.

Suganthan, P.; Hansen, N.; Liang, J.; Deb, K.; Chen, Y.-P.; Auger, A.; Tiwari, S. **Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization**. Nanyang Technological University (Singapore) and KanGAL IIT (Kanpur, India), Technical Report, pp. 1-49, 2005.

Sundaram, R.K. **A first course in optimization theory**. New York: Cambridge University Press, 1996.

Takahama, T; Sakai, S. Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations. **IEEE Transactions on Evolutionary Computation**, v. 9(5), pp. 437-451, 2005.

Trelea, I. The particle swarm optimization algorithm: convergence analysis and parameter selection. **Information Processing Letters**, v. 85, pp. 317-325, 2003.

Tsallis, C. Possible generalization of Boltzmann-Gibbs statistics. **Journal of Statistical Physics**, v. 52, pp. 479-487, 1988.

Wang, H.; Rahnamayan, S; Sun, H.; Omran, M. Gaussian bare-bones differential evolution. **IEEE Transactions on Cybernetics**, v. 43(2), pp. 634-647, 2013.

Wolpert, D; Macready, W. No Free Lunch Theorems for Optimization. **IEEE Transactions on Evolutionary Computation**, v. 1(1), pp. 67-82, 1997.

White, T.; Pagurek, B. Towards multi-swarm problem solving in networks. In **Proceedings of the third International Conference on Multi Agent Systems**, 1998. pp. 333-340.

Yao, X; Liu, Y.; Lin, G. Evolutionary programming made faster. **IEEE Transactions on Evolutionary Computation**, v. 3(2), pp. 82-102, 1999.

Yazdani, D.; Nasiri, B.; Sepas-Moghaddam, A.; Meybodi, M. A novel multi-swarm algorithm for optimization in dynamic optimization based on particle swarm optimization. **Applied Soft Computing**, v. 13, pp. 2144-2158, 2013.

Zhang, Y.; Gong, D.-W.; Sun, X.-Y.; Geng, N. Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis. **Soft Computing**, v. 18(7), pp. 1337-1352, 2014.

Zhang, Y.; Gong, D.-W.; Geng, N.; Sun, X.-Y. Hybrid bare bones PSO for dynamic economic dispatch with valve-point effects. **Applied Soft Computing**, v. 18, pp. 248-260, 2014.

Zhou, Y.; Chen, Z; Zhang, J. Ranking vectors by means of the dominance degree matrix. **IEEE Transactions on Evolutionary Computation**, v. 21(1), pp. 34-51, 2017.

Part IV

Appendices

Appendix A

Optimization

This appendix presents a short review of optimization theory. Basic concepts of the theory are presented and conditions are discussed to guarantee the existence of optimal solutions, to ensure the uniqueness of an optimal solution, and to verify the optimality of a solution for unconstrained and constrained optimization problems. Some computational methods are also presented to solve optimization problems. For a detailed development of this theory, the reader is referred to Luenberger & Ye (2008); Sundaram (1996), and Bazaraa, Sherall & Shetty (1993).

A.1 Mathematical background

The set of positive integers is denoted by $\mathbb{Z}_+ = \{1,2,3,\dots\}$ and the set of integers by $\mathbb{Z} = \{\dots,-2,-1,0,1,2,\dots\}$. The set of real numbers is denoted by $\mathbb{R}$. Given a real number x, its absolute value will be denoted by $|x|$, where $|x| = x$ if $x \geq 0$ and $|x| = -x$ if $x < 0$. The Euclidean distance between two points (elements) x and y in $\mathbb{R}$ is defined as $d(x,y) = |x-y|$.

For any positive integer n, $\mathbb{R}^n$ will denote the n-dimensional Euclidean space. An element in $\mathbb{R}^n$ is a column vector $\mathbf{x} = (x_1,\dots,x_n)'$, where x_i is a real number for each $i = 1,\dots,n$. The number x_i is called the ith coordinate of $\mathbf{x}$. The vector $\mathbf{0}$ denotes the null vector $(0,\dots,0)'$ in $\mathbb{R}^n$. Given two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$, the scalar product of $\mathbf{x}$ and $\mathbf{y}$ is defined as $\mathbf{x}'\mathbf{y} = x_1y_1 + \cdots + x_ny_n$, where $\mathbf{x}' = (x_1,\dots,x_n)$ represents the row vector related to the column vector $\mathbf{x} = (x_1,\dots,x_n)'$. The Euclidean norm of a vector $\mathbf{x}$ in $\mathbb{R}^n$ is defined as $\|\mathbf{x}\|_2 = (\mathbf{x}'\mathbf{x})^{1/2}$. The Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ is given by $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2$. An open ball of center $\mathbf{x} \in \mathbb{R}^n$ and radius $r > 0$ is defined by $N_{\mathbf{x},r} = \{\mathbf{y} \in \mathbb{R}^n : d(\mathbf{x},\mathbf{y}) < r\}$. If $<$ is replaced by $\leq$, then a closed ball is defined. An open ball of center $\mathbf{x}$ and radius r is also called a neighborhood of $\mathbf{x}$. The concept of neighborhood of a point in $\mathbb{R}^n$ will be used to discuss the concepts of global and local minima.

A matrix is a rectangular array of numbers (symbols or expressions) arranged in rows and columns. A matrix having m rows and n columns is denoted by a boldface letter, for example, $\mathbf{A} = [a_{ij}]_{m\times n}$. Such a matrix is referred to as an $m \times n$ matrix. The transpose of a $m \times n$ matrix $\mathbf{A}$ is the $n \times m$ matrix $\mathbf{A}'$ with elements $a'_{ij} = a_{ji}$. A matrix which has the same number of rows and columns is called a square matrix. A square matrix $\mathbf{A}$ is symmetric if $\mathbf{A}' = \mathbf{A}$. A symmetric matrix $\mathbf{A}$ is said to be positive definite if $\mathbf{x}'\mathbf{A}\mathbf{x} > 0$ for all nonzero vectors $\mathbf{x}$. Similarly, a symmetric matrix $\mathbf{A}$ is said to be positive semidefinite if $\mathbf{x}'\mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x}$. These concepts will be used to discuss optimality conditions for candidate solutions of optimization problems.

A.1.1 Topological concepts in $\mathbb{R}^n$

A point $\mathbf{x} \in \mathbb{R}^n$ is called an interior point of $X \subset \mathbb{R}^n$ if $\mathbf{x}$ has a neighborhood consisting entirely of points of X. The set of all interior points of X is denoted by $\text{Int}(X)$ and is called the interior of X. A set $O \subset \mathbb{R}^n$ is said to be open if its points are all interior points. An open ball is an open set and the interior of a set $X \subset \mathbb{R}^n$ is obviously an open set. A set $C \subset \mathbb{R}^n$ is said to be closed if and only if its complement $C^c = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \notin C\}$ is open. An equivalent definition says that a set C is closed if and only if any convergent sequence formed by points of C converges to a limit-point in C. A set $B \subset \mathbb{R}^n$ is said to be bounded if there is a $r > 0$ such that $B \subset N_{\mathbf{0},r}$. A set $Y \subset \mathbb{R}^n$ is said to be compact if and only if it is closed and bounded. Topological concepts along with that of continuous function will be used to discuss the set of conditions under which the existence of solutions to an optimization problem is guaranteed.

A.1.2 Functions

A real-valued function f defined on $X \subset \mathbb{R}^n$ is said to be continuous at $\mathbf{x}_0 \in X$ if the following holds: for any $\varepsilon > 0$ there is a $\delta > 0$ such that for all $\mathbf{x} \in N_{\mathbf{x}_0,\delta}$ implies $f(\mathbf{x}) \in N_{f(\mathbf{x}_0),\varepsilon}$. Intuitively, f is continuous at $\mathbf{x}_0$ if the value of f at any point $\mathbf{x}$ that is close to $\mathbf{x}_0$ is a good approximation of the value of f at $\mathbf{x}_0$. In addition, f is continuous on X if it is continuous at every point of X.

The class C^0 represents the space of continuous functions. A function is said to be of class C^1 if its first derivative exist and is continuous, i.e., C^1 is the space of (continuously) differentiable functions. A function is said to be of class C^2 if its first and second derivative both exist and are continuous, i.e., C^2 is the space of (continuously) twice-differentiable functions. More generally, a function is said to be of class C^k if the first k derivatives of the function all exist and are continuous. If derivatives of a function exist for all positive integers, this function is said to be smooth, or equivalently, of class C^∞.

Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a real-valued function and suppose that $f \in C^1$. The gradient of f at $\bar{\mathbf{x}} \in \text{Int}(X)$ is defined as the vector

$$\nabla_{\mathbf{x}} f(\bar{\mathbf{x}}) = \left(\frac{\partial f(\bar{\mathbf{x}})}{\partial x_1}, \ldots, \frac{\partial f(\bar{\mathbf{x}})}{\partial x_n} \right)'_{n \times 1} = G_f(\bar{\mathbf{x}}). \tag{A.1}$$

Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a real-valued function and suppose that $f \in C^2$. The Hessian of f at $\bar{\mathbf{x}} \in \text{Int}(X)$ is defined as the matrix

$$\nabla^2_{\mathbf{x}} f(\bar{\mathbf{x}}) = \begin{pmatrix} \frac{\partial^2 f(\bar{\mathbf{x}})}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\bar{\mathbf{x}})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\bar{\mathbf{x}})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(\bar{\mathbf{x}})}{\partial x_n^2} \end{pmatrix}_{n \times n} = H_f(\bar{\mathbf{x}}). \tag{A.2}$$

A set of real-valued functions $g_1, \ldots, g_m$, all defined on $X \subset \mathbb{R}^n$, can be regarded as a vector-valued function $\mathbf{g} = (g_1, \ldots, g_m)$ defined on X. This function assigns a vector $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})) \in \mathbb{R}^m$ to every vector $\mathbf{x} \in X$. A vector-valued function is said to be continuous at $\mathbf{x}_0 \in X$ if each of its component functions is continuous at $\mathbf{x}_0$. In addition, a vector-valued function is said to be of class C^k if each of its component functions is of class C^k.

Let $\mathbf{g} = (g_1, \ldots, g_m) : X \subset \mathbb{R}^n \to \mathbb{R}^m$ be a vector-valued function and suppose that $\mathbf{g} \in C^1$. The Jacobian of $\mathbf{g}$ at $\bar{\mathbf{x}} \in \text{Int}(X)$ is defined as the matrix

$$\nabla_{\mathbf{x}} \mathbf{g}(\bar{\mathbf{x}}) = \begin{pmatrix} \nabla_{\mathbf{x}} g_1(\bar{\mathbf{x}})' \\ \vdots \\ \nabla_{\mathbf{x}} g_m(\bar{\mathbf{x}})' \end{pmatrix} = \begin{pmatrix} \frac{\partial g_1(\bar{\mathbf{x}})}{\partial x_1} & \cdots & \frac{\partial g_1(\bar{\mathbf{x}})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m(\bar{\mathbf{x}})}{\partial x_1} & \cdots & \frac{\partial g_m(\bar{\mathbf{x}})}{\partial x_n} \end{pmatrix}_{m \times n} = J_{\mathbf{g}}(\bar{\mathbf{x}}). \tag{A.3}$$

Gradient, Hessian, and Jacobian will be used to provide answers to questions about optimality conditions for candidate solutions of optimization problems.

A.1.3 Convexity

Let W be a subset of $\mathbb{R}^n$ and suppose that $\mathbf{x}_1$ and $\mathbf{x}_2$ are points in W. Any point $(1-\lambda)\mathbf{x}_1 + \lambda \mathbf{x}_2$ with $\lambda \in [0,1]$ is referred to as a convex combination of $\mathbf{x}_1$ and $\mathbf{x}_2$. If $\lambda \in (0,1)$, then it is a strict convex combination. A set $W \subset \mathbb{R}^n$ is said to be convex if the convex combination of any two points in W is also in W.

Suppose that $\emptyset \neq W \subset \mathbb{R}^n$ is a convex set and let f be a real-valued function defined on W. The function f is said to be convex on W if $f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2)$ for every $\mathbf{x}_1, \mathbf{x}_2 \in W$ and every $\lambda \in [0,1]$. The function f is called strictly convex on W if the inequality is true as a strict inequality for every $\mathbf{x}_1 \neq \mathbf{x}_2$ and $\lambda \in (0,1)$.

A.2 Global and local minima

Let f be a real-valued function defined on $\mathbb{R}^D$ (for some positive integer D) and consider a set $\mathbb{S} \subset \mathbb{R}^D$. A point $\bar{\mathbf{x}} \in \mathbb{S}$ is a global minimum of f on $\mathbb{S}$ if $f(\bar{\mathbf{x}}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{S}$. If $\bar{\mathbf{x}}$ is a global minimum of f on $\mathbb{S}$, it is also called a minimizer of f on $\mathbb{S}$ and $f(\bar{\mathbf{x}})$ is called the global minimum value of f on $\mathbb{S}$. A point $\bar{\mathbf{x}} \in \mathbb{S}$ is a local minimum of f on $\mathbb{S}$ if there is a $r > 0$ such that $f(\bar{\mathbf{x}}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in N_{\bar{\mathbf{x}},r} \cap \mathbb{S}$. A point $\bar{\mathbf{x}} \in \mathbb{S}$ is a strict local minimum of f on $\mathbb{S}$ if there is a $r > 0$ such that $f(\bar{\mathbf{x}}) < f(\mathbf{x})$ for all $\mathbf{x} \in N_{\bar{\mathbf{x}},r} \cap \mathbb{S}$ and $\mathbf{x} \neq \bar{\mathbf{x}}$.

Let g be a real-valued function defined on $\mathbb{R}^D$ and consider a set $\mathbb{S} \subset \mathbb{R}^D$. A point $\bar{\mathbf{x}} \in \mathbb{S}$ is a global maximum of g on $\mathbb{S}$ if $g(\bar{\mathbf{x}}) \geq g(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{S}$. If $\bar{\mathbf{x}}$ is a global maximum of g on $\mathbb{S}$, it is also called a maximizer of g on $\mathbb{S}$ and $g(\bar{\mathbf{x}})$ is called the global maximum value of g on $\mathbb{S}$. A point $\bar{\mathbf{x}} \in \mathbb{S}$ is a local maximum of g on $\mathbb{S}$ if there is a $r > 0$ such that $g(\bar{\mathbf{x}}) \geq g(\mathbf{x})$ for all $\mathbf{x} \in N_{\bar{\mathbf{x}},r} \cap \mathbb{S}$. A point $\bar{\mathbf{x}} \in \mathbb{S}$ is a strict local maximum of g on $\mathbb{S}$ if there is a $r > 0$ such that $g(\bar{\mathbf{x}}) > g(\mathbf{x})$ for all $\mathbf{x} \in N_{\bar{\mathbf{x}},r} \cap \mathbb{S}$ and $\mathbf{x} \neq \bar{\mathbf{x}}$.

The term local optimum is used to refer either a local minimum or a local maximum. The concept of global optimum is defined analogously. Every global optimum is also a local optimum. In general, the opposite is not true. Note that local optimality applies only around a neighborhood of the local optimum point and global optimality applies over the entire set $\mathbb{S}$.

A.3 Optimization problems

Let f be a real-valued function defined on $\mathbb{S} \subset \mathbb{R}^D$. A minimization problem can be stated as:

$$\begin{array}{rl} \min & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathbb{S}. \end{array} \tag{A.4}$$

The vector $\mathbf{x} = (x_1, \ldots, x_D)'$ represents a container for the decision variables of the problem, the function f is called the objective function, and the set $\mathbb{S}$ the search space (or feasible set). Some authors assume that $L_d \leq x_d \leq U_d$, for all $d = 1, \ldots, D$, where U_d and L_d are the upper and lower bounds of x_d, respectively. In this case, these bounds define the search space as

$$\mathbb{S} = [L_1, U_1] \times \cdots \times [L_D, U_D] = [\mathbf{L}, \mathbf{U}] \subset \mathbb{R}^D. \tag{A.5}$$

A point $\mathbf{x}$ in $\mathbb{S}$ is called a feasible solution to the problem. The objective function f is to be minimized and $f(\mathbf{x})$ is the objective function value of a feasible solution $\mathbf{x}$. Sometimes, f is also called of cost function and, consequently, $f(\mathbf{x})$ is called cost of $\mathbf{x}$. Regardless of how f is called, $f(\mathbf{x})$ represents a measure of the quality of a feasible solution $\mathbf{x}$. Two types of solutions can occur when a minimization problem is considered: local minimum points or global minimum points. The main goal is to find a global minimum of f on $\mathbb{S}$, i.e., a point $\bar{\mathbf{x}}$ in $\mathbb{S}$ that achieves the global minimum value of f on $\mathbb{S}$ (the smallest value of f on $\mathbb{S}$). The set of all feasible solutions that achieve the global minimum value of f on $\mathbb{S}$ is denoted by $\arg\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{S}\}$.

Let g be a real-valued function defined on $\mathbb{S} \subset \mathbb{R}^D$. A maximization problem can be stated as:

$$\begin{array}{rl} \max & g(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathbb{S}. \end{array} \tag{A.6}$$

This problem has the same elements of the problem (A.4), except that the objective function g is to be maximized. Two types of solutions can occur when a maximization problem is considered: local maximum points or global maximum points. The main goal is to find a global maximum of g on $\mathbb{S}$, i.e., a point $\bar{\mathbf{x}}$ in $\mathbb{S}$ that achieves the global maximum value of g on $\mathbb{S}$. The set of all feasible solutions that achieve the global maximum value of g on $\mathbb{S}$ is denoted by $\arg\max\{g(\mathbf{x}) : \mathbf{x} \in \mathbb{S}\}$

Result A.1. *Consider the problem (A.6). A point $\bar{\mathbf{x}}$ is a maximum of g on $\mathbb{S}$ if and only if $\bar{\mathbf{x}}$ is a minimum of f on $\mathbb{S}$, where f is a function whose value at any point $\mathbf{x} \in \mathbb{S}$ is given by $f(\mathbf{x}) = -g(\mathbf{x})$.*

Result A.1 shows that a maximization problem can be converted into a minimization problem by multiplying the objective function by -1. Therefore, without loss of generality, a minimization problem can be considered as an optimization problem in its general form. It is important to note that a minimization problem can have many local minimizers, each one with its corresponding local minimum value. However, if there is a global minimum value, then that value is unique. But note that the global minimum value may be probably reached by many global minimizers. In practice, it may be difficult to find a global minimizer, then a good alternative result could be when a local minimizer is found.

A.4 Existence of solutions

This section discusses the existence of solutions for a minimization problem. The main goal is to establish the set of conditions on f and $\mathbb{S}$ under which the existence of solutions is guaranteed. First, note that $\inf f(\mathbf{x}) = \inf\{f(\mathbf{x}) :$

$\mathbf{x} \in \mathbb{S}\} = +\infty$ if $\mathbb{S} = \emptyset$ (in this case, the non-existence of solutions is evident). Furthermore, $\inf f(\mathbf{x}) = -\infty$ if f is unbounded below on $\mathbb{S} \neq \emptyset$. However, even if $\inf f(\mathbf{x})$ is finite on $\mathbb{S}$, a global minimizer can not exist (see: minimize e^x subject to $x > 0$). But, if $\inf f(\mathbf{x})$ is finite and there is a global minimum $\bar{\mathbf{x}}$ to the problem (A.4), then $\bar{f} = f(\bar{\mathbf{x}}) = \min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{S}\}$ is the global minimum value of f on $\mathbb{S}$ and $\bar{f} = \inf f(\mathbf{x})$.

Result A.2. *Consider the problem (A.4). If $\mathbb{S}$ is a compact set and f is a continuous function on $\mathbb{S}$, then f attains a minimum and a maximum on $\mathbb{S}$, i.e., there exist points $\bar{\mathbf{x}}_{\min}$ and $\bar{\mathbf{x}}_{\max}$ in $\mathbb{S}$ such that $f(\bar{\mathbf{x}}_{\min}) \leq f(\mathbf{x}) \leq f(\bar{\mathbf{x}}_{\max})$ for all $\mathbf{x}$ in $\mathbb{S}$.*

Result A.2 describes a general set of conditions that represents an answer to the question of the existence of solutions for an optimization problem. It can be used as a criterion to ensure the existence or not of an optimal solution for a given minimization problem. This result implies that the minimization problem of a continuous function over a compact set is guaranteed to have an optimal solution.

A.5 Minimization of a convex function

This section discusses the minimization problem (A.4) when f is a convex function on $\mathbb{S}$. A first-order optimality condition for this problem is presented. In addition, conditions to deal with the uniqueness of an optimal solution (i.e. a single local optimum that is also the global optimum) are also presented.

Result A.3. *Consider the problem (A.4). Assume that $\mathbb{S}$ is a convex set and f is a convex function on $\mathbb{S}$.*

1. *If $\bar{\mathbf{x}}$ is a local minimum of f on $\mathbb{S}$, then $\bar{\mathbf{x}}$ is global minimum.*
2. *The set of minimizers of f on $\mathbb{S}$ is either empty or convex.*
3. *(Uniqueness) If f is strictly convex on $\mathbb{S}$, then the set of minimizers of f on $\mathbb{S}$ is either empty or contains a single point.*
4. *(First-order condition) Finally, assume also that $f \in C^1$ and let $\bar{\mathbf{x}}$ be an interior point of $\mathbb{S}$. Then, $\nabla_{\mathbf{x}} f(\bar{\mathbf{x}}) = \mathbf{0}$ if and only if $\bar{\mathbf{x}}$ is a global minimum of f on $\mathbb{S}$.*

A.6 Optimality conditions

A.6.1 Unconstrained optimization

The minimization problem (A.4) is also called an unconstrained optimization problem (UOP). This section discusses optimality conditions for an UOP.

Result A.4. *Consider the problem (A.4):*

1. *(First-order necessary condition). Suppose that $f \in C^1$ and let $\bar{\mathbf{x}}$ be an interior point of $\mathbb{S}$. If $\bar{\mathbf{x}}$ is a local minimum of f on $\mathbb{S}$, then $\nabla_{\mathbf{x}} f(\bar{\mathbf{x}}) = \mathbf{0}$.*
2. *(Second-order necessary conditions). Suppose that $f \in C^2$ and let $\bar{\mathbf{x}}$ be an interior point of $\mathbb{S}$. If $\bar{\mathbf{x}}$ is a local minimum of f on $\mathbb{S}$, then $\nabla_{\mathbf{x}} f(\bar{\mathbf{x}}) = \mathbf{0}$ and $\nabla^2_{\mathbf{x}} f(\bar{\mathbf{x}})$ is positive semidefinite.*
3. *(Second-order sufficient conditions). Suppose that $f \in C^2$ and let $\bar{\mathbf{x}}$ be an interior point of $\mathbb{S}$. If $\nabla_{\mathbf{x}} f(\bar{\mathbf{x}}) = \mathbf{0}$ and $\nabla^2_{\mathbf{x}} f(\bar{\mathbf{x}})$ is positive definite, then $\bar{\mathbf{x}}$ is a strict local minimum of f on $\mathbb{S}$.*

A.6.2 Constrained optimization

Without loss of generality, a constrained optimization problem (COP) can be stated as:

$$\begin{array}{rll} \min & f(\mathbf{x}) & \text{where } \mathbf{x} = (x_1, \ldots, x_D)' \in \mathbb{R}^D \\ \text{subject to} & g_i(\mathbf{x}) \leq 0 & i = 1, \ldots, I \\ & h_j(\mathbf{x}) = 0 & j = 1, \ldots, J \\ & L_d \leq x_d \leq U_d & d = 1, \ldots, D. \end{array} \tag{A.7}$$

Once again, the vector $\mathbf{x} = (x_1, \ldots, x_D)'$ represents a container for the decision variables of the problem and $f : \mathbb{S} \subset \mathbb{R}^D \to \mathbb{R}$ is the objective function to be minimized. The functions $g_1, \ldots, g_I, h_1, \ldots, h_J : \mathbb{S} \subset \mathbb{R}^D \to \mathbb{R}$ are called of constraint functions. All these functions $(f, g_1, \ldots, g_I, h_1, \ldots, h_J)$ can be linear or nonlinear. As is usually done the search space is defined as $\mathbb{S} = [L_1, U_1] \times \cdots \times [L_D, U_D] = [\mathbf{L}, \mathbf{U}]$ and the set

$$\mathbb{F} = \{\mathbf{x} \in \mathbb{S} : g_1(\mathbf{x}) \leq 0, \ldots, g_I(\mathbf{x}) \leq 0, h_1(\mathbf{x}) = 0, \ldots, h_J(\mathbf{x}) = 0\} \tag{A.8}$$

is called the feasible space. A point $\mathbf{x} \in \mathbb{S}$ that satisfies all the constraints, i.e., a point $\mathbf{x} \in \mathbb{F} \subset \mathbb{S}$, is called a feasible solution to the problem. The problem (A.7) is solved when a global minimum of f on $\mathbb{F}$ is found (when such solution exist). Remember that if $\inf f(\mathbf{x})$ is finite and there is a global minimum $\bar{\mathbf{x}}$ to the problem (A.7), then $\bar{f} = f(\bar{\mathbf{x}}) = \min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{S}\}$ is the global minimum value of f on $\mathbb{S}$ and $\bar{f} = \inf f(\mathbf{x})$.

An alternative notation for the problem (A.7) is given by:

$$\begin{array}{rl} \min & f(\mathbf{x}) \\ \text{subject to} & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{x} \in \mathbb{S} \subset \mathbb{R}^D \end{array} \tag{A.9}$$

where $\mathbf{x} \in \mathbb{R}^D$, $f : \mathbb{S} \subset \mathbb{R}^D \to \mathbb{R}$, $\mathbf{g} : \mathbb{S} \subset \mathbb{R}^D \to \mathbb{R}^I$, and $\mathbf{h} : \mathbb{S} \subset \mathbb{R}^D \to \mathbb{R}^J$. By considering the notation in (A.9), the Lagrangian function $L : \mathbb{R}^D \times \mathbb{R}^I \times \mathbb{R}^J \to \mathbb{R}$ associated with the problem (A.7) (or A.9) is given by:

$$L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\mu}'\mathbf{g}(\mathbf{x}) + \boldsymbol{\lambda}'\mathbf{h}(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{I} \mu_i g_i(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j h_j(\mathbf{x}). \tag{A.10}$$

Observe that

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \nabla_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{\mu}'\nabla_{\mathbf{x}}\mathbf{g}(\mathbf{x}) + \boldsymbol{\lambda}'\nabla_{\mathbf{x}}\mathbf{h}(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x}) + \sum_{i=1}^{I} \mu_i \nabla_{\mathbf{x}} g_i(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j \nabla_{\mathbf{x}} h_j(\mathbf{x}) \tag{A.11}$$

$$\nabla_{\boldsymbol{\mu}} L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \mathbf{g}(\mathbf{x}) \qquad \nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \mathbf{h}(\mathbf{x}) \tag{A.12}$$

and

$$\nabla^2_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \nabla^2_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{\mu}'\nabla^2_{\mathbf{x}}\mathbf{g}(\mathbf{x}) + \boldsymbol{\lambda}'\nabla^2_{\mathbf{x}}\mathbf{h}(\mathbf{x}) = \nabla^2_{\mathbf{x}} f(\mathbf{x}) + \sum_{i=1}^{I} \mu_i \nabla^2_{\mathbf{x}} g_i(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j \nabla^2_{\mathbf{x}} h_j(\mathbf{x}). \tag{A.13}$$

If there is a feasible point $\tilde{\mathbf{x}}$ such that $g_i(\tilde{\mathbf{x}}) = 0$ for some $i = 1, \ldots, I$, then g_i is said to be active at $\tilde{\mathbf{x}}$. Every equality constraint is active for all feasible points. A feasible point $\tilde{\mathbf{x}}$ is said to be a regular point of the constraints of the problem if $\nabla_{\mathbf{x}} g_k(\tilde{\mathbf{x}})$ for $k \in A_{\tilde{\mathbf{x}}} = \{k : g_k(\tilde{\mathbf{x}}) = 0\}$ and $\nabla_{\mathbf{x}} h_j(\tilde{\mathbf{x}})$ for $j = 1, \ldots, J$ are linearly independent. Optimality conditions for a COP are given by Results A.5, A.6, A.7, and A.8.

Result A.5 (First-order necessary conditions)**.** *Consider the problem (A.9). Suppose that $f, \mathbf{g}, \mathbf{h} \in C^1$ and let $\bar{\mathbf{x}}$ be a regular point of the constraints. If $\bar{\mathbf{x}}$ is a local minimum of f on $\mathbb{F}$, then there are vectors $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\lambda}}$ such that*

$$\nabla_{\mathbf{x}} L(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\lambda}}) = \mathbf{0} \tag{A.14}$$

$$\bar{\boldsymbol{\mu}}'\mathbf{g}(\bar{\mathbf{x}}) = 0 \tag{A.15}$$

$$\bar{\boldsymbol{\mu}} \geq \mathbf{0}. \tag{A.16}$$

Eqs. (A.14), (A.15), and (A.16) are named as Karush-Kuhn-Tucker necessary conditions (or KKT conditions).

For the next result, consider the following notation. Let $\bar{\mathbf{x}}$ be a local minimum of f on $\mathbb{F}$ and let $A_{\bar{\mathbf{x}}} = \{k : g_k(\bar{\mathbf{x}}) = 0\}$. In addition, denote $A^+_{\bar{\mathbf{x}}} = \{k \in A_{\bar{\mathbf{x}}} : \bar{\mu}_k > 0\}$ and $A^0_{\bar{\mathbf{x}}} = \{k \in A_{\bar{\mathbf{x}}} : \bar{\mu}_k = 0\}$.

Result A.6 (Second-order necessary conditions)**.** *Consider the problem (A.9). Suppose that $f, \mathbf{g}, \mathbf{h} \in C^2$ and let $\bar{\mathbf{x}}$ be a regular point of the constraints. If $\bar{\mathbf{x}}$ is a local minimum of f on $\mathbb{F}$, then there are vectors $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\lambda}}$ such that the KKT conditions are satisfied by $(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\lambda}})$ and $\mathbf{y}'\nabla^2_{\mathbf{x}} L(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\lambda}})\mathbf{y} \geq 0$ for all $\mathbf{y} \in Y$, where*

$$Y = \{\mathbf{y} : \nabla_{\mathbf{x}} h_{i=1,\ldots,I}(\bar{\mathbf{x}})'\mathbf{y} = 0, \nabla_{\mathbf{x}} g_{k \in A^+_{\bar{\mathbf{x}}}}(\bar{\mathbf{x}})'\mathbf{y} = 0, \nabla_{\mathbf{x}} g_{k \in A^0_{\bar{\mathbf{x}}}}(\bar{\mathbf{x}})'\mathbf{y} \leq 0\}. \tag{A.17}$$

Result A.7 (Second-order sufficient conditions)**.** *Consider the problem (A.9). Suppose that $f, \mathbf{g}, \mathbf{h} \in C^2$ and let $\bar{\mathbf{x}}$ be a regular point of the constraints. If there are vectors $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\lambda}}$ such that*

$$\nabla_{\mathbf{x}} L(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\lambda}}) = \mathbf{0} \tag{A.18}$$

$$\bar{\boldsymbol{\mu}}' \mathbf{g}(\bar{\mathbf{x}}) = 0 \tag{A.19}$$

$$\bar{\boldsymbol{\mu}} \geq \mathbf{0} \tag{A.20}$$

and

$$\mathbf{y}' \nabla_{\mathbf{x}}^2 L(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\lambda}}) \mathbf{y} > 0 \tag{A.21}$$

for all $\mathbf{y} \in Y - \{\mathbf{0}\}$, then $\bar{\mathbf{x}}$ is a strict local minimum of f on $\mathbb{F}$.

Result A.8 (First-order condition)**.** *Consider the problem (A.9). Assume that $\mathbb{S}$ is a convex set, f and $\mathbf{g}$ are convex functions on $\mathbb{S}$, and $\mathbf{h}$ is affine. Assume also that $f, \mathbf{g} \in C^1$ and let $\bar{\mathbf{x}}$ be a regular point of the constraints. Then, there are vectors $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\lambda}}$ such that the KKT conditions are satisfied by $(\bar{\mathbf{x}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\lambda}})$ if and only if $\bar{\mathbf{x}}$ is a global minimum of f on $\mathbb{F}$.*

COPs are classified into different classes of problems based on the properties of the objective function f and the properties of the set $\mathbb{F}$ of feasible solutions. Three of these classes are mentioned here as examples:

Linear programming. Minimize $\mathbf{q}'\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, where $\mathbf{q} \in \mathbb{R}^D$, $\mathbf{b} \in \mathbb{R}^J$, and $\mathbf{A}$ is a $J \times D$ matrix (of real numbers).

Quadratic programming with linear constraints. Minimize $\frac{1}{2}\mathbf{x}'\mathbf{Q}\mathbf{x} + \mathbf{q}'\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, where $\mathbf{Q}$ is a $D \times D$ matrix (of real numbers) positive semidefinite, $\mathbf{q} \in \mathbb{R}^D$, $\mathbf{b} \in \mathbb{R}^I$, and $\mathbf{A}$ is a $I \times D$ matrix (of real numbers).

Convex optimization. When $\mathbb{S}$ is a convex set, $f, g_1, \ldots, g_I$ are convex functions, and $h_1, \ldots, h_J$ are affine (i.e., $h_j(\mathbf{x}) = \mathbf{a}_j'\mathbf{x} + b_j$, for all j, where $\mathbf{a}_j \in \mathbb{R}^D$ and $b_j \in \mathbb{R}$). Note that linear programming and quadratic programming with linear constraints are convex optimization problems.

A.7 Single-point numerical methods for optimization

Consider an UOP such as defined in (A.4). An algorithm for solving this problem can be viewed as an iterative process that generates a sequence of points in $\mathbb{S}$ according to a prescribed set of instructions and a termination criterion. Given a vector $\mathbf{x}^\tau \in \mathbb{S}$, a new point $\mathbf{x}^{\tau+1} \in \mathbb{S}$ is obtained by applying the instructions of the algorithm, where the index τ ($\tau = 0, 1, 2, 3, \ldots$) represents the iteration counter of this process. Formally, this process can be described by an algorithmic map

$$\mathscr{A} : \mathbb{S} \to \mathbb{S} \qquad \mathbf{x}^\tau \mapsto \mathbf{x}^{\tau+1} = \mathscr{A}(\mathbf{x}^\tau). \tag{A.22}$$

Given a initial point $\mathbf{x}^0$, the map $\mathscr{A}$ generates a sequence $\mathbf{x}^1, \mathbf{x}^2, \ldots$ of points in $\mathbb{S}$. The transformation of $\mathbf{x}^\tau$ into $\mathbf{x}^{\tau+1}$ through the map constitutes an iteration of the algorithm. A desirable property of an algorithm for solving an UOP is that it generates a sequence of points converging to a global minimum. However, in many practical cases, it is satisfactory when a less favorable solution is obtained (for example, a local minimum). In fact, the iterative procedure of an algorithm stops if a point belonging to a prescribed set, called solution set, is reached. Given an UOP, the following are some typical solution sets (Luenberger & Ye, 2008; Bazaraa, Sherall & Shetty, 1993):

1. $\{\mathbf{x}^* \in \mathbb{S} : \|\nabla_{\mathbf{x}} f(\mathbf{x}^*)\| < \varepsilon\}$, where $\varepsilon > 0$ is specified.
2. $\{\mathbf{x}^* \in \mathbb{S} : \tau = \tau_{\max}$ and $\mathbf{x}^*$ is the best value obtained so far$\}$, where $\tau_{\max}$ is specified.
3. $\{\mathbf{x}^* \in \mathbb{S} : |f(\mathbf{x}^*) - \bar{f}| < \varepsilon\}$, where $\bar{f}$ is the known global minimum value and $\varepsilon > 0$ is specified.
4. $\{\mathbf{x}^* \in \mathbb{S} : |f(\mathbf{x}^*)| \leq b\}$, where b is an acceptable value.

A number of algorithms have been developed for solving UOPs. Many of these methods evolve a single point according to the following algorithmic map:

$$\mathbf{x}^{\tau+1} = \mathscr{A}(\mathbf{x}^\tau) = \mathbf{x}^\tau + \eta^\tau \mathbf{d}^\tau \tag{A.23}$$

for $\tau = 0, 1, 2, \ldots$, where $\mathbf{d}^\tau$ is the search direction at the point $\mathbf{x}^\tau$ and η^τ is a suitable step size (or a learning rate). Three single-point numerical methods for UOPs can be presented here: the gradient method, the Newton-Raphson method, and the BFGS quasi-Newton method (see Luenberger & Ye, 2008; Bazaraa, Sherall & Shetty, 1993). These algorithms make use of information based on gradients and Hessians to locate a global minimum or, more exactly, to obtain an approximation of a global minimum. The gradient method is a first-order derivative method because it uses the gradient vector of the objective function to obtain an approximation of a global minimum. The Newton-Raphson method is a second-order derivative method because it uses the gradient vector and the inverse Hessian matrix of the objective function to obtain an approximation of a global minimum. The BFGS method uses an approximation to the inverse Hessian in place of the true inverse.

A.7.1 Gradient method

Suppose that $f \in C^1$. The gradient method is defined by the map

$$\mathbf{x}^{\tau+1} = \mathscr{A}(\mathbf{x}^\tau) = \mathbf{x}^\tau - \eta^\tau \nabla_{\mathbf{x}} f(\mathbf{x}^\tau) \tag{A.24}$$

for $\tau = 0, 1, 2, \ldots$, where $\eta^\tau = \arg\min_\eta f(\mathbf{x}^\tau - \eta \nabla_{\mathbf{x}} f(\mathbf{x}^\tau))$ subject to $0 \leq \eta < \eta_{\max}$. The essential steps of this method are summarized in Algorithm 6.

Algorithm 6 Gradient method for UOPs.

Input: $f, \mathbf{x}_0, \eta_{\max}, \varepsilon > 0$, and $\tau_{\max}$
1: $\tau \leftarrow 0$; $\mathbf{x} \leftarrow \mathbf{x}_0$
2: **repeat**
3: $\quad \tau \leftarrow \tau + 1$
4: $\quad \bar{\eta} = \arg\min_\eta \phi(\eta)$ subject to $0 \leq \eta < \eta_{\max}$, where $\phi(\eta) = f(\mathbf{x} - \eta \nabla_{\mathbf{x}} f(\mathbf{x}))$
5: $\quad \mathbf{x}^* \leftarrow \mathbf{x} - \bar{\eta} \nabla_{\mathbf{x}} f(\mathbf{x})$ and $f^* \leftarrow f(\mathbf{x}^*)$
6: $\quad \mathbf{x} \leftarrow \mathbf{x}^*$
7: **until** $\|\nabla_{\mathbf{x}} f(\mathbf{x}^*)\| < \varepsilon$ or $\tau = \tau_{\max}$
8: **return** $\mathbf{x}^*$ and f^*.

A.7.2 The Newton-Raphson method

Suppose that $f \in C^2$. The Newton-Raphson method is defined by the map

$$\mathbf{x}^{\tau+1} = \mathscr{A}(\mathbf{x}^\tau) = \mathbf{x}^\tau - [\nabla^2_{\mathbf{x}} f(\mathbf{x}^\tau)]^{-1} \nabla_{\mathbf{x}} f(\mathbf{x}^\tau) \tag{A.25}$$

for $\tau = 0, 1, 2, \ldots$. The idea behind of the Newton-Raphson method is that the function being minimized is approximated locally by a quadratic function and this approximate function is minimized exactly. The essential steps of this method are summarized in Algorithm 7.

Algorithm 7 Newton-Raphson method for UOPs.

Input: $f, \mathbf{x}_0, \varepsilon > 0$, and $\tau_{\max}$
1: $\tau \leftarrow 0$; $\mathbf{x} \leftarrow \mathbf{x}_0$
2: **repeat**
3: $\quad \tau \leftarrow \tau + 1$
4: $\quad \mathbf{x}^* \leftarrow \mathbf{x} - [\nabla^2_{\mathbf{x}} f(\mathbf{x})]^{-1} \nabla_{\mathbf{x}} f(\mathbf{x})$ and $f^* \leftarrow f(\mathbf{x}^*)$
5: $\quad \mathbf{x} \leftarrow \mathbf{x}^*$
6: **until** $\|\nabla_{\mathbf{x}} f(\mathbf{x}^*)\| < \varepsilon$ or $\tau = \tau_{\max}$
7: **return** $\mathbf{x}^*$ and f^*.

A.7.3 The BFGS method

There are a number of so-called quasi-Newton methods that gradually build up the inverse Hessian in the successive iterations. The BFGS method (named by its developers Broyden, Fletcher, Goldfarb, and Shanno) is defined by

the map

$$\mathbf{x}^{\tau+1} = \mathscr{A}(\mathbf{x}^{\tau}) = \mathbf{x}^{\tau} - \eta^{\tau} H^{\tau} \nabla_{\mathbf{x}} f(\mathbf{x}^{\tau}) \tag{A.26}$$

for $\tau = 0, 1, 2, \ldots$, where $H^0 = \mathbf{I}$ (the identity matrix) and

$$H^{\tau+1} = (I - \rho_{\tau} \mathbf{s}_{\tau} \mathbf{y}_{\tau}') H^{\tau} (I - \rho_{\tau} \mathbf{y}_{\tau} \mathbf{s}_{\tau}') + \rho_{\tau} \mathbf{s}_{\tau} \mathbf{s}_{\tau}' \tag{A.27}$$

approximates $[\nabla_{\mathbf{x}}^2 f(\mathbf{x}^{\tau})]^{-1}$, with $\mathbf{s}_{\tau} = \mathbf{x}^{\tau+1} - \mathbf{x}^{\tau}$, $\mathbf{y}_{\tau} = \nabla_{\mathbf{x}} f(\mathbf{x}^{\tau+1}) - \nabla_{\mathbf{x}} f(\mathbf{x}^{\tau})$, and $\rho_{\tau} = (\mathbf{y}_{\tau}' \mathbf{s}_{\tau})^{-1}$. The essential steps of this method are summarized in Algorithm 8.

Algorithm 8 The BFGS method for UOPs.

Input: $f, \mathbf{x}_0, \varepsilon > 0$, and $\tau_{\max}$
1: $\tau \leftarrow 0$; $\mathbf{x} \leftarrow \mathbf{x}_0$; $H \leftarrow \mathbf{I}$
2: **repeat**
3: $\quad \tau \leftarrow \tau + 1$
4: $\quad \mathbf{p} \leftarrow -H \nabla_{\mathbf{x}} f(\mathbf{x})$
5: $\quad \bar{\eta} = \arg\min_{\eta} \phi(\eta)$ s.t. $0 \leq \eta \leq \eta_{\max}$, where $\phi(\eta) = f(\mathbf{x} - \eta \nabla_{\mathbf{x}} f(\mathbf{x}))$
6: $\quad \mathbf{x}^* \leftarrow \mathbf{x} + \bar{\eta} \mathbf{p}$ and $f^* \leftarrow f(\mathbf{x}^*)$
7: $\quad \mathbf{s} \leftarrow \mathbf{x}^* - \mathbf{x}$ and $\mathbf{y} = \nabla_{\mathbf{x}} f(\mathbf{x}_*) - \nabla_{\mathbf{x}} f(\mathbf{x})$
8: $\quad \rho_{\tau} \leftarrow (\mathbf{y}_{\tau}' \mathbf{s}_{\tau})^{-1}$
9: $\quad H \leftarrow (I - \rho \mathbf{s} \mathbf{y}') H (I - \rho \mathbf{y} \mathbf{s}') + \rho \mathbf{s} \mathbf{s}'$
10: $\quad \mathbf{x} \leftarrow \mathbf{x}^*$
11: **until** $\|\nabla_{\mathbf{x}} f(\mathbf{x}^*)\| < \varepsilon$ or $\tau = \tau_{\max}$
12: **return** $\mathbf{x}^*$ and f^*.

A.8 Single-point metaheuristics for optimization

All algorithms that have been presented so far for UOPs depend on assumptions such as: continuity and derivative information of the objective function or convexity of the objective function and of the search space. These assumptions can not be satisfied for many real-world optimization problems, which are problems characterized by high dimensionality, discontinuities, lack of derivative information, and disjoint search spaces. In addition, none of these approaches can be applied for solving black-box optimization problems, which are characterized by the lack of information related to the analytical form of the functions involved in the problem.

This section introduces two single-point metaheuristics to overcome these drawbacks, namely: hill-climbing and simulated annealing. These metaheuristics are zero-order methods, since these algorithms only use the values of the objective function. Hill-climbing and simulated annealing can be viewed as alternative methods when the classical optimization methods can not be applied (see Simon, 2013). To discuss these two single-point metaheuristics, consider an UOP such as defined in (A.4).

A.8.1 Hill-climbing

Hill-climbing (HC) is a local search method that uses a simple strategy to evolve a single-point in the search space. At each iteration, a candidate solution $\mathbf{x}^c$ is selected by performing a small perturbation in the current solution $\mathbf{x}^{\tau}$. This perturbation can be implemented by simply sampling $\mathbf{x}^c$ from the neighborhood of $\mathbf{x}^{\tau}$ or by adding a small random vector, $\mathbf{w}$, to the current solution: $\mathbf{x}^c = \mathbf{x}^{\tau} + \mathbf{w}$. If the new solution provides a better value for the objective function, then the new solution becomes the current solution. Otherwise, the current solution remains as the reference point for a new perturbation. HC is defined by the map

$$\mathbf{x}^{\tau+1} = \mathscr{A}(\mathbf{x}^{\tau}) = \begin{cases} \mathbf{x}^c & f(\mathbf{x}^c) < f(\mathbf{x}^{\tau}) \\ \mathbf{x}^{\tau} & \text{otherwise} \end{cases} \tag{A.28}$$

for $\tau = 0, 1, 2, \ldots$. HC has some weaknesses, it usually terminates in a local minimum and the minimum found depends on the initial solution. In addition, this method has no mechanism to escape of local minima. These drawbacks can eventually be overcome by introducing multiple initializations and a memory designed to keep the best solution found over all iterations. The essential steps of this metaheuristic are summarized in Algorithm 9.

Algorithm 9 Hill-climbing for UOPs.

Input: $f, \mathbf{x}_0$, and $\tau_{\max}$
1: {Hill-climbing}
2: $\tau \leftarrow 0$; $\mathbf{x} \leftarrow \mathbf{x}_0$; $f_{\mathbf{x}} \leftarrow f(\mathbf{x}_0)$
3: **repeat**
4: $\quad \tau \leftarrow \tau + 1$
5: $\quad \mathbf{x}^c \leftarrow \text{Perturbation}(\mathbf{x})$ and $f^c \leftarrow f(\mathbf{x}^c)$
6: $\quad$ **if** $f^c < f_{\mathbf{x}}$ **then**
7: $\quad\quad \mathbf{x} \leftarrow \mathbf{x}^c$ and $f_{\mathbf{x}} \leftarrow f^c$
8: $\quad$ **end if**
9: **until** $\tau = \tau_{\max}$
10: **return** $\mathbf{x}$ and $f_{\mathbf{x}}$.

Input: $f, \mathbf{x}_0$, and $\tau_{\max}$
1: {Multi-start Hill-climbing}
2: $\mathbf{x}^* \leftarrow \mathbf{x}^0$ and $f^* \leftarrow f(\mathbf{x}^0)$
3: **repeat**
4: $\quad \mathbf{x} \leftarrow \text{HillClimbing}(f, \mathbf{x}^0, \tau_{\max})$ and $f_{\mathbf{x}} \leftarrow f(\mathbf{x})$
5: $\quad$ **if** $f_{\mathbf{x}} < f^*$ **then**
6: $\quad\quad \mathbf{x}^* \leftarrow \mathbf{x}$ and $f^* \leftarrow f_{\mathbf{x}}$
7: $\quad$ **end if**
8: $\quad$ Set a new $\mathbf{x}^0 \in \mathbb{S}$
9: **until** some termination condition is met
10: **return** $\mathbf{x}^*$ and f^*.

A.8.2 Simulated annealing

Annealing is a physical process for obtaining low energy states of a solid in a heat bath. The process contains the following two steps: (1) increase the temperature of the heat bath to a maximum value at which the solid melts and (2) decrease carefully the temperature of the heat bath until the particles arrange themselves in the ground state of the solid. In the liquid phase all particles of the solid arrange themselves randomly. In the ground state the particles are arranged in a highly structured lattice and the energy of the system is minimal. The ground state of the solid is obtained only if the maximum temperature is sufficiently high and the cooling is done sufficiently slow. Otherwise the solid will be frozen into a meta-stable state rather than into the ground state.

Simulated annealing (SA) is a single-point metaheuristic inspired by the annealing process of physical systems. SA was proposed by Kirkpatrick, Gelatt & Vecchi (1983) to find optimal solutions of problems related to computer design. Independently, SA was proposed by Cerny (1985) to solve the well-known travelling salesman problem. SA is formulated as an algorithm whose goal is to find a solution with minimal objective value among a potentially large number of feasible solutions. It is assumed an analogy between a many-particle physical system and the optimization problem (A.4) based on the following equivalences:

Solutions $\times$ states. Solutions of the optimization problem are equivalent to states of a physical system.

Objective value $\times$ energy. The objective function value of a solution is equivalent to the energy of a state.

Based on these equivalences, SA is implemented using the well-known Metropolis algorithm (see Metropolis et al., 1953; Hastings, 1970). SA realizes a random search in terms of a non-homogeneous Markov chain, which not only accepts changes that improve the objective value of a solution, but also keeps some changes that are not ideal. In fact, for a minimization problem, any changes that decrease the objective value of a solution will be accepted. However, some changes that increase the objective value of a solution will also be accepted with a certain probability. This strategy works as a mechanism to escape of local minima.

The essential steps of SA are described as follows and can be summarized in Algorithm 10. Suppose that the current solution is $\mathbf{x}^\tau$ and consider a fixed temperature T. A mechanism specific is applied to randomly generate a candidate solution $\mathbf{x}^c$ from $\mathbf{x}^\tau$. Once again, this perturbation can be implemented by sampling $\mathbf{x}^c$ from the neighborhood of $\mathbf{x}^\tau$. The candidate solution $\mathbf{x}^c$ is accepted from $\mathbf{x}^\tau$ by applying the following probability:

$$\Pr(\mathbf{x}^\tau, \mathbf{x}^c) = \min\left\{1, \frac{\exp[f(\mathbf{x}^\tau) - f(\mathbf{x}^c)]}{T}\right\}. \tag{A.29}$$

SA is defined by the map

$$\mathbf{x}^{\tau+1} = \mathscr{A}(\mathbf{x}^\tau) = \begin{cases} \mathbf{x}^c & \text{with probability } \Pr(\mathbf{x}^\tau, \mathbf{x}^c) \\ \mathbf{x}^\tau & \text{otherwise} \end{cases} \tag{A.30}$$

for $\tau = 0, 1, 2, \ldots$. It continues by selecting and testing new solutions, and setting its current solutions in this way. The temperature T is reduced and the process is repeated again. Note that, at high temperatures a solution less favorable will be accepted with high probability. As previously cited, this feature means that SA can escape from local minima. At very low temperatures a solution less favorable will be accepted with low probability and SA becomes more like a greedy algorithm. SA terminates when the temperature is very low (near zero). If this cooling is done sufficiently slow, the algorithm has a high probability to obtain a globally optimal solution.

Algorithm 10 Simulated annealing for UOPs.

Input: f and $L_{\max}$

Initialize T and $\mathbf{x}$
$\mathbf{x}^* \leftarrow \mathbf{x}$ and $f^* \leftarrow f(\mathbf{x})$ {best solution found so far}
repeat
  for $l \in \{1, \ldots, L_{\max}\}$ **do**
    $\mathbf{x}^c \leftarrow \text{Perturbation}(\mathbf{x})$ and $f^c \leftarrow f(\mathbf{x}^c)$
    $\Delta f \leftarrow f(\mathbf{x}) - f(\mathbf{x}^c)$
    if $\Delta f \geq 0$ **then**
      $\mathbf{x} \leftarrow \mathbf{x}^c$
    else
      $r \sim \text{Unif}(0,1)$ {random number sampled from $(0,1)$}
      if $r < \exp(\Delta f / T)$ **then**
        $\mathbf{x} \leftarrow \mathbf{x}^c$
      end if
    end if
    if $f(\mathbf{x}) < f^*$ **then**
      $\mathbf{x}^* \leftarrow \mathbf{x}$ and $f^* \leftarrow f(\mathbf{x})$
    end if
  end for
  $T \leftarrow \text{CoolingSchedule}(T)$ {define the new value of T}
until some termination condition is met
return $\mathbf{x}^*$ and f^*.

A.9 Population-based metaheuristics for optimization

Two scientific fields will be discussed in this section: swarm and evolutionary computation. These terms are used to describe two categories of population-based metaheuristic algorithms that lie in the intersection between optimization and computational intelligence. Swarm and evolutionary computation are usually adopted in problems where classical optimization methods can not be applied such as black-box optimization problems or optimization problems in which probably neither the objective function nor the constraints are differentiable.

A.9.1 Swarm computation

Appendix B discusses swarm algorithms to solve UOPs. Appendix C provides a survey on constraint handling methods that have been adopted over the years to solve COPs, including special attention on how these methods can be incorporated in swarm algorithms to deal with constrained search spaces. Next subsection is exclusively concerned with evolutionary computation for UOPs. All methods discussed in Appendix C can be also incorporated in evolutionary algorithms to deal with COPs.

A.9.2 Evolutionary computation

Evolutionary computation encompasses a set of optimization algorithms that emulate evolutionary processes based on the principle of the survival of the fittest from Darwin's theory on the origin of species by means of natural selection (Darwin, 1859).

In nature, individuals have to adapt to their environment in order to survive in a process called evolution, in which those features that make an individual more stronger and suited to compete are preserved when it reproduces, and those features that make it weaker are eliminated. Such features are controlled by units called genes which form sets called chromosomes. Over subsequent generations not only the fittest individuals survive, but also their fittest genes which are transmitted to their descendants during the sexual recombination process which is called crossover (Coello, 2002).

Evolution and natural selection are simulated in a computer as a method to solve optimization problems. The general idea of this method is to use a population of individuals to encode solutions of an given problem and to manipulate these individuals by using genetic operators as: selection, crossover, and mutation. The aim is to

reach a good approximation of a global minimum for the problem as an individual which evolved through the generations.

There are three main paradigms within evolutionary computation, whose motivations and origins were independent from each other: genetic algorithms, evolution strategies, and evolutionary programming (Holland, 1975; Back & Schwefel, 1993; Fogel, 1994; Yao, Liu & Lin, 1999; Beyer & Schwefel, 2002; Engelbrecht, 2007; Simon, 2013). However, the current trend has been to decrease the difference among these three paradigms and refer, in generic terms, simply to evolutionary algorithms when talking about any of them. In general, some elements are required to describe an evolutionary algorithm (Coello, 2002):

Representation. A suitable representation of the potential solutions of the problem to be solved.

Initial population. A mechanism to generate an initial population of individuals representing candidate solutions.

Fitness function. A fitness function that plays the role of the environment, rating solutions in terms of their fitness.

Natural selection. A selection operator that chooses the parents that will reproduce.

Crossover and Mutation. Evolutionary operators that alter the composition of children.

Parameters. Values for various parameters that the algorithm uses like population size and probabilities to be used during the crossover and mutation processes.

In a general way, to solve a optimization problem using an evolutionary algorithm the following steps should be performed: (1) generate a random initial population of individuals; (2) select the fittest individuals based on their fitness to reproduce; (3) apply the crossover and mutation operators to generate new individuals (offspring or children); (4) loop this process until a stop condition is satisfied. These essential steps are summarized in Algorithm 11.

Algorithm 11 Generic evolutionary algorithm for UOPs.

Input: f, population size, and other parameter values.
1: Generate randomly an initial population of solutions
2: Calculate the fitness of each solution in the initial population.
3: **repeat**
4: Select a pair of solutions (or parents) to create two offspring using crossover
5: Apply mutation to each offspring
6: The new population is formed by all offspring
7: Calculate the fitness of each solution in the new population.
8: **until** some termination condition is met
9: **return** The best solution and its fitness.

A.10 The no-free-lunch theorem

The no-free-lunch theorem (Wolpert & Macready, 1997) states that, if no a priori assumption can be made about an optimization problem that must be solved, no optimization strategy can be expected to perform better than any other. In fact, all optimization algorithms perform exactly the same when averaged over all possible optimization problems. This result is an impossibility theorem that tells us that there is no strategy of any kind that outperforms all others on all optimization problems. Put in another way, a general-purpose universal optimization strategy is theoretically impossible. The only way that a strategy can outperform another if it is specialized to the specific problem under consideration.

Appendix B

Swarm Computation for Unconstrained Optimization

This appendix presents a short review about swarm algorithms for solving unconstrained optimization problems (UOPs). Basic concepts are discussed, including neighborhood systems for exchanging information between particles. A convergence analysis of the standard PSO is presented along with guidelines for parameter selection. The review aims to provide concepts and results related to the main topic of research of this thesis.

B.1 Introduction

Swarm computation has increasingly become an important tool for solving optimization problems. The advantages of these approaches over the traditional techniques from non-linear programming (discussed in Appendix A) are their robustness and flexibility. The first two swarm algorithms proposed in the literature were: ant colony optimization (ACO) and particle swarm optimization (PSO). ACO was introduced by Dorigo (1992); Dorigo, Maniezzo & Colorni (1996) and it has been widely used for solving combinatorial optimization problems (see Dorigo, Birattari & Stützle (2006) for a nice introduction in ACO).

PSO was introduced by Kennedy & Eberhart (1995) and it has been widely used for solving optimization problems in continuous search space (continuous optimization problems). It is a population-based optimization algorithm and its original idea was inspired by the social behaviour of some species of animals to work as a whole in locating desirable positions in a given area. This seeking behaviour was associated with that of a search for solutions to a given optimization problem. Therefore, PSO is motivated by the simulation of social behaviour, instead of evolution as in evolutionary algorithms. In addition, like a population-based algorithm, PSO evolves a population of solutions for a given optimization problem, unlike hill-climbing and simulated annealing that are single-point algorithms. The main advantage of population-based approaches over single-point strategies is that the large number of members that make up the population makes the technique resilient to the problem of local minima. PSO has been designed to address continuous optimization problems that cannot be tackled by traditional techniques from non-linear programming. These problems are black-box optimization problems and optimization problems characterized by discontinuities, lack of derivative information, and disjoint search spaces.

The interest in swarm computation has grown steadily. Many researchers have contributed to this field by proposing new algorithms, applications, and developing empirical and theoretical studies on the effects of various parameters and aspects of the proposed algorithms. Besides ACO and PSO, other examples of swarm algorithms are: bare bones particle swarm optimization (Kennedy, 2003) and fully informed particle swarm (Mendes, Kennedy & Neves, 2004; Kennedy & Mendes, 2006). Examples of swarm algorithms recently developed are: gravitational search algorithm, artificial bee colony algorithm, and the firefly algorithm (see Simon, 2013, Chap. 17).

B.2 Particle swarm optimization and its early variants

Consider an UOP such as defined in Eq. (1.4). The original PSO (Kennedy & Eberhart, 1995) can be viewed as an iterative process that evolves a population of solutions in the search space of the problem according to a prescribed set of instructions and a termination criterion. Given a population $\mathbf{P}^{\tau}$ of solutions in the search space, a

new population $\mathbf{P}^{\tau+1}$ of solutions is obtained by applying the instructions of the PSO on $\mathbf{P}^{\tau}$. This iterative process can be described by an algorithmic map

$$\mathscr{A} : \mathbb{S} \to \mathbb{S} \qquad \mathbf{P}^{\tau} \mapsto \mathbf{P}^{\tau+1} = \mathscr{A}(\mathbf{P}^{\tau}). \tag{B.1}$$

Formally, given a initial population $\mathbf{P}^0$, $\mathscr{A}$ generates a sequence $\mathbf{P}^1, \mathbf{P}^2, \ldots$ of populations in the search space. The transformation of $\mathbf{P}^{\tau}$ into $\mathbf{P}^{\tau+1}$ through the map constitutes an iteration of the PSO and the index τ ($\tau = 0, 1, 2, 3, \ldots$) represents the iteration counter. The population $\mathbf{P}^{\tau}$ is referred to as swarm and its individuals are referred to as particles. Each particle is characterized by four vectors $(\mathbf{x}_k^{\tau}, \mathbf{v}_k^{\tau}, \mathbf{p}_k^{\tau}, \mathbf{g}^{\tau})$, where the index k ($k = 1, \ldots, K$) represents the label of a particle in $\mathbf{P}^{\tau}$, K is the population size, and:

1. (Position) $\mathbf{x}_k^{\tau} = (x_{k1}^{\tau}, \ldots, x_{kD}^{\tau})'$ is the position of a particle in the τ-th iteration. This is a potential solution to the problem and it is used to evaluate the particle quality (in terms of objective value) in the τ-th iteration.

2. (Velocity) $\mathbf{v}_k^{\tau} = (v_{k1}^{\tau}, \ldots, v_{kD}^{\tau})'$ is the velocity of a particle in the τ-th iteration. This is the direction and length of movement of a particle in the τ-th iteration.

3. (Pbest position) $\mathbf{p}_k^{\tau} = (p_{k1}^{\tau}, \ldots, p_{kD}^{\tau})'$ is the personal-best position of a particle in the τ-th iteration. This is the best position (in terms of objective value) that the particle has visited until iteration τ. The role of this vector is to store the knowledge of the best solution found by the particle.

4. (Gbest position) $\mathbf{g}^{\tau} = (g_1^{\tau}, \ldots, g_D^{\tau})'$ is the global-best position in the τ-th iteration. This is the best position (in terms of objective value) that any particle in $\mathbf{P}^{\tau}$ has visited until iteration τ. The role of this vector is to store the knowledge of the best solution found by the swarm as whole. The global-best position is the best personal-best position in the swarm.

PSO is initialized with a population of particles with random positions and velocities. On the initialization,

$$\mathbf{v}_k^0 \approx \mathbf{0} = (0, \ldots, 0)' \text{ and } \mathbf{p}_k^0 = \mathbf{x}_k^0 \tag{B.2}$$

for all k. The global-best position is initialized as

$$\mathbf{g}^0 = \text{BEST}(\mathbf{p}_k^0 : k = 1, \ldots, K) = \arg\min\{f(\mathbf{p}_k^0) : k = 1, \ldots, K\}. \tag{B.3}$$

The velocity of a particle is updated by

$$v_{kd}^{\tau+1} = v_{kd}^{\tau} + c_1(p_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_1 + c_2(g_d^{\tau} - x_{kd}^{\tau}) \cdot r_2 \tag{B.4}$$

and its position is updated by

$$x_{kd}^{\tau+1} = x_{kd}^{\tau} + v_{kd}^{\tau+1} \tag{B.5}$$

for all k, d ($d = 1, \ldots, D$), and $\tau \geq 0$. The second part of the Eq. (B.4) ($c_1(p_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_1$) is the cognitive part[1], which represents the private thinking of a particle. The third part ($c_2(g_d^{\tau} - x_{kd}^{\tau}) \cdot r_2$) is the social part, which represents the collaboration among the particles. The parameters c_1, c_2 are positive constants used to scale the contribution of the cognitive and social components of the learning process of a particle and r_1, r_2 are random numbers uniformly distributed in the range $[0, 1]$. After updating the position, the personal-best position is updated as follows

$$\mathbf{p}_k^{\tau+1} = \text{BEST}(\mathbf{x}_k^{\tau+1}, \mathbf{p}_k^{\tau}). \tag{B.6}$$

Finally, the global-best position is updated by

$$\mathbf{g}^{\tau+1} = \text{BEST}(\mathbf{p}_k^{\tau+1} : k = 1, \ldots, K). \tag{B.7}$$

The process is repeated until some stopping criterion is met. At the end of this process, PSO returns the global-best position and its objective value. Note that Eq. (B.7) informs that the swarm can be thought of as a social network having a communication structure, since the best solution is reported for all particles. In fact, all particles are interconnected and each particle can communicate with every other particles. Each particle is attracted towards the best solution found by swarm. The essential steps of PSO are summarized in Algorithm 12.

[1]Cognitive abilities are processes involved in acquisition and understanding of knowledge, formation of beliefs, and decision making.

Algorithm 12 Particle swarm optimization for UOPs.

Input: D, K, c_1, c_2, and f

$\tau \leftarrow 0$
for $k \in \{1,\ldots,K\}$ **do**
  Initialize $\mathbf{x}_k$ and set $\mathbf{v}_k \approx \mathbf{0}$
  $\mathbf{p}_k \leftarrow \mathbf{x}_k$
end for
$\mathbf{g} \leftarrow \text{BEST}(\mathbf{p}_k : k = 1,\ldots,K)$
repeat
  $\tau \leftarrow \tau + 1$
  for $k \in \{1,\ldots,K\}$ **do**
    for $d \in \{1,\ldots,D\}$ **do**
      Update v_{kd} (Eq. (B.4))
      Update x_{kd} (Eq. (B.5))
    end for
    $\mathbf{p}_k \leftarrow \text{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  $\mathbf{g} \leftarrow \text{BEST}(\mathbf{p}_k : k = 1,\ldots,K)$
until some termination condition is met
return $\mathbf{x}^* = \mathbf{g}$ and $f^* = f(\mathbf{g})$.

In summary, PSO utilizes a population of particles during the optimization process. Each particle moves in the search space obeying dynamic rules to update its position. Each particle searches a global solution to the problem learning from its own past experience and from the experiences of the other particles. The swarm as a whole explores the search space, first at random, and then, when better solutions are found and communicated, the swarm begins to converge by refining its search until a good enough solution is found.

A desirable property of a swarm algorithm in solving an UOP is that it generates a sequence of populations converging to a global optimal solution of the problem. Formally, this means that: for all $\varepsilon > 0$,

$$\lim_{\tau \to \infty} \Pr(\|\mathbf{g}^\tau - \bar{\mathbf{x}}\| < \varepsilon) = 1 \tag{B.8}$$

where $\bar{\mathbf{x}}$ is a global minimum. The original PSO presented a satisfactory performance for simple optimization problems. However, it also presented some deficiencies when it was applied to harder problems involving large search spaces and multimodal functions. This deficiencies can be listed as:

1. Swarm explosion effect. This refers to the uncontrolled increase of magnitude of the velocities, resulting in swarm divergence.
2. Inability to converge. This refers to the fact that the swarm was unable either to achieve convergence on a promising position or perform a refined search around it.
3. Inability to escape from local minima. This refers to the fact that particles were highly susceptible to being trapped in local minima.

The refinements developed to address these deficiencies are discussed in the following subsections.

B.2.1 PSO with velocity clamping

The swarm explosion effect was addressed by using strict bounds for velocity clamping at desirable levels, preventing particles from taking extremely large steps from their current positions. More specifically, a user-defined maximum velocity threshold was considered. After determining the new velocity of each particle with Eq. (B.4), the following restriction is applied prior to the position update with Eq. (B.5):

$$-v_{\max,d} \leq v_{kd}^{\tau+1} \leq v_{\max,d} \tag{B.9}$$

where $v_{\max,d} \propto (U_d - L_d)$, $d = 1,\ldots,D$. In case of violation, the corresponding velocity component is defined as

$$v_{kd}^{\tau+1} = v_{\max,d} \text{ if } v_{kd}^{\tau+1} > v_{\max,d} \text{ or } v_{kd}^{\tau+1} = -v_{\max,d} \text{ if } v_{kd}^{\tau+1} < -v_{\max,d}. \tag{B.10}$$

Velocity clamping offered a solution to the problem of swarm explosion. However, the choice of these parameters ($v_{\max,d}$, $d = 1,\ldots,D$) require some care because their influences on the swarm diversity and consequently on the balance between global search (exploration or diversification) and local search (exploitation or intensification). Global search is the ability to test various regions in the problem search space in order to locate a good solution, hopefully the global optimum. Local search is the ability to concentrate the search around of a promising candidate solution in order to locate the optimum precisely. Finally, swarm diversity is defined as a quantitative measure that reflects how many different particles (or different solutions) exist in the swarm (or solution population).

B.2.2 PSO with inertia weight

The second deficiency presented by PSO was the inability to converge. This problem was addressed by the introduction of a new parameter in the original PSO. In fact, Shi & Eberhart (1998) introduced a new parameter, called inertia weight, into the original PSO. This new parameter plays the role of balancing the global search and local search, at the same time that promotes good convergence ability, if this parameter is well tuned. The result is a PSO with the following update rule for the velocity of a particle:

$$v_{kd}^{\tau+1} = w v_{kd}^{\tau} + c_1(p_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_1 + c_2(g_d^{\tau} - x_{kd}^{\tau}) \cdot r_2 \tag{B.11}$$

for all k, d, and $\tau \geq 0$. For $w \geq 1$, the velocity increases over time up to the maximum velocity (when velocity clamping is used) and the swarm diverges. For $w < 1$, the velocity decreases until to reach zero (depending on the values of c_1 and c_2). Large values of w facilitate global search, with increased diversity. Small values of w promotes local search. However, too small values of w completely eliminate the swarm exploration ability.

Simulation results showed the significant and effective impact of this new parameter on the PSO. Initial implementations of PSO with inertia weight used a static value for w during the search process of a global optimal solution. Later implementations made use of dynamically changing inertia values, usually starting with large inertia values which decreases over time to smaller values. Shi & Eberhart (1998) suggested that a decreasing inertia weight present a better performance. In general, a linearly decreasing scheme for w can be defined as follows:

$$w^{\tau} = w_{\max} - (w_{\max} - w_{\min}) \cdot \frac{\tau}{\tau_{\max}}. \tag{B.12}$$

For instance, w can be started at 0.9 and be linearly decreased until 0.4.

B.2.3 PSO with neighborhood system

The use of inertia weight provided a PSO algorithm with convergence capability, but did not solve the deficiency related to the fact that particles were still highly susceptible to being trapped in local minima. The swarm reaches convergence, but after a number of iterations, shows total diversity loss. As a consequence, further exploration is not possible and the particles can perform only local search around their convergence points. Although the effect of fast convergence can be satisfactory in some optimization problem, such as optimization of unimodal and convex functions, it becomes detrimental in optimization problems involving multimodal functions. This deficiency can be attributed to the global information exchange scheme that allows each particle to know the global best position at each iteration. Using this scheme, all particles assume new positions in regions of the search space that are related to the same information, reducing the exploration capability of the swarm.

This problem was addressed by introducing the concept of neighborhood of a particle. Following this idea, each particle has a neighborhood consisting of a set of particles which it can communicate with. The neighborhood of a particle is denoted by $\mathcal{N}_k$ and the neighborhood system $\mathcal{N} = \{\mathcal{N}_k : k = 1,\ldots,K\}$ represents a communication structure often thought of as a social network. There is a number of different schemes to connect the particles. Most implementations use one of two simple sociometric principles. The first, called global topology (also known as global-best model or Gbest model), connects each particle in the population with all others, i.e., $\mathcal{N}_k = \mathbf{P}$ for all k. The second, called local topology (also known as local-best model or Lbest model), creates a neighborhood for each particle comprising generally of the particle itself and L neighbors in the population $\mathbf{P}$, i.e., $\mathcal{N}_k \subset \mathbf{P}$ for all k, with $|\mathcal{N}_k| = L + 1 < K$.

Each particle keeps the memory of the best position (in terms of objective value) that the particle has visited until iteration τ (the personal-best position or pbest position). This vector stores the knowledge of the best solution found by the particle, as a personal learning. In addition, the particles use the neighborhood system to exchange

information between them. As a result, each particle also keeps the memory of the best position (in terms of objective value) that any particle in its neighborhood has visited until iteration τ. This is the local-best position in the τ-th iteration (or lbest position). This vector stores the knowledge of the best solution found by its neighborhood, as a social learning. The personal-best position, as already established, is denoted by $\mathbf{p}_k^\tau = (p_{k1}^\tau, \ldots, p_{kD}^\tau)'$ and the local-best position is denoted by $\mathbf{n}_k^\tau = (n_{k1}^\tau, \ldots, n_{kD}^\tau)'$. Finally, the global-best position is defined as the best local-best position. In local-best model, $\mathbf{p}_k^\tau$, $\mathbf{n}_k^\tau$, and $\mathbf{g}^\tau$ are updated as follows:

$$\begin{aligned}
\text{(Pbest position)} \quad \mathbf{p}_k^{\tau+1} &= \text{BEST}(\mathbf{x}_k^{\tau+1}, \mathbf{p}_k^\tau) && \text{(B.13)}\\
\text{(Lbest position)} \quad \mathbf{n}_k^{\tau+1} &= \text{BEST}(\mathbf{p}_l^{\tau+1} : l \in \mathcal{N}_k) && \text{(B.14)}\\
\text{(Gbest position)} \quad \mathbf{g}^{\tau+1} &= \text{BEST}(\mathbf{n}_k^{\tau+1} : k = 1, \ldots, K). && \text{(B.15)}
\end{aligned}$$

Figures B.1 - B.4 show examples of neighborhood systems (Mendes, Kennedy & Neves, 2004; Kennedy & Mendes, 2006):

1. In the star structure, all particles are interconnected as illustrated in Figure B.1. Each particle can communicate with every other particles. In this case, each particle is attracted towards the best solution found by the entire swarm. The first implementation of the PSO and its first variants used a star structure. As this structure is a global topology, all algorithms that use this structure are referred to as global-best PSO (or Gbest PSO). The global-best PSO has been shown to converge faster than other neighborhood systems, but with greater susceptibility to be trapped in local minima. Generally, the global-best PSO performs better for unimodal problems.

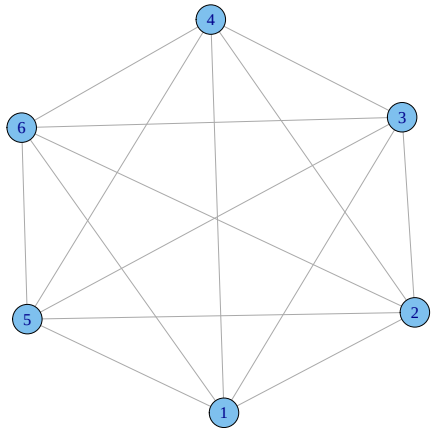

Figure B.1: Star structure.

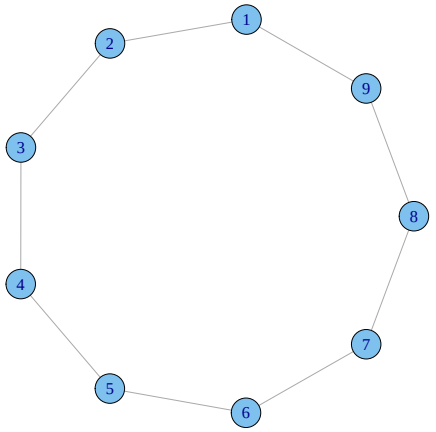

Figure B.2: Ring structure.

2. In the ring structure, each particle communicates with its L immediate neighbors. In the case of $L = 2$, each particle communicates with its immediately adjacent neighbors as illustrated in Figure B.2. The ring structure is a local topology, then the resulting PSO is referred to as local-best PSO (or Lbest PSO). Each particle attempts to imitate its best neighbor by moving towards the best solution found within of its neighborhood. Since the information flow through the structure is smaller due to a smaller connectivity between the particles, the speed of convergence towards an optimal solution is slower compared to the star structure, but with the advantage of lower susceptibility to be trapped in local minima. As a consequence, the local-best PSO provides better performance in terms of the quality of solutions than the global-best PSO for multimodal problems.

3. The Von Neumann structure is also a local topology, where the particles are connected in a grid structure as illustrated in Figure B.3. It is important to note that the Von Neumann structure is a toroidal grid, because it

is connected circularly in both dimensions. It means that all particles has four neighbors. For example, the neighbors of the particle 1 are the particles 2, 5, 4, and 13. Once again, the neighbors of the particle 15 are the particles 16, 3, 14, and 11.

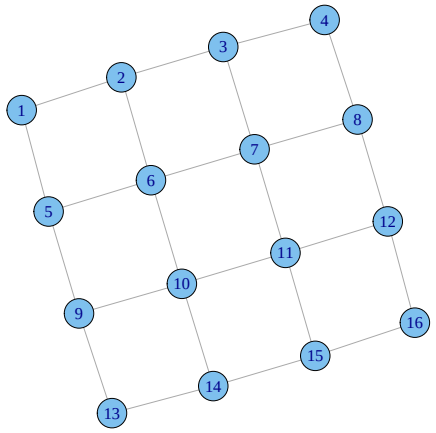

Figure B.3: Von Neumann structure.

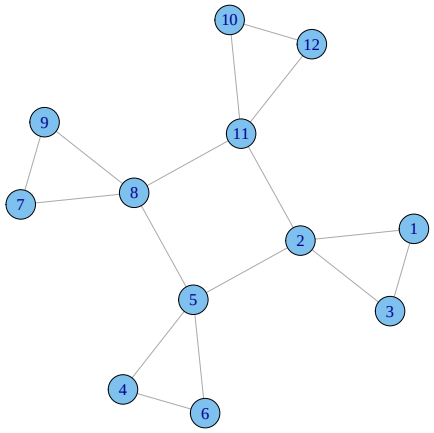

Figure B.4: Four-clusters structure.

4. The clusters structure is also a local topology, where the structure is organized by clusters of particles as illustrated in Figure B.4. The clusters are connected in ring, considering one particle in each cluster as a link point. Within of each cluster, the particles can be connected in ring or in star.

In summary, the velocity of a particle and its position in local-best PSO with inertia weight are updated as

$$v_{kd}^{\tau+1} = wv_{kd}^{\tau} + c_1(p_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_1 + c_2(n_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_2 \tag{B.16}$$

$$x_{kd}^{\tau+1} = x_{kd}^{\tau} + v_{kd}^{\tau+1} \tag{B.17}$$

for all k, d, and $\tau \geq 0$. The velocity of a particle and its position in global-best PSO with inertia weight are updated as

$$v_{kd}^{\tau+1} = wv_{kd}^{\tau} + c_1(p_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_1 + c_2(g_{d}^{\tau} - x_{kd}^{\tau}) \cdot r_2 \tag{B.18}$$

$$x_{kd}^{\tau+1} = x_{kd}^{\tau} + v_{kd}^{\tau+1} \tag{B.19}$$

for all k, d, and $\tau \geq 0$. The essential steps of PSO with inertia weight are summarized in Algorithm 13.

B.3 Particle swarm optimization with constriction factor

Clerc & Kennedy (2002) introduced a PSO with constriction factor. In this case, the velocity of a particle is updated as follows:

$$v_{kd}^{\tau+1} = \begin{cases} \chi[v_{kd}^{\tau} + \varphi_1(p_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_1 + \varphi_2(n_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_2] & \text{for the local-best model} \\ \chi[v_{kd}^{\tau} + \varphi_1(p_{kd}^{\tau} - x_{kd}^{\tau}) \cdot r_1 + \varphi_1(g_{d}^{\tau} - x_{kd}^{\tau}) \cdot r_2] & \text{for the global-best model} \end{cases} \tag{B.20}$$

for all k, d, and $\tau \geq 0$. The parameters φ_1, φ_2 are positive constants used to scale the contribution of the cognitive and social components and r_1, r_2 are random numbers uniformly distributed in the range $[0,1]$. The parameter χ, called constriction factor, is given by

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \tag{B.21}$$

Algorithm 13 Particle swarm optimization with inertia weight for UOPs.

Input: $D, K, \mathcal{N}, w, c_1, c_2$, and f
{Local-best PSO}
$\tau \leftarrow 0$
for $k \in \{1,\ldots,K\}$ **do**
  Initialize $\mathbf{x}_k$ and set $\mathbf{v}_k \approx \mathbf{0}$
  $\mathbf{p}_k \leftarrow \mathbf{x}_k$
end for
for $k \in \{1,\ldots,K\}$ **do**
  $\mathbf{n}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
end for
repeat
  $\tau \leftarrow \tau + 1$
  for $k \in \{1,\ldots,K\}$ **do**
    for $d \in \{1,\ldots,D\}$ **do**
      Update v_{kd} (Eq. (B.16))
      Update x_{kd} (Eq. (B.17))
    end for
    $\mathbf{p}_k \leftarrow \text{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  for $k \in \{1,\ldots,K\}$ **do**
    $\mathbf{n}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
  end for
until some termination condition is met
$\mathbf{g} \leftarrow \text{BEST}(\mathbf{n}_k : k = 1,\ldots,K)$
return $\mathbf{x}^* = \mathbf{g}$ and $f^* = f(\mathbf{g})$.

Input: $D, K, \mathcal{N}, w, c_1, c_2$, and f
{Global-best PSO}
$\tau \leftarrow 0$
for $k \in \{1,\ldots,K\}$ **do**
  Initialize $\mathbf{x}_k$ and set $\mathbf{v}_k \approx \mathbf{0}$
  $\mathbf{p}_k \leftarrow \mathbf{x}_k$
end for
$\mathbf{g} \leftarrow \text{BEST}(\mathbf{p}_k : k = 1,\ldots,K)$
repeat
  $\tau \leftarrow \tau + 1$
  for $k \in \{1,\ldots,K\}$ **do**
    for $d \in \{1,\ldots,D\}$ **do**
      Update v_{kd} (Eq. (B.18))
      Update x_{kd} (Eq. (B.19))
    end for
    $\mathbf{p}_k \leftarrow \text{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  $\mathbf{g} \leftarrow \text{BEST}(\mathbf{p}_k : k = 1,\ldots,K)$
until some termination condition is met
return $\mathbf{x}^* = \mathbf{g}$ and $f^* = f(\mathbf{g})$.

where $\varphi = \varphi_1 + \varphi_2 > 4$. The parameter φ is commonly adjusted to 4.1 (for example, if $\varphi_1 = \varphi_2 = 2.05$) and $\chi = 0.7298$ (Clerc & Kennedy, 2002).

Using vector notation, the velocity of a particle and its position in local-best model are updated as

$$\mathbf{v}_k^{\tau+1} = \chi\left[\mathbf{v}_k^{\tau} + \varphi_1(\mathbf{p}_k^{\tau} - \mathbf{x}_k^{\tau}) \odot \mathbf{r}_1 + \varphi_2(\mathbf{n}_k^{\tau} - \mathbf{x}_k^{\tau}) \odot \mathbf{r}_2\right] \quad \text{(B.22)}$$

$$\mathbf{x}_k^{\tau+1} = \mathbf{x}_k^{\tau} + \mathbf{v}_k^{\tau+1} \quad \text{(B.23)}$$

for all k and $\tau \geq 0$, where $\odot$ represents the componentwise vector multiplication and $\mathbf{r}_{1,2}$ are vectors of random numbers uniformly distributed in $[0,1]$. The velocity of a particle and its position in global-best model are updated as

$$\mathbf{v}_k^{\tau+1} = \chi\left[\mathbf{v}_k^{\tau} + \varphi_1(\mathbf{p}_k^{\tau} - \mathbf{x}_k^{\tau}) \odot \mathbf{r}_1 + \varphi_2(\mathbf{g}^{\tau} - \mathbf{x}_k^{\tau}) \odot \mathbf{r}_2\right] \quad \text{(B.24)}$$

$$\mathbf{x}_k^{\tau+1} = \mathbf{x}_k^{\tau} + \mathbf{v}_k^{\tau+1} \quad \text{(B.25)}$$

for all k and $\tau \geq 0$. The PSO with constriction factor is currently known as standard PSO (Bratton & Kennedy, 2007) and it is algebraically equivalent to PSO with inertia weight, where

$$w = \chi,\ c_1 = \chi\varphi_1, \text{ and } c_2 = \chi\varphi_2. \quad \text{(B.26)}$$

However, PSO with constriction factor is distinguished in literature due to its theoretical properties discussed in details by Clerc & Kennedy (2002).

B.4 Fully informed particle swarm

Mendes, Kennedy & Neves (2004) developed a fully informed particle swarm (FIPS), where each particle is influenced by all of its neighbors rather than just by the best one in its neighborhood. The FIPS algorithm was further improved by Kennedy & Mendes (2006) and it can be considered a generalization of the standard PSO

(Bratton & Kennedy, 2007). While the standard PSO adds two terms to the velocity and divides the constant φ in half to weight each one of them, FIPS distributes the weight of φ across the entire neighborhood. In this case, the velocity and the position of a particle are updated as

$$\mathbf{v}_k^{\tau+1} = \chi\left[\mathbf{v}_k + \sum_{l\in\mathcal{N}_k} \frac{c_l(\mathbf{p}_l - \mathbf{x}_k)\odot\mathbf{r}_l}{|\mathcal{N}_k|}\right] \tag{B.27}$$

$$\mathbf{x}_k^{\tau+1} = \mathbf{x}_k^{\tau} + \mathbf{v}_k^{\tau+1} \tag{B.28}$$

for all k and $\tau \geq 0$, where $|\mathcal{N}_k|$ is the number of neighbors of a particle and $\mathbf{r}_l$ are vectors of random numbers uniformly distributed in $[0,1]$. FIPS has been applied to solve multimodal optimization problems providing very good results (Kennedy & Mendes, 2006). It is considered a powerful optimization algorithm with good capacity to maintain the diversity among particles due to the strong social influence to update the velocity. Next section presents a detailed discussion about convergence and parameter selection for the PSO with inertia weight.

B.5 Convergence analysis of the PSO algorithm

Theoretical analyses of the PSO algorithm have been offered in the literature. Some of these studies use results of the theory of dynamical systems (Clerc & Kennedy, 2002; Trelea, 2003) and others use results of the theory of stochastic processes (Jiang, Luo & Yang, 2007; Poli, 2009). It has been hard to find general results. Nevertheless, some progress has been made by considering simplifying assumptions such as: swarm with a single particle in one dimension, stagnation (i.e., when no improved solution is found and memories are kept constant), and deterministic behaviour without randomness.

Clerc & Kennedy (2002) studied the behaviour of one particle in one dimension during stagnation (without randomness). Under these conditions the swarm is described as a linear dynamical system in discrete time. The dynamics of the state of the particle (position and velocity) can be determined by finding the eigenvalues and eigenvectors of the dynamic matrix of the system. The proposed model predicts that the particle will converge to equilibrium if the magnitude of the eigenvalues is smaller than 1. This study presented a model for the PSO algorithm, containing a set of parameters to control the convergence of the system. Trelea (2003) developed, under the same assumptions adopted by Clerc & Kennedy (2002), an dynamic analysis of a PSO model with four-parameter. This study identified regions in the parameter space where the model exhibits different dynamic behaviours such as: explosion, stability (or equilibrium point), non-oscillatory convergence, convergence with harmonic oscillations, zigzagging convergence with harmonic oscillations, and zigzagging convergence.

Applying the approach used by Trelea (2003), a dynamic analysis is presented for the PSO with inertia weight. For convenience, stagnation is initially assumed as a simplifying assumption, but subsequently this hypothesis will be relaxed. During stagnation, each particle behaves independently, since each dimension is updated independently. Thus, the behaviour of each particle can be analysed in isolation, considering one-particle swarm ($K = 1$) in one dimension ($D = 1$). In this case, the PSO with inertia weight can be described as follows:

$$v_{\tau+1} = wv_\tau + c_1(p - x_\tau)\cdot r_1 + c_2(g - x_\tau)\cdot r_2 \tag{B.29}$$

$$x_{\tau+1} = x_\tau + v_{\tau+1}. \tag{B.30}$$

For convenience in notation, note that τ is being used as a subscript index rather than superscript as done in last sections. The deterministic model of this PSO is obtained by setting

$$r_1 = r_2 = \frac{1}{2} \qquad c = c_1 = c_2 \qquad \tilde{p}_x = \frac{p+g}{2}. \tag{B.31}$$

Using this notation, the deterministic PSO can be expressed as:

$$v_{\tau+1} = wv_\tau + c(\tilde{p}_x - x_\tau) \tag{B.32}$$

$$x_{\tau+1} = x_\tau + v_{\tau+1}. \tag{B.33}$$

Eqs. (B.32) and (B.33) can be combined and written in matrix form as

$$\mathbf{y}_{\tau+1} = \mathbf{A}\mathbf{y}_\tau + \tilde{p}_x\mathbf{C} \tag{B.34}$$

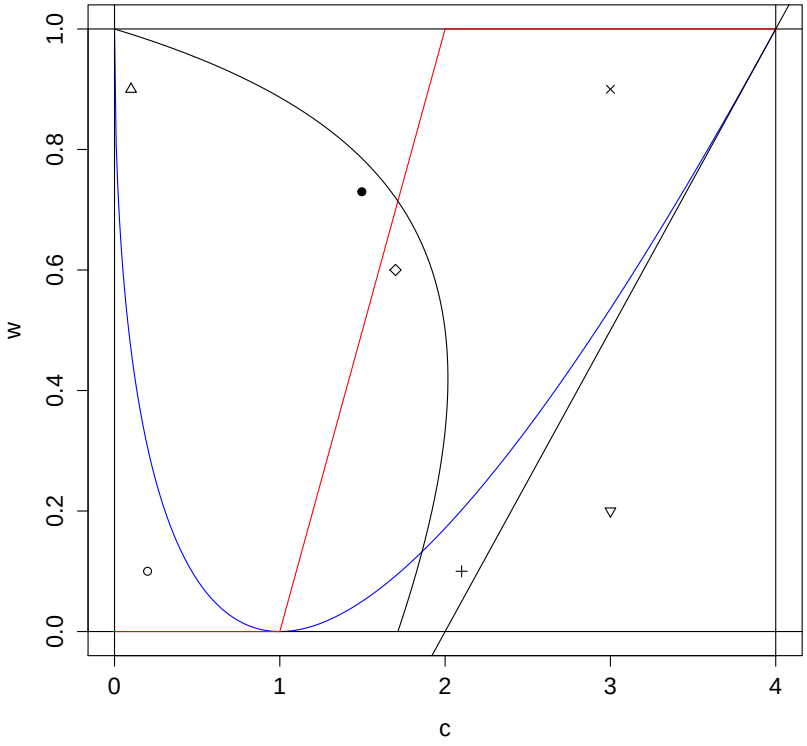

Figure B.5: The convergence domain in the (c,w)-parameter space.

where

$$\mathbf{y}_\tau = (x_\tau, v_\tau)' \qquad \mathbf{A} = \begin{pmatrix} 1-c & w \\ -c & w \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} c \\ c \end{pmatrix}. \tag{B.35}$$

In this context, $\mathbf{y}_\tau$ is the particle state, $\mathbf{A}$ is the dynamic matrix whose properties determine the time behaviour of the particle, $\tilde{p}_x$ is the external input used to drive the particle towards a specified position, and $\mathbf{C}$ is the input matrix that gives the effect of the external input on the particle state.

Consider the deterministic PSO defined by Eq. (B.34). If $\mathbf{y}_0 = \mathbf{y}_{\text{eq}} = (\tilde{p}_x, 0)'$, then $\mathbf{y}_\tau = \mathbf{y}_{\text{eq}} = (\tilde{p}_x, 0)'$ for all $\tau > 0$. The state $\mathbf{y}_{\text{eq}} = (\tilde{p}_x, 0)'$ is the equilibrium state of the particle. This result is intuitively reasonable. At equilibrium, the particle must have zero velocity and must be positioned at the attraction point $\tilde{p}_x$. In general, the initial state of the particle is not the equilibrium. Therefore, it is important to determine whether the particle will eventually reach the equilibrium and how the particle will move in the state space over time. The time behaviour of the particle depends on the eigenvalues of the dynamic matrix $\mathbf{A}$. The eigenvalues λ_1 and λ_2 of $\mathbf{A}$ (either real or complex) are the solutions of the characteristic equation associated to $\mathbf{A}$, derived as

$$\lambda^2 - \text{trace}(\mathbf{A})\lambda + \text{determinat}(\mathbf{A}) = \lambda^2 - (1 + w - c)\lambda + w = 0. \tag{B.36}$$

The necessary and sufficient condition for the stability of the equilibrium state is that both eigenvalues of $\mathbf{A}$ have magnitude less than 1. In this case, the equilibrium state will be an attractor, the particle will converge to its equilibrium, and the deterministic PSO will converge. Result B.1 provides a set of conditions to ensure the convergence of the particle in the deterministic PSO to its equilibrium state.

Result B.1 (Trelea (2003))**.** *Consider the deterministic PSO defined by Eq. (B.34). For any initial state, the particle will converge to its equilibrium state if and only if the parameters w and c are selected such that* $0 \leq w < 1$ *and* $0 < c < 2(1+w)$.

Figure B.5 shows the convergence domain in the (c,w)-parameter space. Two sets of parameters w and c are frequently used by users of PSO: (i) $w = 0.6$ and $c = 1.7$ was recommended by Trelea (2003) and (ii) $w = \chi =$

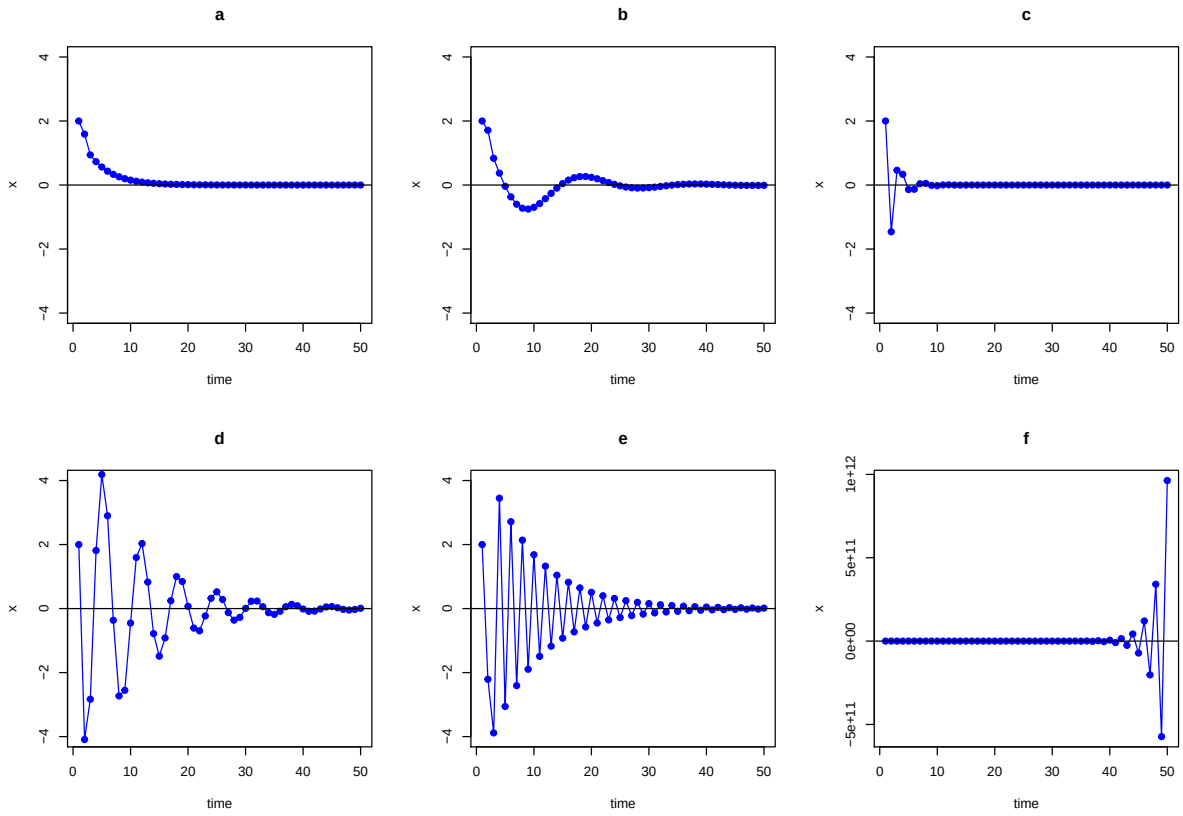

Figure B.6: (a) Non-oscillatory convergence ($\circ$, $c = 0.20$, $w = 0.10$). (b) Convergence with harmonic oscillations ($\triangle$, $c = 0.10$, $w = 0.90$). (c) Fast zigzagging convergence with harmonic oscillations ($\Diamond$, $c = 1.70$, $w = 0.60$). (d) Slow zigzagging convergence with harmonic oscillations ($\times$, $c = 3.00$, $w = 0.90$). (e) Zigzagging convergence ($+$, $c = 2.10$, $w = 0.10$). (f) Explosion ($\bigtriangledown$, $c = 3.00$, $w = 0.20$). See Figure B.5 to locate the point (c,w) according to the symbols $\circ, \triangle, \Diamond, \times, +$, and $\bigtriangledown$.

0.7298 and $c = 1.4961$ ($\varphi = 2c/\chi = 4.1$) was recommended by Clerc & Kennedy (2002). These points (c,w) are shown in Figure B.5 using the symbols $\Diamond$ and $\bullet$ respectively.

Before convergence, the particle exhibits different dynamic behaviours. In fact, the convergence domain is divided into areas, so that each area is associated with a different dynamic behaviour that can be classified as: non-oscillatory convergence; convergence with harmonic oscillations; zigzagging convergence with harmonic oscillations; and zigzagging convergence. Outside of the convergence domain, the particle diverges and the explosion effect is observed in deterministic PSO. Formally, the dynamic behaviour of the particle depends on whether the eigenvalues of $\mathbf{A}$ are real or complex and each one of these cases can be easily summarized. Consider the deterministic PSO defined by Eq. (B.34) and suppose that $0 \leq w < 1$ and $0 < c < 2(1+w)$. The dynamic behaviour of the particle is described as follows:

1. If $(1+w-c)^2 - 4w > 0$, then the eigenvalues of $\mathbf{A}$ are real and given by

$$\lambda_{1,2} = \frac{(1+w-c)}{2} \pm \frac{\sqrt{(1+w-c)^2 - 4w}}{2}. \tag{B.37}$$

   In addition, $|\lambda_{1,2}| < 1$ and $x_\tau = C_1\lambda_1^\tau + C_2\lambda_2^\tau + \tilde{p}_x$ for all $\tau \geq 0$.

2. If $(1+w-c)^2 - 4w = 0$, then the repeated eigenvalues of $\mathbf{A}$ are given by

$$\lambda = \lambda_1 = \lambda_2 = \frac{(1+w-c)}{2}. \tag{B.38}$$

   In addition, $|\lambda| < 1$ and $x_\tau = C_1\lambda^\tau + C_2\tau\lambda^\tau + \tilde{p}_x$ for all $\tau \geq 0$.

3. If $(1+w-c)^2-4w<0$, then the eigenvalues of $\mathbf{A}$ are complex and given by

$$\lambda_{1,2}=\frac{(1+w-c)}{2}\pm i\frac{\sqrt{4w-(1+w-c)^2}}{2}. \tag{B.39}$$

In addition, $|\lambda_{1,2}|<1$ and $x_\tau=w^\tau[C_1\sin\tau\theta+C_2\cos\tau\theta]+\tilde{p}_x$ for all $\tau\geq 0$, where

$$\theta=\arctan\left(\frac{\sqrt{4w-(1+w-c)^2}}{1+w-c}\right). \tag{B.40}$$

For all these cases, the values of C_1 and C_2 for any given trajectory are obtained from the initial conditions x_0 and v_0 of the particle. Finally, $\lim_{\tau\to\infty}x_\tau=\tilde{p}_x$ for all cases discussed. In order to illustrate this result, simulations of the dynamic behaviour of the particle for different parameter couples are shown in Figure B.6 (see also Figure B.5 to locate the point (c,w) according to the symbols $\circ,\triangle,\Diamond,\times,+$, and $\bigtriangledown$.). All simulations were performed with $\mathbf{y}_0=(x_0=2,v_0=-0.1)'$, $\tilde{p}_x=0$, and $\tau_{\max}=50$ iterations. It is important to note that the analysis presented so far does not consider that r_1 and r_2 are random numbers. Instead, these variables were defined as $1/2$. However, simulation studies show that the presence of random numbers just enhances the zigzagging tendency and slows down convergence, thus improving the state space exploration and preventing premature convergence to local minima. Then, qualitatively the analysis presented remains valid.

A rigorous analysis of the PSO algorithm should be done considering that r_1 and r_2 are random numbers and, consequently, x_τ should be seen as a random variable whose distribution should be investigated in order to reveal its functional form and its properties when $\tau\to\infty$. These problems are open problems in swarm computation. However, some concrete steps have been taken in this direction. Applying the approach used by Poli (2009), a set of coupled difference equations for the moments $\mathrm{E}(x_\tau),\mathrm{E}(x_\tau^2)$, and $\mathrm{E}(x_\tau x_{\tau-1})$ can be presented considering $r_{1,2}\sim\mathrm{Unif}(0,1)$. To get this set of coupled equations, the first step is combine the Eqs. (B.29) and (B.30) to obtain a difference equation for x_τ as follows:

$$x_{\tau+1}=(1+w-c_1r_1-c_2r_2)x_\tau-wx_{\tau-1}+pc_1r_1+gc_2r_2=ax_\tau-wx_{\tau-1}+b \tag{B.41}$$

where $a=a(r_1,r_2)=(1+w-c_1r_1-c_2r_2)$ e $b=b(r_1,r_2)=pc_1r_1+gc_2r_2$. Consequently, it follows that

$$x_{\tau+1}^2=a^2x_\tau^2+w^2x_{\tau-1}^2+2abx_\tau-2wbx_{\tau-1}-2wax_\tau x_{\tau-1}+b^2 \tag{B.42}$$

and

$$x_{\tau+1}x_\tau=ax_\tau^2-wx_\tau x_{\tau-1}+bx_\tau. \tag{B.43}$$

Note that the stagnation assumption has been assumed again in the context where a swarm with a single particle in one dimension is considered. When the expectation operator is applied to both sides of the Eqs (B.41), (B.42), and (B.43), a set of coupled difference equations for $\mathrm{E}(x_\tau),\mathrm{E}(x_\tau^2)$, and $\mathrm{E}(x_\tau x_{\tau-1})$ are given by:

$$\mathrm{E}(x_{\tau+1})=\mathrm{E}(a)\mathrm{E}(x_\tau)-w\mathrm{E}(x_{\tau-1})+\mathrm{E}(b) \tag{B.44}$$
$$\mathrm{E}(x_{\tau+1}^2)=\mathrm{E}(a^2)\mathrm{E}(x_\tau^2)+w^2\mathrm{E}(x_{\tau-1}^2)+2\mathrm{E}(ab)\mathrm{E}(x_\tau)-2w\mathrm{E}(b)\mathrm{E}(x_{\tau-1})-2w\mathrm{E}(a)\mathrm{E}(x_\tau x_{\tau-1})+\mathrm{E}(b^2) \tag{B.45}$$
$$\mathrm{E}(x_{\tau+1}x_\tau)=-w\mathrm{E}(x_\tau x_{\tau-1})+\mathrm{E}(a)\mathrm{E}(x_\tau^2)+\mathrm{E}(b)\mathrm{E}(x_\tau) \tag{B.46}$$

for all $\tau\geq 1$, where

$$\mathrm{E}(a)=1+w-c_1/2-c_2/2 \tag{B.47}$$
$$\mathrm{E}(a^2)=(1+w)^2-(1+w)(c_1+c_2)+c_1^2/3+c_2^2/3+c_1c_2/2 \tag{B.48}$$
$$\mathrm{E}(b)=c_1p/2+c_2g/2 \tag{B.49}$$
$$\mathrm{E}(b^2)=c_1^2p^2/3+c_2^2g^2/3+c_1c_2pg/2 \tag{B.50}$$
$$\mathrm{E}(ab)=(1+w)c_1p/2+(1+w)c_2g/2-c_1^2p/3-c_2^2g/3-c_1c_2p/4-c_1c_2g/4. \tag{B.51}$$

These set of equations can be integrated numerically if values for

$$\mathrm{E}(x_0),\mathrm{E}(x_1),\mathrm{E}(x_0^2),\mathrm{E}(x_1^2),\mathrm{E}(x_0x_1) \tag{B.52}$$

are known (initial conditions). In addition, the standard deviation of x_τ is defined by

$$\mathrm{SD}(x_\tau) = \sqrt{\mathrm{Var}(x_\tau)} = \sqrt{\mathrm{E}(x_\tau^2) - (\mathrm{E}(x_\tau))^2} \tag{B.53}$$

for all $\tau \geq 0$. Although the functional form of the distribution of x_τ is unknown, at least the first and second moments of x_τ can be calculated by using these results.

The stability analysis based on the difference equation for $\mathrm{E}(x_\tau)$ is similar to stability analysis of the deterministic PSO defined by Eq. (B.34). The characteristic equation associated to difference equation for $\mathrm{E}(x_\tau)$ is given by

$$\lambda^2 - (1 + w - c_1/2 - c_2/2)\lambda + w = 0 \tag{B.54}$$

which is equivalent to Eq. (B.36). According to Poli (2009), the PSO algorithm is order-1 stable if $\mathrm{E}(x_\tau)$ has a stable fixed point and $\mathrm{E}(x_\tau)$ converges to this fixed point. In addition, the PSO algorithm is order-2 stable if $\mathrm{E}(x_\tau)$, $\mathrm{E}(x_\tau^2)$, and $\mathrm{E}(x_\tau x_{\tau-1})$ have stable fixed points and $\mathrm{E}(x_\tau)$, $\mathrm{E}(x_\tau^2)$, and $\mathrm{E}(x_\tau x_{\tau-1})$ converge to these fixed points. As a consequence, if PSO is order-2 stable, then it is ensured that the focus of the search volume reach the equilibrium and the spread of the search volume does not expand without limit (swarm stability). Thus, the following results can be presented.

Result B.2 (Jiang, Luo & Yang (2007); Poli (2009))**.** *Consider the PSO with inertia weight and with a single particle in one dimension, where $r_{1,2} \sim Unif(0,1)$, and assume that $c = c_1 = c_2$. Under the stagnation assumption, the following can be stated:*

1. *If $\mathrm{E}(x_0) = \tilde{p}_x = (p+g)/2$, then $\mathrm{E}(x_\tau) = \tilde{p}_x$ for all $\tau \geq 1$.*
2. *For any value of $\mathrm{E}(x_0)$, $\lim_{\tau\to\infty} \mathrm{E}(x_\tau) = \tilde{p}_x$ if and only if $0 \leq w < 1$ and $0 < c < 2(1+w)$.*
3. *This PSO is order-1 stable if and only if $0 \leq w < 1$ and $0 < c < 2(1+w)$.*

Result B.2 ensures that PSO with inertia weight is order-1 stable if the parameters of this algorithm are selected appropriately. Analysing all fixed points of the set of coupled equations for $\mathrm{E}(x_\tau), \mathrm{E}(x_\tau^2)$, and $\mathrm{E}(x_\tau x_{\tau-1})$, it is also possible to show that PSO with inertia weight is order-2 stable if the parameters are appropriately selected again.

Result B.3 (Jiang, Luo & Yang (2007); Poli (2009))**.** *Consider the PSO with inertia weight and with a single particle in one dimension, where $r_{1,2} \sim Unif(0,1)$, and assume that $c = c_1 = c_2$. Under the stagnation assumption, this PSO is order-2 stable if and only if $0 \leq w < 1$ and $0 < c < 12(w^2-1)/(5w-7)$. In addition, the stable fixed points of $\mathrm{E}(x_\tau)$, $\mathrm{E}(x_\tau^2)$, and $SD(x_\tau)$ are respectively given by*

$$\lim_{\tau\to\infty} \mathrm{E}(x_\tau) = \tilde{p}_x = \frac{p+g}{2} \tag{B.55}$$

$$\lim_{\tau\to\infty} \mathrm{E}(x_\tau^2) = \tilde{p}_{x^2} = \left[1 + \frac{2(\tilde{\nu} - \tilde{\mu}^2)}{\tilde{\Delta}}\right]\tilde{p}^2 - \left[\frac{4p(\tilde{\nu} - \tilde{\mu}^2)}{\tilde{\Delta}}\right]\tilde{p} + \left[\frac{2p^2(\tilde{\nu} - \tilde{\mu}^2)}{\tilde{\Delta}}\right] \tag{B.56}$$

$$\lim_{\tau\to\infty} SD(x_\tau) = \tilde{p}_{sd} = \frac{1}{2}\sqrt{\frac{2(\tilde{\nu} - \tilde{\mu}^2)}{\tilde{\Delta}}}|p-g| \tag{B.57}$$

where $\tilde{\mu} = c/2$, $\tilde{\nu} = c^3/3$, and $\tilde{\Delta} = [c^2(5w-7) - 12c(w^2-1)]/[6(1+w)]$.

Note that, in general, $(\tilde{p}_x)^2 \neq \tilde{p}_{x^2}$ and $\tilde{p}_{sd} \neq 0$. Consequently, the process x_τ does not converge in mean square to $\tilde{p}_x$ during stagnation. However, if $p = g$, then $\mathrm{E}(x_\tau) = g$, $\mathrm{E}(x_\tau^2) = g^2$, and $\mathrm{SD}(x_\tau) = 0$ for all τ. To complete our analysis of convergence, the following situation should be considered: the values of p_τ (pbest) and g_τ (gbest) are now constantly updated during the search process. This means that the stagnation assumption is relaxed. As a consequence, the following results can be presented.

Result B.4 (Jiang, Luo & Yang (2007))**.** *Consider the PSO with inertia weight and with a single particle in one dimension, where $r_{1,2} \sim Unif(0,1)$. Assume that $c = c_1 = c_2$, $0 \leq w < 1$, and $0 < c < 12(w^2-1)/(5w-7)$. If $g_\tau \to g^*$ and $p_\tau \to g^*$ with probability 1, then the process x_τ converges in mean square to g^*.*

Result B.5 (Jiang, Luo & Yang (2007))**.** *Consider the PSO with inertia weight and K particles in a D-dimensional search space. Assume that $0 \leq w < 1$ and $0 < c_1 + c_2 < 24(w^2-1)/(5w-7)$. If $g_{kd}^\tau \to g_d^*$ and $p_{kd}^\tau \to g_d^*$ with probability 1, then x_{kd}^τ converges in mean square to g_d^* for all $d = 1,\ldots,D$ and $\mathbf{x}_k^\tau$ converges in mean square to $\mathbf{g}^* = (g_1^*,\ldots,g_D^*)$. Finally, this conclusion applies to each particle, thus the swarm as a whole will converge to $\mathbf{g}^*$ when τ tends to infinity.*

A search process allowing changes in $\mathbf{p}_k^\tau$ for all k over time implies that the spatial extension of the search volume decreases over time. In fact, Blackwell (2005) suggested that the maximum spatial extension of a swarm along any axis decreases exponentially with time. It is important to emphasize that this section provides an analysis to prove that the particles converge to a stable point. Note that there is no guarantee that the equilibrium point is a high quality solution (for instance, a local minimum). Hence, there is no guarantee that PSO with inertia weight is locally convergent. In fact, the study developed in this section is not a proof of convergence to a local minimum, but only states that the swarm will reach an equilibrium point with high probability. A formal proof of convergence to a local minimum is currently an open problem in swarm computation (a hard problem still unsolved).

B.6 Bare bones particle swarm optimization

Bare bones PSO (BBPSO) is a variant of the PSO algorithm originally introduced by Kennedy (2003). Compared to PSO, it is simpler and has only two parameters to be tuned by users, namely: the swarm size and the neighborhood system. BBPSO uses a probability distribution to update the position of a particle instead of adding a velocity in the current position as is done in the standard PSO (Bratton & Kennedy, 2007). Global-best and local-best models are possible such as in the PSO. The position of a particle is updated as

$$\mathbf{x}_k^{\tau+1} = \boldsymbol{\mu}_k^{\tau+1} + \boldsymbol{\sigma}_k^{\tau+1} \odot \mathbf{z} \tag{B.58}$$

for all k and $\tau \geq 0$, where

- $\boldsymbol{\mu}_k^{\tau+1} = 0.5 \cdot (\mathbf{p}_k^\tau + \mathbf{g}^\tau)$ or $\boldsymbol{\mu}_k^{\tau+1} = 0.5 \cdot (\mathbf{p}_k^\tau + \mathbf{n}_k^\tau)$
- $\boldsymbol{\sigma}_k^{\tau+1} = |\mathbf{p}_k^\tau - \mathbf{g}^\tau|$ or $\boldsymbol{\sigma}_k^{\tau+1} = |\mathbf{p}_k^\tau - \mathbf{n}_k^\tau|$
- $\mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I}) = (N(0,1), \overset{iid}{\ldots}, N(0,1))'$.

As already discussed $\mathbf{p}_k^\tau$, $\mathbf{n}_k^\tau$, and $\mathbf{g}^\tau$ are respectively the personal best, local best, and global best positions, and $\mathbf{z}$ is a random variable which has multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix $\mathbf{I}$, such as discussed in Chapter 3.

Result B.6. *Consider the global-best BBPSO. The distribution of* $\mathbf{x}_k^\tau$ *is given by* $\mathbf{x}_k^\tau \sim \mathbf{N}\left(\boldsymbol{\mu}_k^\tau, \boldsymbol{\Sigma}_k^\tau\right)$ *for all* k *and* $\tau > 0$*, where*

$$\mathrm{E}(\mathbf{x}_k^\tau) = \boldsymbol{\mu}_k^\tau = \frac{1}{2} \cdot (\mathbf{p}_k^{\tau-1} + \mathbf{g}^{\tau-1}) \qquad \textit{and} \qquad \mathrm{Cov}(\mathbf{x}_k^\tau) = \boldsymbol{\Sigma}_k^\tau = diag(|\mathbf{p}_k^{\tau-1} - \mathbf{g}^{\tau-1}|). \tag{B.59}$$

A similar result can be reported for the local-best model.

Result B.7. *Consider the global-best BBPSO. Under the stagnation assumption, each particle behaves independently, since each dimension is updated independently. Therefore*

$$\mathrm{E}(\mathbf{x}_k^\tau) = \frac{1}{2} \cdot (\mathbf{p} + \mathbf{g}) \qquad \textit{and} \qquad \mathrm{Cov}(\mathbf{x}_k^\tau) = diag(|\mathbf{p} - \mathbf{g}|) \tag{B.60}$$

for all k *and* $\tau > 0$*. As a consequence, this algorithm is order-2 stable. A similar result can be reported for the local-best model.*

The original idea of the BBPSO was inspired by a plot of the distribution of positions attained by a single particle in the PSO under the influence of the personal and global best positions fixed in two constant values. This plot is presented in Figure B.7, considering a PSO with inertia weight in two dimensions, $\mathbf{p} = (2,1)'$, and $\mathbf{g} = (1,3)'$. Note that the empirical distribution resembles a bell curve, such as a normal density, but with heavier tails than those of a normal distribution. Nevertheless, the original proposal of BBPSO considers the normal density to sample new positions during the optimization process. On the other hand, the empirical distribution seems actually centered midway between $\mathbf{p}$ and $\mathbf{g}$, having the absolute difference between $\mathbf{p}$ and $\mathbf{g}$ as an important parameter for scaling the amplitude of the trajectory of the particle.

Result B.8. *Consider the global-best BBPSO. If the stagnation assumption is relaxed,* $\mathbf{g}_k^\tau \to \mathbf{g}^*$ *with probability* 1*, and* $\mathbf{p}_k^\tau \to \mathbf{g}^*$ *with probability* 1*, then* $\mathbf{x}_k^\tau$ *converges in mean square to* $\mathbf{g}^*$ *for all* k*. This conclusion applies to each particle, thus the swarm as a whole will converge to* $\mathbf{g}^*$ *when* τ *tends to infinity. A similar result can be reported for the local-best model.*

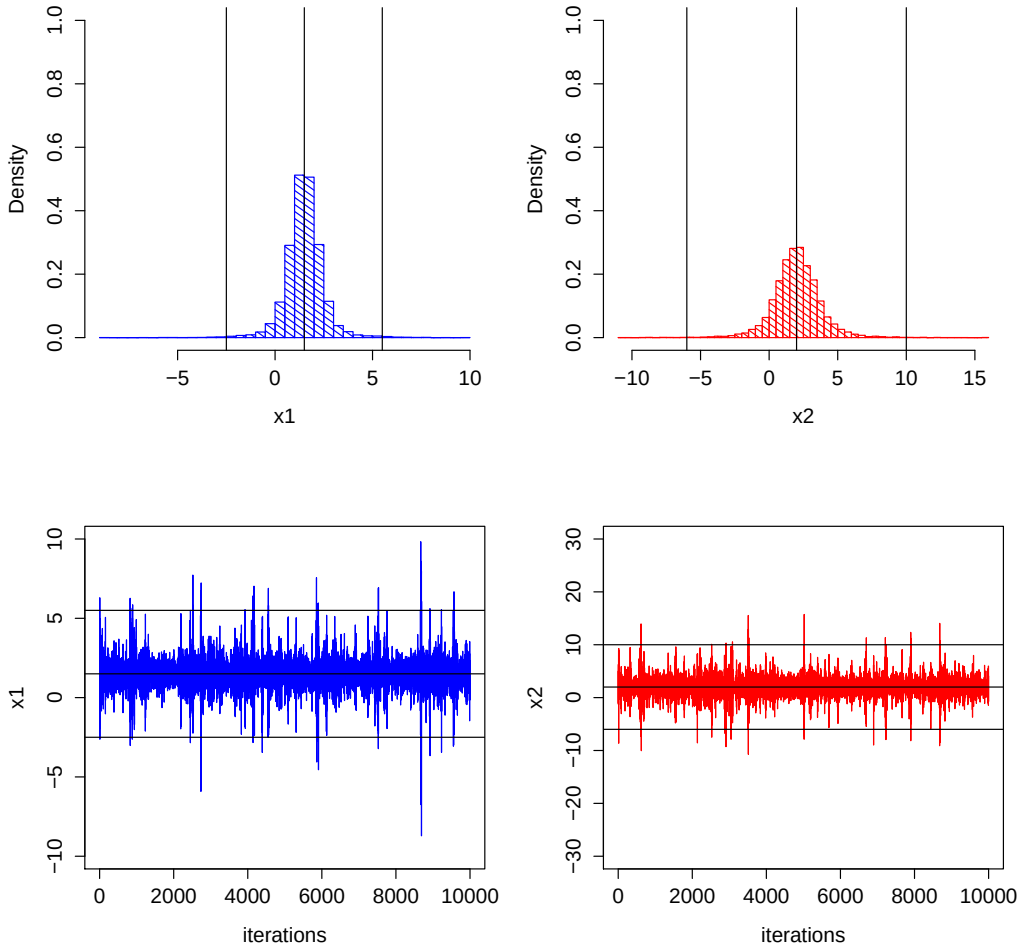

Figure B.7: Distribution of points sampled by a PSO with inertia weight constituted of a single particle in two dimensions, and considering $\mathbf{p} = (2,1)'$ and $\mathbf{g} = (1,3)'$. Note that $\boldsymbol{\mu} = \frac{1}{2}(\mathbf{p}+\mathbf{g}) = (\frac{3}{2},2)'$ and $\boldsymbol{\sigma} = |\mathbf{p}-\mathbf{g}| = (1,2)'$.

The search for solutions of an optimization problem using BBPSO is the result of a constructive cooperation between particles applying explicit probabilistic models whose parameters are defined in terms of the information associated with some promising candidate solutions, namely the values related with the positions pbest, lbest, and gbest.

B.7 Bare bones particle swarm with heavy-tailed distributions

The design of algorithms in swarm and evolutionary computation has shown that to avoid the premature convergence problem during an optimization process of multimodal functions, the algorithms must provide a proper equilibrium between global search (exploration) and local search (exploitation) at any stage of the search. Several strategies have been investigated, including the use of heavy-tailed distributions for sampling the search space of the problem. It has been conjectured that heavy-tailed distributions can increase the chances of a particle (or individual in a population) to move away from a local optimum. The general idea is to allow exploration in regions far away from the current position. Following this idea, two works in evolutionary computation using heavy-tailed distributions can be highlighted, namely the study developed by Yao, Liu & Lin (1999) and also the study developed Lee & Yao (2004). Yao, Liu & Lin (1999) introduced a fast evolutionary programming (FEP) which uses mutations based on the Cauchy distribution instead of mutations based on the Gaussian distribution, which is used

in the classical evolutionary programming (CEP). Experimental results show that FEP performs much better than CEP for multimodal functions with many local minima, while it remains comparable in performance to CEP for unimodal functions and multimodal functions with only a few local minima. This work report the fact that Cauchy mutation performs better when the current search point is far away from the global minimum, while Gaussian mutation is better at finding a local minimum in a good region. Lee & Yao (2004) introduced a evolutionary programming (EP) with mutations based on the Lévy distribution. This distribution is symmetrical with respect to zero and has two parameters α and γ, where γ is the scaling factor satisfying $\gamma > 0$ and α controls the shape of the distribution satisfying $0 < \alpha \leq 2$. Its density resembles a bell curve, such as a Gaussian density, but with heavier tails (α can be used to control the heaviness of the tails). The Lévy distribution includes some models as special cases: the Cauchy distribution when $\alpha = 1$ and the normal distribution when $\alpha = 2$. An algorithm for generating Lévy random numbers was introduced in Mantegna (1994). The work developed by Lee & Yao (2004) reports empirical evidences that the performance of the EP with Lévy mutation was better than that of the CEP for multimodal functions with many local minima since that Lévy mutation is more general and flexible than Cauchy and Gaussian mutations.

BBPSO has shown potential for solving optimization problems defined on continuous search spaces. However, it suffers from the premature convergence problem when solving multimodal problems. In order to address this drawback and improve the performance of the original algorithm, some strategies have been developed. New variants of BBPSO have been proposed (Richer & Blackwell, 2006; Krohling & Mendel, 2009; Hsieh & Lee, 2010; Blackwell, 2012; Liu, Ding & Wang, 2014; Campos, Krohling & Enriquez, 2014) and some of these variants are discussed in Sections B.7 and B.8.

Richer & Blackwell (2006) investigated the effectiveness of using the Lévy distribution in BBPSO. Based on a series of trials, this work reports that the Lévy BBPSO with $\alpha = 1.4$ reproduces the behaviour of the standard PSO (Bratton & Kennedy, 2007). These results support the conjecture that heavy-tailed distributions to update the position of a particle provide an increase in its ability to move away from a local minimum, improving at the same time the balance between exploration and exploitation.

Krohling & Mendel (2009) introduced a BBPSO with a particular jump strategy, when no improvement on the value of the objective function is observed. This jump strategy was implemented based on the Gaussian or Cauchy distributions. The algorithm was tested on a well-known suite of benchmark multimodal functions and the results were compared with those obtained by the BBPSO algorithm. Simulation results have shown that the BBPSO algorithm with jump strategy has performed well in all functions investigated. Krohling & Mendel (2009) also pointed that the improved performance was due to a successful number of Gaussian or Cauchy jumps, with a performance slightly better for the case of Cauchy jumps. This result is essentially compatible with the conjecture that distributions with heavy tails increase the chances of a particle to escape from local minima, exploring the search space without losing exploitation in promising regions.

Blackwell (2012) formulated the dynamic update rule of the PSO as a second-order stochastic difference equation. This formulation was used to derive general expressions for search focus, search spread, and swarm stability at stagnation. The results were applied to three swarm algorithms: the standard PSO, PSO with discrete recombination, and BBPSO. Blackwell (2012) proposed a generalized BBPSO (GBBPSO) such that the search focus and the search spread can each one be chosen from the global or local neighborhoods. The position of a particle is updated as

$$\mathbf{x}_k^{\tau+1} = \boldsymbol{\mu}_k^{\tau+1} + \alpha \boldsymbol{\delta}_k^{\tau+1} \odot \mathbf{z} \tag{B.61}$$

for all k and $\tau \geq 0$, where $\alpha > 0$[2] and

- $\boldsymbol{\mu}_k^{\tau+1} = \text{BEST}(\mathbf{p}_l^\tau : l \in \mathcal{N}_k)$ or $\text{BEST}(\mathbf{p}_k^\tau : k = 1, \ldots, K)$
- $\boldsymbol{\delta}_k^{\tau+1} = |\mathbf{p}_k^\tau - \mathbf{n}_k^\tau|$ or $|\mathbf{p}_k^\tau - \mathbf{g}^\tau|$ or $|\mathbf{p}_{k+1}^\tau - \mathbf{p}_{k-1}^\tau|$ (mod K)
- $\mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I}) = (N(0,1), \overset{iid}{\ldots}, N(0,1))'$.

The theoretical analysis, presented by Blackwell (2012), predicts a no-collapse condition when $\alpha > \alpha_c = 0.65$. Collapse is a swarm pathology, where the particles approach each other faster than the swarm as a whole approaches a local minimum. As a result, the convergence toward a limit point becomes exponentially slow and the swarm stagnates. Another important result is that the fastest rate of convergence of the GBBPSO occurs at the critical

[2]Note that in the GBBPSO algorithm, the parameter α has a conceptual meaning completely different from that the parameter α has in the Lévy distribution.

value α_c. Experimental results confirm that GBBPSO situated at the edge of collapse is comparable to the standard PSO and PSO with discrete recombination. The main loop of GBBPSO is shown in Algorithm 14.

Algorithm 14 GBBPSO and GBBPSOwJ.

Input: $\alpha > 0$

{GBBPSO}
loop
  for $k \in \{1,\ldots,K\}$ **do**
    $\boldsymbol{\mu}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
    $\boldsymbol{\delta}_k \leftarrow |\mathbf{p}_k - \mathbf{n}_k|$
    for $d \in \{1,\ldots,D\}$ **do**
      $z \sim N(0,1)$
      $x_{kd} \leftarrow \mu_{kd} + \alpha\delta_{kd} \cdot z$
    end for
    $\mathbf{p}_k \leftarrow \text{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  for $k \in \{1,\ldots,K\}$ **do**
    $\mathbf{n}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
  end for
end loop

Input: $\alpha > 0$ and $0 \leq p_J < 1$

{GBBPSOwJ}
loop
  for $k \in \{1,\ldots,K\}$ **do**
    $\boldsymbol{\mu}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
    $\boldsymbol{\delta}_k \leftarrow |\mathbf{p}_k - \mathbf{n}_k|$
    for $d \in \{1,\ldots,D\}$ **do**
      $r \sim \text{Unif}(0,1)$
      if $r < p_J$ **then**
        $x_{kd} \sim \text{Unif}(L_d, U_d)$
      else
        $z \sim N(0,1)$
        $x_{kd} \leftarrow \mu_{kd} + \alpha\delta_{kd} \cdot z$
      end if
    end for
    $\mathbf{p}_k \leftarrow \text{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  for $k \in \{1,\ldots,K\}$ **do**
    $\mathbf{n}_k \leftarrow \text{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
  end for
end loop

Additional results indicated that the performance of the proposed algorithm can be still further improved with the use of an adaptive distribution with heavy tails. In fact, Blackwell (2012) also proposed a GBBPSO with jumps (GBBPSOwJ) following a different strategy from that which was considered by Krohling & Mendel (2009). GBBPSOwJ can be seen as the GBBPSO combined with a probabilistic jumping mechanism: a particle may jump uniformly in any dimension with probability p_J. This strategy can be seen as a partial re-initialization (since, in general, not every component undergoes a jump) or, alternatively, as a mechanism to fatten the tails of the normal distribution that generates new positions in the search space, allowing search in areas where the tails of the normal distribution are thin. This adaptive distribution allows exploration throughout the search volume at any stage of the optimization. The main loop of GBBPSOwJ is shown in Algorithm 14.

Hsieh & Lee (2010) also introduced a modified BBPSO. Empirical studies, considering a well-known set of test functions, have showed that the proposed algorithm is a competitive optimizer due to its good performance and fast convergence rate. The BBPSO proposed by Hsieh & Lee (2010) shows some similar characteristics to BBPSO proposed by Blackwell (2012), with the former having an additional parameter to focus the search.

B.8 Scale matrix adaptation bare bones particle swarm

Campos, Krohling & Enriquez (2014) recently introduced a BBPSO with scale matrix adaptation for solving unconstrained optimization problems (SMABBPSO for short). In the SMABBPSO, the position of a particle is selected from a multivariate t-distribution with a rule for adaptation of its scale matrix. The multivariate t-distribution is used in its hierarchical form as a scale mixture of normal distributions (Andrews & Mallows, 1974; Choy & Chan, 2008). The t-distribution has heavier tails than those of the normal distribution, which increases the ability of the particles to escape from a local optimum. The proposed approach includes the normal distribution as a particular case. A simple update rule was proposed to adapt the scale matrix associated with a particle, such that the best position found by any particle in its neighborhood is sampled with maximum likelihood in the next iteration. As a consequence, each particle selects new positions in the search space of the problem using an adaptive distribution and considering its accumulated learning until the current iteration without discarding any information so far. This approach uses an adaptive distribution with heavy tails in order to improve the balance between exploration and exploitation during the optimization process.

B.8.1 Swarm structure and dynamic rule

The swarm structure of the SMABBPSO is equivalent to the structure of the canonical BBPSO. SMABBPSO has a swarm $\mathbf{P}$ with K particles and a neighborhood system $\mathscr{N}$ defined by a local topology with $|\mathscr{N}_k|$ neighbors to each particle. Each particle is characterized by a vector $(\mathbf{x}_k, \mathbf{p}_k, \mathbf{n}_k, \mathbf{g})'$. The position of a particle is updated by

$$\mathbf{x}_k^{\tau+1}|\lambda = \boldsymbol{\mu}_k^{\tau+1} + \lambda^{-1/2}\sqrt{\boldsymbol{\Sigma}_k^{\tau+1}}\mathbf{z} \qquad \lambda \sim Ga\left(\frac{\nu}{2}, \frac{\nu}{2}\right) \quad (\nu > 0) \tag{B.62}$$

for all k and $\tau \geq 0$, where

- $\boldsymbol{\mu}_k^{\tau+1} = \mathbf{n}_k^{\tau} = \text{BEST}(\mathbf{p}_l^{\tau} : l \in \mathscr{N}_k)$
- $\boldsymbol{\Sigma}_k^{\tau+1} = (1-\beta)\boldsymbol{\Sigma}_k^{\tau} + \beta\mathbf{n}_k^{\tau}\mathbf{n}_k^{\tau,\prime}$, with $\boldsymbol{\Sigma}_k^0 = \mathbf{I}$ and $0 \leq \beta \leq 1$
- $\mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I}) = (N(0,1), \overset{iid}{\ldots}, N(0,1))'$.

The following results can be presented to clarify the proposed approach and its implementation.

Result B.9. *If the Eq. (B.62) is adopted to update the position of a particle, then* $\mathbf{x}_k^{\tau} \sim \mathbf{t}(\nu, \boldsymbol{\mu}_k^{\tau}, \boldsymbol{\Sigma}_k^{\tau})$ *for all* k *and* $\tau > 0$*, where*

$$\text{E}(\mathbf{x}_k^{\tau}) = \boldsymbol{\mu}_k^{\tau} = \mathbf{n}_k^{\tau-1} \qquad \textit{if } \nu > 1 \tag{B.63}$$

and

$$\text{Cov}(\mathbf{x}_k^{\tau}) = \left(\frac{\nu}{\nu-2}\right)\boldsymbol{\Sigma}_k^{\tau} = (1-\beta)\text{Cov}(\mathbf{x}_k^{\tau-1}) + \beta\left(\frac{\nu}{\nu-2}\right)\mathbf{n}_k^{\tau-1}\mathbf{n}_k^{\tau-1,\prime} \qquad \textit{if } \nu > 2. \tag{B.64}$$

The normal-distributed SMABBPSO can be obtained when $\lambda = 1$ *almost surely, or when* $\nu \to \infty$*. Therefore, the parameter* ν *can be used to control the heaviness of the tails of the distribution of* $\mathbf{x}_k^{\tau}$*.*

Result B.10. *For purposes of implementation, it is important to know that the Eq. (B.62) is equivalent to*

$$\mathbf{x}_k^{\tau+1}|\lambda = \boldsymbol{\mu}_k^{\tau+1} + \lambda^{-1/2}\mathbf{P}_k^{\tau+1}\sqrt{\boldsymbol{\Theta}_k^{\tau+1}}\mathbf{z} \qquad \lambda \sim Ga\left(\frac{\nu}{2}, \frac{\nu}{2}\right) \quad (\nu > 0) \tag{B.65}$$

for all k *and* $\tau \geq 0$*, where* $\mathbf{P}_k^{\tau+1}$ *is an orthogonal matrix whose columms are the eingenvectors of* $\boldsymbol{\Sigma}_k^{\tau+1}$ *and*

$$\sqrt{\boldsymbol{\Theta}_k^{\tau+1}} = diag(\theta_1^{1/2}, \ldots, \theta_D^{1/2}) \tag{B.66}$$

where $\theta_1, \ldots, \theta_D$ *are the eigenvalues associated. This result follows from the spectral decomposition of* $\boldsymbol{\Sigma}_k^{\tau+1}$ *(a* $D \times D$ *symmetric positive definite matrix). In addition, it follows that SMABBPSO can be implemented simply knowing how to simulate from univariate probability distributions.*

When a swarm algorithm is used to solve an optimization problem, each particle in the swarm is subject to a sequential learning process, where $\mathscr{L}_k^{\tau} = (\mathbf{n}_k^0, \mathbf{n}_k^1, \ldots, \mathbf{n}_k^{\tau})$ represents the accumulated learning of a particle until time τ (the iteration counter). If the neighborhood system chosen admits that the particle itself belongs to its neighborhood (i.e. $k \in \mathscr{N}_k$ for all k), then $\mathscr{L}_k^{\tau}$ does not exclude personal information, since $\mathbf{n}_k^s$ can eventually be equal to $\mathbf{p}_k^s$ for some $s \leq \tau$. In addition, $\mathbf{n}_k^s \preceq \mathbf{p}_l^s$ for all $s \leq \tau$ and $l \in \mathscr{N}_k$. Considering these facts, the following result can be presented.

Result B.11. *For the SMABBPSO, the scale matrix of a particle at time* $\tau + 1$ *uses its accumulated learning until* τ*, without discarding any information so far. This means that*

$$\boldsymbol{\Sigma}_k^{\tau+1} = (1-\beta)^{\tau+1}\mathbf{I} + \beta\sum_{s=0}^{\tau}(1-\beta)^{\tau-s}\mathbf{n}_k^s\mathbf{n}_k^{s,\prime} \tag{B.67}$$

for all k *and* $\tau \geq 0$*. In addition,*

$$\mathbf{x}_k^{\tau+1}|\lambda = \mathbf{n}_k^{\tau} + \left[\frac{(1-\beta)^{\tau+1}}{\lambda}\right]^{1/2}\mathbf{I} + \left(\frac{\beta}{\lambda}\right)^{1/2}\sum_{s=0}^{\tau}(1-\beta)^{(\tau-s)/2}\mathbf{n}_k^s z_s \qquad \lambda \sim Ga\left(\frac{\nu}{2}, \frac{\nu}{2}\right) \quad (\nu > 0) \tag{B.68}$$

for all k*,* $\tau \geq 0$*, and* $z_0, \ldots, z_{\tau} \overset{iid}{\sim} N(0,1)$*. This result implies that each particle selects new positions in the search space using an adaptive distribution with heavy tails and considering its accumulated learning until the current iteration. The value of* β *determines the influence of the accumulated learning on the new position. This influence is strong when* β *is near to* 0 *and weak when* β *is near to* 1*, with no influence when* $\beta = 1$*.*

Result B.12. *Under the stagnation assumption and $\nu > 2$, it follows that*

$$\mathrm{E}(\mathbf{x}_k^\tau) = \mathbf{n} \qquad \textit{and} \qquad \lim_{\tau\to\infty} \mathrm{Cov}(\mathbf{x}_k^\tau) = \left(\frac{\nu}{\nu-2}\right)\beta(2-\beta)\mathbf{n}\mathbf{n}' \tag{B.69}$$

for all k. In addition, SMABBPSO is order-2 stable, if $\nu > 2$ and $\beta \to 1$ when $\tau \to \infty$. Note that $[\nu/(\nu-2)]\mathbf{n}\mathbf{n}'$ is a fixed point for $\mathrm{Cov}(\mathbf{x}_k^\tau)$. *The pseudo code of SMABBPSO for UOPs is presented in Algorithm 15.*

Algorithm 15 SMABBPSO for UOPs.

Input: $D, K, \mathcal{N}, \nu, \beta$, and f

$\tau \leftarrow 0$
for $k \in \{1,\ldots,K\}$ **do**
  Initialize $\mathbf{x}_k$ and set $\mathbf{\Sigma}_k = \mathbf{I}$ (the identity matrix)
  $\mathbf{p}_k \leftarrow \mathbf{x}_k$
end for
for $k \in \{1,\ldots,K\}$ **do**
  $\mathbf{n}_k \leftarrow \mathrm{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
end for
repeat
  $\tau \leftarrow \tau + 1$
  for $k \in \{1,\ldots,K\}$ **do**
    $\lambda \sim \mathrm{Ga}(\frac{\nu}{2}, \frac{\nu}{2})$
    $\mathbf{z} \sim (N(0,1), \overset{iid}{\ldots}, N(0,1))'$
    $\boldsymbol{\mu}_k \leftarrow \mathbf{n}_k$
    $\mathbf{\Sigma}_k \leftarrow (1-\beta)\mathbf{\Sigma}_k + \beta\mathbf{n}_k\mathbf{n}_k'$
    $\mathbf{x}_k \leftarrow \boldsymbol{\mu}_k + \lambda^{-1/2}\sqrt{\mathbf{\Sigma}_k}\mathbf{z}$
    $\mathbf{p}_k \leftarrow \mathrm{BEST}(\mathbf{x}_k, \mathbf{p}_k)$
  end for
  for $k \in \{1,\ldots,K\}$ **do**
    $\mathbf{n}_k \leftarrow \mathrm{BEST}(\mathbf{p}_l : l \in \mathcal{N}_k)$
  end for
until some termination condition is met
$\mathbf{g} \leftarrow \mathrm{BEST}(\mathbf{n}_k : k = 1,\ldots,K)$
return $\mathbf{x}^* = \mathbf{g}$ and $f^* = f(\mathbf{g})$.

B.8.2 Experimental setup

An experimental analysis was conducted to investigate empirically the performance of the SMABBPSO to solve UOPs. A suite of 15 benchmark problems was used in this experimental study. These problems along with their references are shown in Table B.1. According to their properties, these problems are divided into two classes: f_{01-04} are unimodal problems and f_{05-15} are multimodal problems with many local minima. These functions have been widely used in many works. Yao, Liu & Lin (1999); Liang et al. (2006); Suganthan et al. (2005), and Wang et al. (2013) provide a detailed description of each problem. All benchmark functions used in the experiments are to be minimized over a search space D-dimensional with $D = 30$.

For purposes of comparison, 9 algorithms were considered in this experimental study. These algorithms along with their parameters settings are listed as follows:

1. SMABBPSO($K, D, \mathcal{N}, \nu, \beta$) where $K = 30, D = 30, \nu = 1.0$, and $\mathcal{N}$ defined by a ring topology with $|\mathcal{N}_k| = 3$ for all k. Table B.2 shows the value of β for each benchmark function.

2. Clerc & Kennedy (2002): PSO($K, D, \mathcal{N}, \varphi_1, \varphi_2$) where $K = 30, D = 30, \varphi_1 = \varphi_2 = 2.05$, and $\mathcal{N}$ such as in SMABBPSO.

3. Kennedy (2003): BBPSO($K, D, \mathcal{N}$) where $K = 30, D = 30$, and $\mathcal{N}$ such as in SMABBPSO.

Table B.1: Benchmark problems used in the experiments.

f	Problem	References
f_{01}	Sphere function	Yao, Liu & Lin (1999); Liang et al. (2006); Wang et al. (2013)
f_{02}	Schwefel's problem 1.2	Yao, Liu & Lin (1999); Wang et al. (2013)
f_{03}	Rosenbrock's function	Yao, Liu & Lin (1999); Liang et al. (2006); Wang et al. (2013)
f_{04}	Schwefel's problem 2.22	Yao, Liu & Lin (1999); Wang et al. (2013)
f_{05}	Schwefel's problem 2.26	Yao, Liu & Lin (1999); Liang et al. (2006); Wang et al. (2013)
f_{06}	Rastrigin's function	Yao, Liu & Lin (1999); Liang et al. (2006); Wang et al. (2013)
f_{07}	Ackley's function	Yao, Liu & Lin (1999); Liang et al. (2006); Wang et al. (2013)
f_{08}	Griewank's function	Yao, Liu & Lin (1999); Liang et al. (2006); Wang et al. (2013)
f_{09}	Weierstrass' function	Liang et al. (2006)
f_{10}	Rotated Rastrigin's function	Liang et al. (2006)
f_{11}	Rotated Ackley's function	Liang et al. (2006)
f_{12}	Rotated Griewank's function	Liang et al. (2006)
f_{13}	Rotated Weierstrass' function	Liang et al. (2006)
f_{14}	Shifted Rastrigin's function	Suganthan et al. (2005)
f_{15}	Shifted rotated Griewank's function	Suganthan et al. (2005)

Table B.2: Values of β used by SMABBPSO for each benchmark function.

f	f_{14}, f_{15}	$f_{03}, f_{05}, f_{06}, f_{08}, f_{09}, f_{10}, f_{12}, f_{13}$	f_{07}, f_{11}	f_{02}, f_{04}	f_{01}
β	0.01	0.05	0.10	0.20	0.30

4. Blackwell (2012): BBPSOwJ($K, D, \mathscr{N}, \alpha, p_J$) where $K = 30, D = 30, \alpha = 0.7, p_J = 0.01$, and $\mathscr{N}$ such as in SMABBPSO.

5. Mendes, Kennedy & Neves (2004); Kennedy & Mendes (2006): FIPS($K, D, \mathscr{N}, \varphi$) where $K = 30, D = 30, \varphi = 4.1$, and $\mathscr{N}$ such as in SMABBPSO.

6. Liang et al. (2006): Comprehensive learning PSO (CLPSO).

7. Richer & Blackwell (2006): Lévy BBPSO($K, D, \mathscr{N}, \alpha$) where $K = 30, D = 30, \alpha = 1.4$, and $\mathscr{N}$ such as in SMABBPSO.

8. Beyer & Schwefel (2002): ES(μ/ρ,λ) where $\mu = 30, \rho = 7$, $\lambda = \rho\mu$, $\tau_0 = (\sqrt{2D})^{-1}$, and $\tau = (\sqrt{2\sqrt{D}})^{-1}$.

9. Beyer & Finck (2012): CMSAES(μ/ρ,λ) where μ, ρ, λ, and τ_0 such as in ES, and $\tau_c = 1 + D(D+1)/(2\mu)$.

B.8.3 Results

A set of experiments was carried out to compare empirically 9 algorithms (including SMABBPSO) on 15 benchmark problems. In each experiment, a pair problem/algorithm was considered and a sequence of empirical errors was obtained from 30 independent runs of 1500 iterations, where in each run the algorithm was applied to solve the problem. The empirical error is defined by

$$\text{Error} = |f(\mathbf{x}^*) - f(\bar{\mathbf{x}})| \tag{B.70}$$

where f is the benchmark function, $\mathbf{x}^*$ is the best solution obtained by the algorithm into a particular run, and $\bar{\mathbf{x}}$ is the global minimum of f. Summary statistics were calculated to each sequence of empirical errors. In each experiment, a nonuniform and asymmetric initialization that does not contain the global minimum was used.

Table B.3 shows the mean and the standard deviation of the errors obtained by each algorithm on 15 benchmark problems. The best result for each problem is shown in boldface. It is important to observe that for the specific case of the CLPSO algorithm, the mean and the standard deviation presented in Table B.3 are those available in the original paper (Liang et al., 2006), where the algorithm was proposed. All the other results in Table B.3 were derived from computational simulations, considering the benchmark functions listed in Table B.1 and the experimental setup described for each algorithm.

Table B.3: Comparisons between SMABBPSO, PSO, BBPSO, BBPSOwJ, FIPS, CLPSO, Lévy BBPSO, ES, and CMAES

-	Error mean (standard deviation)				
Problem	PSO	BBPSO	BBPSOwJ	FIPS	SMABBPSO
f_{01}	1.13e-10 (7.94e-11)	1.58e-06 (1.75e-06)	4.34e-03 (2.37e-02)	1.03e-04 (6.95e-05)	**2.42e-154** (2.71e-154)
f_{02}	1.41e+03 (7.93e+02)	5.11e+03 (1.90e+03)	2.14e+03 (2.62e+03)	1.44e+03 (1.32e+03)	**5.24e-153** (5.13e-153)
f_{03}	5.26e+01 (4.20e+01)	9.47e+01 (8.55e+01)	6.50e+01 (4.22e+01)	4.29e+01 (4.01e+01)	**2.87e+01** (1.37e-02)
f_{04}	4.04e-04 (1.33e-04)	9.67e-04 (4.44e-04)	4.25e-03 (8.31e-03)	2.67e-04 (1.08e-04)	**2.32e-84** (2.17e-84)
f_{05}	3.19e+03 (1.18e+02)	2.51e+03 (2.88e+02)	**3.81e+01** (5.79e+01)	3.36e+03 (2.48e+02)	1.61e+02 (4.14e+01)
f_{06}	8.36e+01 (1.98e+01)	8.28e+01 (1.83e+01)	1.11e+01 (3.45e+00)	7.78e+01 (1.63e+01)	**0.00e+00** (0.00e+00)
f_{07}	1.90e+01 (3.59e+00)	5.08e-01 (8.70e-01)	1.54e-01 (3.55e-01)	6.86e-01 (1.04e+00)	**2.22e-15** (1.81e-15)
f_{08}	2.10e-03 (4.16e-03)	5.23e-03 (7.96e-03)	3.05e-02 (2.84e-02)	5.53e-02 (8.15e-02)	**0.00e+00** (0.00e+00)
f_{09}	3.19e-03 (4.95e-04)	5.23e-03 (3.18e-03)	5.21e-01 (1.22e-01)	8.01e-02 (7.05e-02)	**4.80e-12** (2.39e-12)
f_{10}	1.60e+02 (4.31e+01)	1.88e+02 (3.44e+01)	1.61e+02 (3.71e+01)	1.10e+02 (4.61e+01)	**0.00e+00** (0.00e+00)
f_{11}	4.24e+00 (6.99e+00)	1.91e+00 (4.86e-01)	4.15e+00 (9.56e-01)	1.64e+00 (8.12e-01)	**1.98e-15** (1.79e-15)
f_{12}	4.20e-03 (6.84e-03)	5.69e-03 (7.01e-03)	1.97e-02 (1.88e-02)	6.99e-02 (9.03e-02)	**0.00e+00** (0.00e+00)
f_{13}	2.30e+01 (1.48e+00)	2.61e+01 (1.88e+00)	2.86e+01 (2.34e+00)	1.79e+01 (2.97e+00)	**7.28e-12** (2.16e-12)
f_{14}	6.70e+01 (1.54e+01)	6.11e+01 (1.45e+01)	**1.02e+01** (2.86e+00)	1.26e+02 (3.19e+01)	4.64e+01 (1.31e+01)
f_{15}	3.21e-02 (1.06e-01)	6.62e-03 (5.20e-03)	1.69e-02 (1.94e-02)	1.15e-01 (1.00e-01)	**5.19e-03** (7.63e-03)
$W/L/T$	15/0/0	15/0/0	13/2/0	15/0/0	-
Problem	Lévy BBPSO	CLPSO	ES	CMAES	SMABBPSO
f_{01}	2.23e-10 (4.59e-10)	4.46e-14 (1.73e-14)	2.49e-104 (7.62e-104)	**6.15e-156** (7.33e-156)	2.42e-154 (2.71e-154)
f_{02}	1.04e+00 (7.75e-01)	Not available	1.62e+03 (5.12e+02)	5.13e-147 (1.16e-146)	**5.24e-153** (5.13e-153)
f_{03}	6.01e+02 (2.29e+02)	2.10e+01 (2.98e+00)	1.98e+01 (9.76e+00)	**9.59e+00** (1.26e+00)	2.87e+01 (1.37e-02)
f_{04}	5.98e-02 (2.58e-02)	Not available	3.88e+00 (2.87e+00)	7.68e-70 (2.94e-70)	**2.32e-84** (2.17e-84)
f_{05}	1.46e+03 (1.78e+02)	**0.00e+00** (8.79e-13)	8.87e+03 (3.28e+02)	1.03e+04 (4.21e+02)	1.61e+02 (4.14e+01)
f_{06}	3.19e+01 (7.71e+00)	4.85e-10 (3.63e-10)	3.64e+01 (7.96e+00)	9.37e+00 (2.19e+00)	**0.00e+00** (0.00e+00)
f_{07}	1.27e+00 (5.43e-01)	**0.00e+00** (0.00e+00)	4.00e-15 (0.00e+00)	4.00e-15 (0.00e+00)	2.22e-15 (1.81e-15)
f_{08}	6.12e-01 (1.72e-01)	3.14e-10 (4.64e-10)	**0.00e+00** (0.00e+00)	**0.00e+00** (0.00e+00)	**0.00e+00** (0.00e+00)
f_{09}	4.93e-01 (1.01e-01)	3.45e-07 (1.94e-07)	**0.00e+00** (0.00e+00)	**0.00e+00** (0.00e+00)	4.80e-12 (2.39e-12)
f_{10}	1.80e+02 (1.05e+01)	3.46e+01 (4.59e+00)	9.63e+01 (1.96e+01)	8.16e+00 (2.34e+00)	**0.00e+00** (0.00e+00)
f_{11}	3.10e+00 (4.73e-01)	3.43e-04 (1.91e-04)	4.00e-15 (0.00e+00)	4.00e-15 (0.00e+00)	**1.98e-15** (1.79e-15)
f_{12}	6.57e-01 (1.33e-01)	7.04e-10 (1.25e-11)	**0.00e+00** (0.00e+00)	**0.00e+00** (0.00e+00)	**0.00e+00** (0.00e+00)
f_{13}	1.95e+01 (3.17e+00)	3.07e+00 (1.61e+00)	1.27e-01 (3.46e-01)	6.86e-04 (1.90e-03)	**7.28e-12** (2.16e-12)
f_{14}	3.82e+01 (6.18e+00)	Not available	1.21e+02 (2.34e+01)	**2.04e+01** (5.53e+00)	4.64e+01 (1.31e+01)
f_{15}	7.63e-01 (1.35e-01)	Not available	1.06e+00 (5.74e+00)	**4.93e-04** (1.88e-03)	5.19e-03 (7.63e-03)
$W/L/T$	14/1/0	8/3/0	11/2/2	8/5/2	-

A single-problem analysis was developed considering the sequence of results obtained from all runs of an algorithm as an optimizer for a given problem. The results of this analysis are summarized in Table B.4.

Table B.4: Wilcoxon signed-ranks test for pairwise comparisons of algorithms on each benchmark function

	Comparison: SMABBPSO versus						
Problem	PSO	BBPSO	BBPSOwJ	FIPS	Lévy BBPSO	ES	CMSAES
f_{01}	$+$	$+$	$+$	$+$	$+$	$+$	$-$
f_{02}	$+$	$+$	$+$	$+$	$+$	$+$	$+$
f_{03}	$+$	$+$	$+$	$=$	$+$	$-$	$-$
f_{04}	$+$	$+$	$+$	$+$	$+$	$+$	$+$
f_{05}	$+$	$+$	$-$	$+$	$+$	$+$	$+$
f_{06}	$+$	$+$	$+$	$+$	$+$	$+$	$+$
f_{07}	$+$	$+$	$+$	$+$	$+$	$+$	$+$
f_{08}	$+$	$+$	$+$	$+$	$+$	$=$	$=$
f_{09}	$+$	$+$	$+$	$+$	$+$	$-$	$-$
f_{10}	$+$	$+$	$+$	$+$	$+$	$+$	$+$
f_{11}	$+$	$+$	$+$	$+$	$+$	$+$	$+$
f_{12}	$+$	$+$	$+$	$+$	$+$	$=$	$=$
f_{13}	$+$	$+$	$+$	$+$	$+$	$+$	$+$
f_{14}	$+$	$+$	$-$	$+$	$-$	$+$	$-$
f_{15}	$+$	$=$	$+$	$+$	$+$	$+$	$-$

$(+)$: means that SMABBPSO is significantly better than its competitor algorithm.
$(=)$: means that SMABBPSO does not differ significantly from its competitor algorithm.
$(-)$: means that the competitor algorithm is significantly better than SMABBPSO.

The Wilcoxon signed-rank test (Demsar, 2006; Derrac et al., 2011) was used for pairwise comparisons of algorithms when both are applied to optimize a benchmark function. It was tested whether the results obtained by SMABBPSO are statistically different from those obtained by its competitor algorithm on each benchmark function. The symbol $(+)$ denotes that SMABBPSO is significantly better than its competitor algorithm. The symbol $(=)$ denotes that SMABBPSO does not differ significantly from its competitor algorithm. Finally, the symbol $(-)$ denotes that the competitor algorithm is significantly better than SMABBPSO, which means that the mean error of the SMABBPSO is greater than the mean error of its competitor. In all comparisons, a level of significance of 5% was considered.

A quick and easy test to compare the performance of two algorithms in a multi-problem analysis is to count the number of cases on which an algorithm is the winner. If two algorithms are equivalent (as assumed under the null hypotheses), then it is hoped that each algorithm wins on approximately $N/2$ out of N problems investigated. Since tied matches support the null hypothesis, they should not be discounted when applying this test, but equally divided between the two algorithms. If there is an odd number of tied matches, one of them should be ignored. This procedure is known as the Sign test (Demsar, 2006; Derrac et al., 2011). The Sign test can provide a first snapshot about the pairwise comparisons of algorithms on the set of benchmark problems.

Comparisons between SMABBPSO and its competitors are summarized in Table B.3 as $W/L/T$ counts, which means that SMABBPSO wins in W problems, loses in L problems, and ties in T problems. In our experimental framework, the critical value for the two-tailed sign test (with a significance level of 5%) for paired comparisons of algorithms is 12, since 15 benchmark problems were used. This means that an algorithm is significantly better than another if it performs better on at least 12 out of 15 problems. Our experimental results show that SMABBPSO is significantly better than PSO, BBPSO, BBPSOwJ, FIPS, Lévy BBPSO, and ES with a level of significance of 5%. However, based on the Sign test, there is no experimental evidence to assert that SMABBPSO presents a statistically significant improvement compared to CMSAES, that is, there is no significant difference between these algorithms.

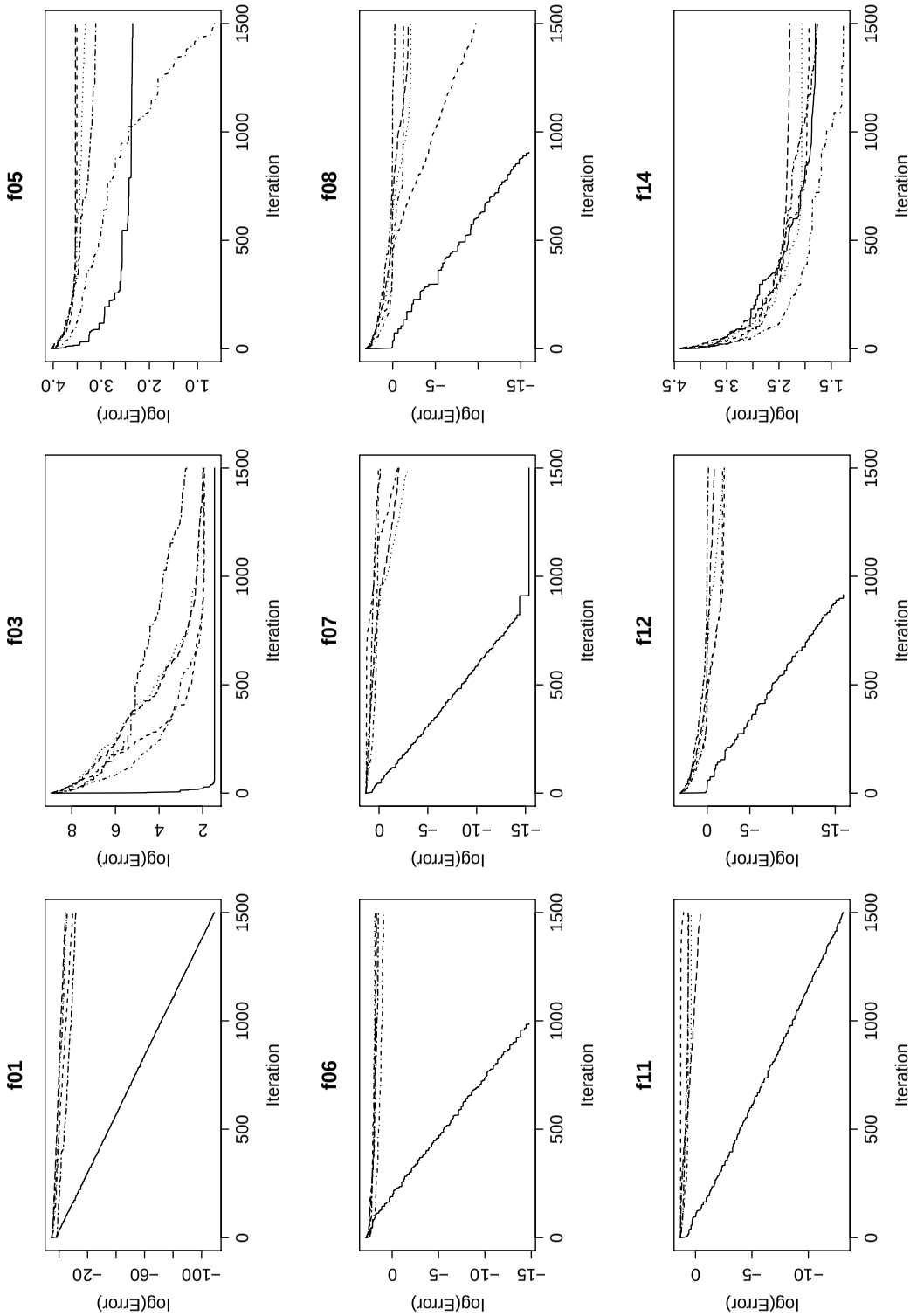

Figure B.8: Convergence plots for SMA-BBPSO (solid line), PSO (dashed line), BBPSO (dotted line), BBPSOwJ (dot-dash line), FIPS (long-dash line), and Lévy BBPSO (two-dash line) over nine benchmark functions.

Once again, a multi-problem analysis was developed using the Wilcoxon signed-rank test for pairwise comparisons of algorithms on the set of benchmark problems. The results of this analysis are summarized in Table B.5.

Table B.5: Wilcoxon signed-ranks test for pairwise comparisons of algorithms on the set of benchmark functions

Comparison	p-value	Conclusion
SMABBPSO $\times$ PSO	6.104E-06	SMABBPSO shows a significant improvement over PSO
SMABBPSO $\times$ BBPSO	6.104E-06	SMABBPSO shows a significant improvement over BBPSO
SMABBPSO $\times$ BBPSOwJ	0.04126	SMABBPSO shows a significant improvement over BBPSOwJ
SMABBPSO $\times$ FIPS	6.104E-06	SMABBPSO shows a significant improvement over FIPS
SMABBPSO $\times$ LBBPSO	0.002625	SMABBPSO shows a significant improvement over LBBPSO
SMABBPSO $\times$ ES	0.01709	SMABBPSO shows a significant improvement over ES
SMABBPSO $\times$ CMSAES	0.6355	there is no significant difference

The results of the Wilcoxon signed-rank test are in agreement with the results of the Sign test previously obtained. SMABBPSO shows a significant improvement over PSO, BBPSO, BBPSOwJ, FIPS, Lévy BBPSO, and ES with a level of significance of 5%. However, based on the Wilcoxon signed-rank test, there is no significant difference between SMABBPSO and CMSAES.

Fig. B.8 shows convergence plots for SMABBPSO, PSO, BBPSO, BBPSOwJ, FIPS, and Lévy BBPSO over nine benchmark functions: f_{01}, f_{03}, f_{05}, f_{06}, f_{07}, f_{08}, f_{11}, f_{12}, and f_{14}.

B.8.4 Discussion

A variant of the BBPSO was introduced and named as BBPSO with scale matrix adaptation (SMABBPSO). A theoretical analysis was presented to explain how the SMABBPSO works, emphasizing the differences between the proposed approach and those of other swarm algorithms. Each particle in SMABBPSO selects new positions in the search space using an adaptive distribution with heavy tails and considering its accumulated learning until the current iteration. This strategy induces exploration and exploitation at any stage of the search process.

Experimental results obtained from an empirical study revealed the suitability of the proposed approach in terms of effectiveness to find good solutions for all benchmark problems investigated. Non-parametric statistical tests for pairwise comparisons of algorithms on a set of benchmark functions indicated that SMABBPSO shows a statistically significant improvement compared with other swarm algorithms.

SMABBPSO is very flexible, allowing the inclusion of other distributions with heavy tails besides the t-distribution. From a suitable choice of the mixing density, a rich class of continuous, symmetric and unimodal distributions can be described. Furthermore, the particular case of the normal distribution can be recovered when $\lambda = 1$ almost surely or when $\nu \to \infty$ if the mixing density is given by $Ga(\frac{\nu}{2}, \frac{\nu}{2})$. Finally, several constraint-handling strategies can be combined with SMABBPSO for solving constrained optimization problems, with many potential applications in science and engineering.

B.9 Convergence analysis of the SMABBPSO algorithm

Result B.13. *Assume that the Eq.(B.67) is rewritten as*

$$\boldsymbol{\Sigma}_k^{\tau+1} = (1-\beta)^{\tau+1}\mathbf{I} + \beta \sum_{s=0}^{\tau-1} (1-\beta)^{\tau-s} \mathbf{n}_k^s \mathbf{n}_k^{s,\prime} + \eta \mathbf{n}_k^\tau \mathbf{n}_k^{\tau,\prime} \tag{B.71}$$

for all k and $\tau \geq 0$. When $\tau \to \infty$, if $\mathbf{g}_k^\tau \to \mathbf{g}^*$ and $\mathbf{n}_k^\tau \to \mathbf{g}^*$ with probability 1, $\beta \to 1$, and $\eta \to 0$, then $\mathbf{x}_k^\tau$ converges in mean square to $\mathbf{g}^*$ for all k. Thus, the swarm as a whole will converge to $\mathbf{g}^*$ when $\tau \to \infty$.

In order to empirically illustrate the Result B.13, simulations of the behaviour of a single particle in the SMABBPSO are shown in Fig. B.9. Selected samples from the distribution of $\mathbf{x}^\tau$ are presented for $\tau \geq 0$, considering $\nu = 2.1$, $\beta \to 1$, $\eta \to 0$, and $\mathbf{n}^0 = (8,9)'$, $\mathbf{n}^1 = (6,-3)'$, $\mathbf{n}^2 = (-2,4)'$, $\mathbf{n}^3 = (-1,3/2)'$, $\mathbf{n}^4 = (-2,3)'$, $\mathbf{n}^5 = (-1,-2)'$, $\mathbf{n}^6 = (-5/2,-3)'$, $\mathbf{n}^7 = (-3,-5/2)'$, and $\mathbf{n}^8 = (-3,-3)'$ (with $\mathbf{n}^{\tau+1} \prec \mathbf{n}^\tau$). Note that the focus of the search volume reaches the best solution $(-3,-3)'$ and the spread of the search volume does not expand without limit. Instead, the distribution of $\mathbf{x}^\tau$ converges to a degenerated distribution on $(-3,-3)'$.

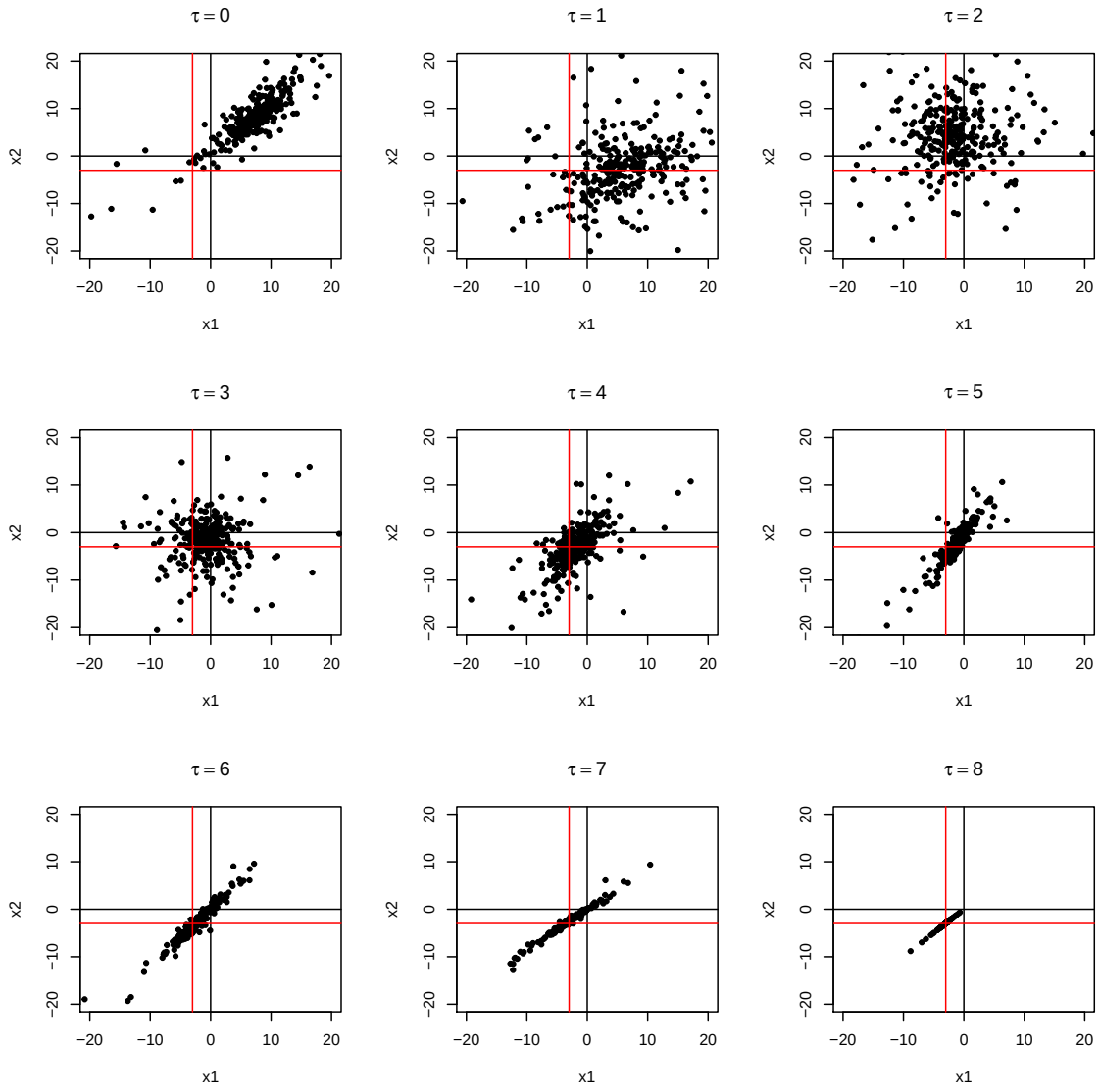

Figure B.9: Focus and spread of the search volume of a single particle in the SMABBPSO when τ increases.

Appendix C

Constraint Handling Methods in Swarm Computation

Particle swarm optimization (PSO) was originally proposed for unconstrained optimization problems (UOPs). However, many problems in science and engineering are described as optimization problems involving constraints, which the optimal solution of the problem must satisfy. This appendix provides a survey on constraint handling methods that have been adopted over the years to deal with constrained search spaces. Special attention is given on how these methods can be incorporated in the PSO for solving constrained optimization problems (COPs).

C.1 The penalty function method

Consider a COP such as defined in Eq. (1.3). In trying to solve this problem, the penalty function method has been the most standard approach because of its simplicity of implementation. This method converts a constrained problem in an unconstrained problem (or in a sequence of unconstrained problems). The constraints are placed into the objective function via a penalty function in a way that it penalizes any violation of the constraints. A suitable penalty function is usually of the form

$$p(\mathbf{x};\beta_1,\beta_2) = \sum_{i=1}^{I}[\max\{0,g_i(\mathbf{x})\}]^{\beta_1} + \sum_{j=1}^{J}|h_j(\mathbf{x})|^{\beta_2} \tag{C.1}$$

where β_1 and β_2 are normally 1 or 2 and the function

$$\phi(\mathbf{x}) = f(\mathbf{x}) + Cp(\mathbf{x};\beta_1,\beta_2) \tag{C.2}$$

is called of penalized objective function, where $C > 0$ is the penalty parameter. The purpose of this parameter is to make the penalty $p(\mathbf{x};\beta_1,\beta_2)$ of $\mathbf{x}$ of the same order of magnitude of its objective function value $f(\mathbf{x})$. A penalty function associates a positive penalty for infeasible solutions (i.e., $\mathbf{x} \in \mathbb{S} - \mathbb{F}$) and no penalty for feasible solutions (i.e., $\mathbf{x} \in \mathbb{F}$). This method can deal with inequality and equality constraints, but a common approach is to transform equality constraints in inequality constraints as follows:

$$g_j(\mathbf{x}) = |h_j(\mathbf{x})| - \delta \leq 0 \qquad j = 1,\ldots,J \tag{C.3}$$

where δ is a tolerance allowed ($\delta = 10^{-4}$ or 10^{-5}). This approach increases the number of inequality constraints to $M = I + J$ constraints and it establishes that the penalized objective function is written as

$$\phi(\mathbf{x}) = f(\mathbf{x}) + Cp(\mathbf{x};\beta) = f(\mathbf{x}) + C\sum_{m=1}^{M}\upsilon_m^{\beta}(\mathbf{x}) \tag{C.4}$$

where

$$\upsilon_m(\mathbf{x}) = \begin{cases} \max\{0,g_m(\mathbf{x})\} & m = 1,\ldots,I \\ \max\{0,|h_m(\mathbf{x})| - \delta\} & m = I+1,\ldots,M = I+J. \end{cases} \tag{C.5}$$

The penalty function method is a generic approach that can be applied to any COP. The most difficult aspect of this method is to find an appropriate penalty parameter needed to guide the search towards the constrained global optimum. Result C.1 justifies the use of the method to solve COPs.

Result C.1. *Consider a COP such as defined in Eq. (1.3). Suppose that the original constrained problem has an optimal solution and let $p(\mathbf{x};\beta)$ be a continuous penalty function ($p \in C^0$). Furthermore, suppose that for each $C > 0$ there exists a solution $\bar{\mathbf{x}}_C$ to the problem*

$$\text{minimize } \phi(\mathbf{x}) = f(\mathbf{x}) + Cp(\mathbf{x};\beta) \text{ subject to } \mathbf{x} \in \mathbb{S} \tag{C.6}$$

and suppose that the sequence $(\bar{\mathbf{x}}_C)_{C>0}$ is contained in a compact subset of the search space. Then,

$$\lim_{C\to\infty} \phi(\bar{\mathbf{x}}_C) = \inf\{f(\mathbf{x}) : \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x}) = \mathbf{0}, \mathbf{x} \in \mathbb{S}\}. \tag{C.7}$$

Furthermore, the limit of any convergent subsequence of $(\bar{\mathbf{x}}_C)_{C>0}$ is an optimal solution to the original constrained problem and $Cp(\bar{\mathbf{x}}_C;\beta) \to 0$ as $C \to \infty$.

Algorithm 16 The penalty function method for COPs.

Input: $f, \beta, c > 1, \varepsilon > 0$, and $\tau_{\max}$
1: $\tau \leftarrow 0$
2: Choose an initial point $\mathbf{x}^0$ and a penalty parameter $C > 0$
3: $\mathbf{x}^* \leftarrow \mathbf{x}^0$ and $\phi^* \leftarrow \phi(\mathbf{x}^0)$
4: $p^* \leftarrow p(\mathbf{x}^0;\beta)$
5: **repeat**
6: $\quad\tau \leftarrow \tau + 1$
7: $\quad C \leftarrow c \cdot C$
8: $\quad$Given $\mathbf{x}^0$ and C, find $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \phi(\mathbf{x};\beta,C)$ subject to $\mathbf{x} \in \mathbb{S}$
9: $\quad\mathbf{x}^0 \leftarrow \hat{\mathbf{x}}$ and $\phi_{\hat{\mathbf{x}}} \leftarrow \phi(\hat{\mathbf{x}})$
10: $\quad$**if** $\phi_{\hat{\mathbf{x}}} < \phi^*$ **then**
11: $\quad\quad\mathbf{x}^* \leftarrow \hat{\mathbf{x}}$ and $\phi^* \leftarrow \phi_{\hat{\mathbf{x}}}$
12: $\quad\quad p^* \leftarrow p(\hat{\mathbf{x}};\beta)$
13: $\quad$**end if**
14: **until** $C \cdot p^* < \varepsilon$ or $\tau = \tau_{\max}$
15: **return** $\mathbf{x}^*$ and ϕ^*.

Result C.1 informs that the optimal solution to the penalized objective function can be made arbitrarily close to the optimal solution of the original constrained problem by choosing C sufficiently large. However, if a very large C is selected and directly applied to solve the penalized problem, some computational difficulties may be encountered. When a large C is used, more emphasis is placed on feasibility and the procedure used to solve the penalized problem could quickly move toward a feasible solution, even being a feasible solution far from of the optimal solution, resulting in premature convergence. Due to this difficulty, a common approach is to use a sequence of increasing values for the penalty parameter as shown in Algorithm 16.

The performance of the penalty function method is not always satisfactory. Consequently, researchers have developed new strategies and some of these approaches are discussed in the following subsections. For all these approaches, assume that the total number of inequality constraints is $M = I + J$, including transformed equality constraints in inequality constraints. Finally, note that swarm algorithms can be used to solve COPs transformed into UOPs by using the penalty function method and all penalty approaches discussed here can be easily combined with these algorithms.

C.1.1 Static penalty

Homaifar, Lai & Qi (1994) proposed an approach that uses different penalty parameters for different levels of violation of each constraint. The penalized objective function is written as

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \sum_{m=1}^{M} C_{lm}[\max\{0, g_m(\mathbf{x})\}]^{\beta} \qquad l = 1,\ldots,L \tag{C.8}$$

where l denotes one of the L levels of violation defined for the mth constraint. At least in principle, this approach allows greater control of the penalization process by allowing to define a set of penalty parameters for each constraint of the problem. The main drawback of this approach is the high number of parameters required to be hand-tuned by the user. In fact, it requires $(2L-1)M$ parameters since

$$C_{lm} = \begin{cases} C_{1m} & \text{if } \upsilon_m^{\beta}(\mathbf{x}) \in (0, V_{2m}) \\ C_{2m} & \text{if } \upsilon_m^{\beta}(\mathbf{x}) \in [V_{2m}, V_{3m}) \\ \vdots & \\ C_{Lm} & \text{if } \upsilon_m^{\beta}(\mathbf{x}) \in [V_{Lm}, \infty) \end{cases} \tag{C.9}$$

where the V_{lm} values are user-defined constraint thresholds.

Hoffmeister & Sprave (1996) proposed another static method with the following penalized objective function:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \sqrt{\sum_{m=1}^{M} H(g_m(\mathbf{x}))(g_m(\mathbf{x}))^2} = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \text{ is feasible} \\ f(\mathbf{x}) + |g_m(\mathbf{x})| & \text{otherwise} \end{cases} \tag{C.10}$$

where $H : \mathbb{R} \to \mathbb{R}$ is the Heaviside function defined as

$$H(y) = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{if } y \leq 0. \end{cases} \tag{C.11}$$

The weakness of this method is that it is based on the assumption that infeasible solutions will always be evaluated worse than feasible ones, and this is not always the case.

Morales & Quezada (1998) proposed a static penalty method with the following penalized objective function:

$$\phi(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \text{ is feasible} \\ K\left(1 - \frac{\bar{u}(\mathbf{x})}{M}\right) & \text{otherwise} \end{cases} \tag{C.12}$$

where K is a large constant ($K \approx 10^9$), M is the total number of constraints, and $\bar{u}(\mathbf{x}) = |\{m : g_m(\mathbf{x}) \leq 0\}|$ is the number of constraints that are satisfied by $\mathbf{x}$. The user-defined constant K needs to be large enough to guarantee that $\phi(\mathbf{x}) \leq \phi(\mathbf{y})$ for all $\mathbf{x} \in \mathbb{F}$ and $\mathbf{y} \notin \mathbb{F}$. The main drawback of this method is related with the fact that when a solution is infeasible, its penalized objective function value is not computed and all solutions that violate the same number of constraints receive the same penalty regardless of how close these solutions are from the feasible region. This characteristic may reduce the swarm diversity, which seriously limits the application of this method combined with PSO to solve COPs, especially in highly constrained search space.

C.1.2 Dynamic penalty

These methods use penalty parameters which depend of the iteration counter in such a way that the parameters increase during the optimization process. Joines & Houck (1994) proposed a dynamic penalty method in which the penalty parameter varies during the search according to a pre-defined schedule. In this case, the penalty parameter increases with the iteration counter τ, i.e. $C(\tau) = k\tau$ (for $k > 0$), and the penalized objective function is written as

$$\phi(\mathbf{x}) = f(\mathbf{x}) + (k\tau)^{\alpha} \sum_{m=1}^{M} [\max\{0, g_m(\mathbf{x})\}]^{\beta} \tag{C.13}$$

where k, α, and β are constant defined by the user (the authors used $k = 0.5$, $\alpha = 1$ or 2, and $\beta = 1$ or 2). Joines & Houck (1994) also experimented the following penalized objective function:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \exp\left\{ (k\tau)^{\alpha} \sum_{m=1}^{M} [\max\{0, g_m(\mathbf{x})\}]^{\beta} \right\} \tag{C.14}$$

with $k = 0.05$ and $\alpha = \beta = 1$. The quality of the solution found by this method was very sensitive to changes in the values of α and β and there were no clear guidelines regarding the sensitivity of this approach to different values of C. In general, the problem associated with static penalty methods is also present with dynamic penalty methods. If a bad schedule is chosen to increase the penalty parameter, the search may converge to either non-optimal feasible solutions (if the penalty is too high) or to infeasible solutions (if the penalty is too low).

C.1.3 Adaptive penalty

These methods modify the penalty parameter according to the feedback taken from the optimization process. Hadj-Alouane & Bean (1997) developed a method that uses a penalized objective function

$$\phi(\mathbf{x}) = f(\mathbf{x}) + C(\tau) \sum_{m=1}^{M} [\max\{0, g_m(\mathbf{x})\}]^{\beta} \tag{C.15}$$

with the penalty parameter $C(\tau)$ adapted at each iteration by the following rule:

$$C(\tau+1) = \begin{cases} C(\tau)/\gamma_1 & \text{if case 1} \\ \gamma_2 C(\tau) & \text{if case 2} \\ C(\tau) & \text{otherwise} \end{cases} \tag{C.16}$$

where γ_1 and γ_2 are constants satisfying $\gamma_1 > \gamma_2 > 1$ and cases 1 and 2 denote situations where the best solution in the last k iterations was always feasible (case 1) or was never feasible (case 2). The update rule implies that if the optimization process is searching mainly in feasible regions, the penalty is decreased to increase the exploration in infeasible regions and, conversely, if the search is being conducted insufficiently in infeasible regions, the penalty is increased to attract solutions more towards feasible regions. The main drawback of this method is the definitions of the parameters γ_1, γ_2, k, and $C(0)$.

Lemonge & Barbosa (2004) introduced an adaptive penalty method without any type of user-defined penalty parameter. This method uses information from the population such as the average of the objective function values and the level of violation of each constraint during the optimization process. The penalized objective function is written as

$$\phi(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \text{ is feasible} \\ \bar{f}(\mathbf{x}) + \sum_{m=1}^{M} C_m \max\{0, g_m(\mathbf{x})\} & \text{otherwise} \end{cases} \tag{C.17}$$

where

$$\bar{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } f(\mathbf{x}) > \langle f \rangle \\ \langle f \rangle & \text{otherwise} \end{cases} \tag{C.18}$$

and

$$C_m = \frac{\langle f \rangle \langle \max\{0, g_m\} \rangle}{\sum_{m=1}^{M} \langle \max\{0, g_m\} \rangle^2} \tag{C.19}$$

with $\langle f \rangle$ representing the average of the objective function values (weighted over the current population) and $\langle \max\{0, g_m\} \rangle$ the average of the level of violation of the m-th constraint (also weighted over the current population). The main idea is that the values of the penalty parameters should be distributed in a way that those constraints which are more difficult to be satisfied should have a relatively higher penalty parameter.

C.2 Superiority of feasible solutions

Deb (2000) introduced a constraint-handling method to be implemented in genetic algorithms, which uses a tournament selection operator where two solutions are compared by using the following criteria (Deb's rules):

- Any feasible solution is preferred to any infeasible solution.
- Among two feasible solutions, the one having better objective function value is preferred.
- Among two infeasible solutions, the one having smaller constraint violation value is preferred.

Note that, penalty parameters are not needed because in any of the three criteria mentioned above, solutions are never compared in terms of both objective function and constraint violation information. In the first criterion, neither objective function value nor the constraint violation information are used, simply the feasible solution is preferred. In the second criterion, solutions are compared exclusively in terms of objective function values. Finally, in the third criterion, solutions are compared exclusively in terms of the constraint violation information.

This method can also be seen as a member of the class of penalty methods with the following penalized objective function:

$$\phi(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \text{ is feasible} \\ f_{\text{worst}} + \sum_{m=1}^{M} \max\{0, g_m(\mathbf{x})\} & \text{otherwise} \end{cases} \tag{C.20}$$

where f_{worst} is the objective function value of the worst feasible solution in the population. Thus, the penalized objective value of an infeasible solution not only depends on the amount of constraint violation, but also on the population of solutions at hand. However, the penalized objective value of a feasible solution is always equal to its original objective value.

Using the penalized objective function ϕ, this approach can be combined with PSO to solve COPs. In fact:

- Any feasible solution is preferred to any infeasible solution, since

$$\phi(\mathbf{x}) = f(\mathbf{x}) < f_{\text{worst}} + \sum_{m=1}^{M} \max\{0, g_m(\mathbf{y})\} = \phi(\mathbf{y}) \tag{C.21}$$

  for two solutions $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{x}$ is feasible and $\mathbf{y}$ is infeasible.

- Among two feasible solutions, the one having better objective function value is preferred, since

$$\phi(\mathbf{x}) = f(\mathbf{x}) < f(\mathbf{y}) = \phi(\mathbf{y}) \tag{C.22}$$

  for two feasible solutions $\mathbf{x}$ and $\mathbf{y}$ with $f(\mathbf{x}) < f(\mathbf{y})$.

- Among two infeasible solutions, the one having smaller constraint violation value is preferred, since

$$\phi(\mathbf{x}) = f_{\text{worst}} + \sum_{m=1}^{M} \max\{0, g_m(\mathbf{x})\} < f_{\text{worst}} + \sum_{m=1}^{M} \max\{0, g_m(\mathbf{y})\} = \phi(\mathbf{y}) \tag{C.23}$$

  for two infeasible solutions $\mathbf{x}$ and $\mathbf{y}$ with $\sum_{m=1}^{M} \max\{0, g_m(\mathbf{x})\} < \sum_{m=1}^{M} \max\{0, g_m(\mathbf{y})\}$. Alternatively, the Deb's rules can be implemented directly in any swarm algorithm through a classical bubble-sort procedure to update the local-best and global-best positions. This procedure is presented as a pseudo-code in Algorithm 17.

Algorithm 17 Deb's rule to rank candidate solutions of a population $\mathbf{P}'$.

Input: $\mathbf{P}' = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ $(k \leq K)$

```
for i ∈ (k − 1) : 1 do
  for j ∈ 1 : i do
    if s(x_j) = s(x_{j+1}) = 0 then
      {Comment: where s(x) = Σ_m max{0, g_m(x)}, m = 1, ..., M}
      if f(x_j) > f(x_{j+1}) then
        swap(x_j, x_{j+1})
      end if
    else
      if s(x_j) > s(x_{j+1}) then
        swap(x_j, x_{j+1})
      end if
    end if
  end for
end for
return x_(k) ≺ x_(k−1) ≺ ··· ≺ x_(2) ≺ x_(1) since φ(x_(k)) ≥ φ(x_(k−1)) ≥ ··· ≥ φ(x_(2)) ≥ φ(x_(1)).
```

The proposed approach by Deb is a constraint handling method widely used in evolutionary computation to solve COPs. However, it seems to have problems to maintain diversity in the population, and the use of niching methods combined with high mutation rates is apparently necessary to avoid stagnation.

C.3 Ranking methods

The main idea of penalty methods to solve COPs is to modify the objective value of a candidate solution with a penalty value based on a measure of its degree of constraint violation and, thus, promote the selection of feasible

solutions and, at the same time, also allow some infeasible solutions in the population with low degree of constraint violation. To achieve this goal, penalty methods require a careful fine-tuning of their penalty parameters to determine the severity of the penalties. However, it is difficult to strike the proper balance between objective and penalty functions using penalty parameters, which are highly problem-dependent. This section presents some alternative approaches based on ranking methods, which can be combined with PSO to deal with optimization problems having constrained search spaces.

C.3.1 Stochastic ranking

Runarsson & Yao (2000) developed a constraint-handling method that uses a random procedure to rank candidate solutions. This method is called stochastic ranking and it applies an user-defined probability to choose the criterion used for comparison of solutions, in order to determine which one is the best. That is, given a pair of solutions in the search space, the probability of comparing them according to the objective function is 1, if both solutions are feasible. Otherwise (i.e., if at least one solution is infeasible), the probability of comparing them according to the objective function is P and, finally, the probability of comparing them according to the sum of constraint violation is $1-P$.

The comparison of solutions is defined as an order relation on the set of values associated with $(f(\mathbf{x}), s(\mathbf{x}) = \sum_{m=1}^{M} \upsilon_m^2(\mathbf{x}))$. Let $f_1(f_2)$ and $s_1(s_2)$ be the objective value and the constraint violation, respectively, at a candidate solution $\mathbf{x}_1(\mathbf{x}_2)$. Then, for any P satisfying $0 \leq P \leq 1$, the comparison of $\mathbf{x}_1$ and $\mathbf{x}_2$ is defined as follows:

$$\mathbf{x}_1 \prec \mathbf{x}_2 \text{ ($\mathbf{x}_1$ is better than $\mathbf{x}_2$) when } \begin{cases} f_1 < f_2 & \text{if } (s_1 \text{ and } s_2 = 0) \\ f_1 < f_2 & \text{if } (s_1 > 0 \text{ or } s_2 > 0) \text{ and } r < P \\ s_1 < s_2 & \text{if } (s_1 > 0 \text{ or } s_2 > 0) \text{ and } r \geq P \end{cases} \tag{C.24}$$

where $r \sim \text{Unif}(0,1)$. Stochastic ranking uses a bubble-sort-like procedure to rank solutions of a population. This procedure is presented as a pseudo-code in Algorithm 18) and it can be implemented in any swarm algorithm to update the local-best and global-best positions.

Algorithm 18 Stochastic ranking to rank candidate solutions of a population $\mathbf{P}'$.

Input: $\mathbf{P}' = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ ($k \leq K$) and $0 \leq P \leq 1$

```
for i ∈ (k − 1) : 1 do
    for j ∈ 1 : i do
        r ∼ Unif(0, 1)
        if s(x_j) = s(x_{j+1}) = 0 or r < P then
            if f(x_j) > f(x_{j+1}) then
                swap(x_j, x_{j+1})
            end if
        else
            if s(x_j) > s(x_{j+1}) then
                swap(x_j, x_{j+1})
            end if
        end if
    end for
end for
return x_(k) ≺ x_(k−1) ≺ ··· ≺ x_(2) ≺ x_(1).
```

C.3.2 α-constrained method

Takahama & Sakai (2005) introduced the α-constrained method to deal with constraints in optimization problems. This method is based on a function that describes the satisfaction level of the constraints of a candidate solution (i.e., how well a candidate solution satisfies the constraints of the problem). This function is defined as

$$\mu(\mathbf{x}) = \min_{i,j}\{\mu_{g_i}(\mathbf{x}), \mu_{h_j}(\mathbf{x})\} = \min\{\min_{i}\{\mu_{g_i}(\mathbf{x})\}, \min_{j}\{\mu_{h_j}(\mathbf{x})\}\} \tag{C.25}$$

where

$$\mu_{g_i}(\mathbf{x}) = \begin{cases} 1 & \text{if } g_i(\mathbf{x}) \leq 0 \\ 1 - \frac{g_i(\mathbf{x})}{b_i} & \text{if } 0 < g_i(\mathbf{x}) \leq b_i \\ 0 & \text{otherwise} \end{cases} \tag{C.26}$$

is the satisfaction level of the constraint $g_i(\mathbf{x}) \leq 0$ with a fixed tolerance $b_i > 0$ and

$$\mu_{h_j}(\mathbf{x}) = \begin{cases} 1 - \frac{|h_j(\mathbf{x})|}{b_j} & \text{if } 0 \leq |h_j(\mathbf{x})| \leq b_j \\ 0 & \text{otherwise} \end{cases} \tag{C.27}$$

is the satisfaction level of the constraint $h_j(\mathbf{x}) = 0$ with a fixed tolerance $b_j > 0$. Note that the satisfaction level of the constraints satisfies

$$\begin{cases} \mu(\mathbf{x}) = 1 & \text{if } g_i(\mathbf{x}) \leq 0 \text{ and } h_j(\mathbf{x}) = 0 \text{ for all } i, j \\ 0 \leq \mu(\mathbf{x}) < 1 & \text{otherwise.} \end{cases} \tag{C.28}$$

Alternatively, it is possible to define the satisfaction level of the constraints as

$$\mu(\mathbf{x}) = \begin{cases} 1 - \frac{p(\mathbf{x};\beta_1,\beta_2)}{B} & \text{if } 0 \leq p(\mathbf{x};\beta_1,\beta_2) \leq B \\ 0 & \text{otherwise} \end{cases} \tag{C.29}$$

where $p(\mathbf{x};\beta_1,\beta_2)$ is the well-known penalty function (see Eq. (C.1)) and B is a positive fixed tolerance.

After the definition of μ, the α-level comparison of solutions is defined as an order relation on the set of values associated with $(f(\mathbf{x}),\mu(\mathbf{x}))$ in which $\mu(\mathbf{x})$ precedes $f(\mathbf{x})$ (if $\mu(\mathbf{x}) < \alpha$), because the feasibility of $\mathbf{x}$ is more important than the minimization of $f(\mathbf{x})$. Let $f_1(f_2)$ and $\mu_1(\mu_2)$ be the objective value and the satisfaction level, respectively, at a candidate solution $\mathbf{x}_1(\mathbf{x}_2)$. Then, for any α satisfying $0 \leq \alpha \leq 1$, the α-level comparison between $\mathbf{x}_1$ and $\mathbf{x}_2$ is defined as follows:

$$\mathbf{x}_1 \prec \mathbf{x}_2 \text{ when } \begin{cases} f_1 < f_2 & \text{if } \mu_1, \mu_2 \geq \alpha \\ f_1 < f_2 & \text{if } \mu_1 = \mu_2 \\ \mu_1 > \mu_2 & \text{otherwise.} \end{cases} \tag{C.30}$$

Note that, if $\alpha = 0$, then the α-level comparison is equivalent to the ordinal comparison using solely the objective function (i.e., candidate solutions are ranked solely based on their objective values). If $\alpha = 1$, then the α-level comparison works like the Deb's rules (i.e., feasible solutions are ranked solely based on their objective values, infeasible solutions are ranked solely based on their satisfaction levels of the constraints, and feasible solutions are always ranked better than infeasible solutions). Finally, the tuning process for α, b_i, and b_j (or α and B) is the main concern in the α-constraint method. Takahama & Sakai (2005) used a dynamic strategy for the parameter α with $b_i = b_j = 1000$ to solve the all benchmark problems mentioned in Runarsson & Yao (2000) and the experimental results obtained by the α-constrained method were compared with the results obtained by stochastic ranking.

C.3.3 Addition of ranking terms

Ho & Shimizu (2007) developed a ranking method to deal with constraints in optimization problems. The method is based on a new objective function to be minimized that comprises three ranking terms focused on the objective function values, constraint violation values, and number of constraints violated. The multiple rankings objective function is defined as

$$\phi(\mathbf{x}) = \begin{cases} \text{rank}(s(\mathbf{x}), \mathbf{x} \in \mathbb{S}) + \text{rank}(u(\mathbf{x}), \mathbf{x} \in \mathbb{S}) \text{ if all solutions in } \mathbb{S} \text{ are infeasible} \\ \text{rank}(f(\mathbf{x}), \mathbf{x} \in \mathbb{S}) + \text{rank}(s(\mathbf{x}), \mathbf{x} \in \mathbb{S}) + \text{rank}(u(\mathbf{x}), \mathbf{x} \in \mathbb{S}) \text{ otherwise} \end{cases} \tag{C.31}$$

where $\mathbf{x}$ is a candidate solution, $f(\mathbf{x})$ is the objective function value of $\mathbf{x}$, $s(\mathbf{x}) = \sum_{m=1}^{M} \upsilon_m^2(\mathbf{x})$ is the constraint violation value of $\mathbf{x}$, and $u(\mathbf{x}) = |\{m : g_m(\mathbf{x}) > 0\}|$ is the number of constraints violated by $\mathbf{x}$. The comparison between two solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ is defined as follows:

$$\mathbf{x}_1 \prec \mathbf{x}_2 \text{ when } \phi(\mathbf{x}_1) < \phi(\mathbf{x}_2). \tag{C.32}$$

The method uses sum of ranks to balance the objective function versus the constraint violation degree. When all solutions in the search space are infeasible, the main goal is to seek the first feasible solution in the feasible space.

Algorithm 19 Sum of ranks to find the best solution of a population $\mathbf{P}'$.

Input: $\mathbf{P}' = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ $(k \leq K)$

for each solution in $\mathbf{P}'$ **do**
  Calculate the properties $f(\mathbf{x}), s(\mathbf{x})$, and $u(\mathbf{x})$
end for
Rank the solutions with respect to f, s, and u independently
if feasible solutions exist in $\mathbf{P}'$ **then**
  for each solution **do**
    $\phi(\mathbf{x}) \leftarrow \text{rank}(f(\mathbf{x}), \mathbf{x} \in \mathbb{S}) + \text{rank}(s(\mathbf{x}), \mathbf{x} \in \mathbb{S}) + \text{rank}(u(\mathbf{x}), \mathbf{x} \in \mathbb{S})$
  end for
else
  for each solution **do**
    $\phi(\mathbf{x}) \leftarrow \text{rank}(s(\mathbf{x}), \mathbf{x} \in \mathbb{S}) + \text{rank}(u(\mathbf{x}), \mathbf{x} \in \mathbb{S})$
  end for
end if
Sort the solutions according to ϕ
TheBestSolution $\leftarrow \arg\min\{\phi(\mathbf{x}) : \mathbf{x} \in \mathbf{P}'\}$
return TheBestSolution

In this case, the information from $f(\mathbf{x})$ becomes unimportant and, hence, only $\text{rank}(s(\mathbf{x}), \mathbf{x} \in \mathbb{S})$ and $\text{rank}(u(\mathbf{x}), \mathbf{x} \in \mathbb{S})$ are used. When feasible solutions exist in the search space, the method explores the search space in order to find an optimum solution. In this case, $\text{rank}(s(\mathbf{x}), \mathbf{x} \in \mathbb{S})$ and $\text{rank}(u(\mathbf{x}), \mathbf{x} \in \mathbb{S})$ serve as terms to penalize infeasible solutions. On the other hand, the term $\text{rank}\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{S}\}$ allows to retain for the next iteration only those infeasible solutions with small objective value and small constraint violation degree. This strategy is necessary to maintain the population diversity and exploration ability in infeasible regions of the search space. Algorithm 19 outlines the method proposed by Ho & Shimizu (2007).

Clearly ϕ attempts to integrate the information from the objective function and constraint violation for candidate solutions. However, it is important to note that ϕ does not require a penalty coefficient and this is the most important feature of the proposed strategy. By transforming the relevant numerical properties into ranking terms of the same order of magnitude, different terms can be added directly without invoking a penalty coefficient. Ho & Shimizu (2007) applied the proposed method to solve all benchmark problems mentioned in Runarsson & Yao (2000) and the results obtained by this method were compared with the results obtained by stochastic ranking.

C.3.4 TOPSIS-based ranking

TOPSIS is a method proposed by Hwang & Yoon (1981) for multi-criteria decision making problems. It evaluates the performances of the alternatives through the similarity with the best possible solution. According to this technique, the best alternative would be the one which is closest to the positive ideal solution and farthest from the negative ideal solution. The positive ideal solution is the one that minimizes the cost criteria and maximizes the benefit criteria. The negative ideal solution is the one that maximizes the cost criteria and minimizes the benefit criteria. In summary, the positive ideal solution is composed of all best attainable values of the criteria and the negative ideal solution consists of all the worst attainable values of the criteria. To make use of TOPSIS, the attribute values must be numeric and TOPSIS makes use of this information to provide a ranking for all the alternatives within a population of candidate solutions. For a broad survey about TOPSIS, the reader is referred to Behzadian et al. (2012).

Lai, Liu & Hwang (1994) extended TOPSIS to solve a multi objective decision making problem. The proposed approach reduces a k-dimensional objective space to a two-dimensional objective space by a first-order compromise procedure. Then, membership functions of fuzzy set theory are used to represent the satisfaction level for both criteria, obtaining a single-objective programming problem by using the $\max - \min$ operator for the second-order compromise operation.

Schneider & Krohling (2014) proposed a method to find multiple solutions of constrained nonlinear integer optimization problems. This approach transforms the constrained optimization problem into a bi-objective optimization problem. The transformed problem is solved by using a hybrid approach with differential evolution and tabu search as search engines, and TOPSIS combined with membership functions to deal with the constraints.

C.4 Multi-objective optimization concepts to handle constraints

Among the several approaches that have been proposed as alternatives to the use of penalty methods, there is a class of methods in which the constraints of the problem are handled as objective functions, i.e., a single-objective COP is cast as a multi-objective (unconstrained) optimization problem (MOOP). This section discusses the concept of Pareto optimality and how a COP can be viewed as a MOOP.

A MOOP is defined as follows:

$$\begin{array}{rll} \min & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})) & \text{where } \mathbf{x} = (x_1, \ldots, x_D)' \in \mathbb{R}^D \\ \text{subject to} & L_d \leq x_d \leq U_d & d = 1, \ldots, D \end{array} \tag{C.33}$$

where $\mathbf{x}$ is the vector of the D decision variables and $\mathbf{f}(\mathbf{x})$ is the vector of K functions to be minimized over the search space $\mathbb{S} = [\mathbf{L}, \mathbf{U}] = [L_1, U_1] \times \cdots \times [L_D, U_D]$. It is not possible to minimize a vector of functions in the typical sense of the word minimize. Having several objective functions, the concept of an optimum solution for the problem changes to the concept of trade-off solution (or compromise), i.e., a solution in which one must balance objectives that are in conflict and cannot be satisfied at the same time. In MOOPs the main goal is to find good trade-offs (or compromises) between the objectives rather than a single optimum solution as in single-objective optimization. The notion of trade-off solutions is normally referred to as Pareto optimality, which is defined based on following concepts (see Simon, 2013, chap. 20):

Pareto dominance. A solution $\mathbf{x}_1$ is said to dominate the solution $\mathbf{x}_2$ if the following two conditions hold: (1) $f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2)$ for all $k \in \{1, \ldots, K\}$, and (2) $f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2)$ for at least one $k \in \{1, \ldots, K\}$. That is, $\mathbf{x}_1$ is at least as good as $\mathbf{x}_2$ for all objective function values and $\mathbf{x}_1$ is better than $\mathbf{x}_2$ for at least one objective function value. The notion of Pareto dominance is denoted as $\mathbf{x}_1 \prec \mathbf{x}_2$ to indicates that $\mathbf{x}_1$ dominates $\mathbf{x}_2$.

Nondominated solution. A solution $\bar{\mathbf{x}}$ is said to be nondominated, if there is no solution $\mathbf{x}$ that dominates it.

Pareto optimality. A solution $\bar{\mathbf{x}}$ in the search space is Pareto optimal if it is a nondominated solution in the search space. That is, $\bar{\mathbf{x}}$ is Pareto optimal if and only if there is no solution $\mathbf{x}$ in the search space such that: (1) $f_k(\mathbf{x}) \leq f_k(\bar{\mathbf{x}})$ for all $k \in \{1, \ldots, K\}$, and (2) $f_k(\mathbf{x}) < f_k(\bar{\mathbf{x}})$ for at least one $k \in \{1, \ldots, K\}$.

Pareto set. The Pareto set is defined as the set of all $\bar{\mathbf{x}}$ in the search space that are Pareto optimal solutions. That is,

$$\text{Pareto set} = \{\bar{\mathbf{x}} \in \mathbb{S} : \nexists \mathbf{x} \in \mathbb{S}, \mathbf{x} \prec \bar{\mathbf{x}}\}. \tag{C.34}$$

Pareto front. The Pareto front is defined as

$$\text{Pareto front} = \{\mathbf{f}(\bar{\mathbf{x}}) : \bar{\mathbf{x}} \text{ belongs to Pareto set}\}. \tag{C.35}$$

The main idea of adopting multi-objective optimization concepts to handle constraints is to transform a single-objective COP in a MOOP by defining the first objective as the objective function of the original problem and defining the remaining objectives as the constraint functions. As a result, the original COP with M constraints is transformed in a MOOP with $K = M + 1$ objectives and algorithms designed for MOOPs can be used to solve COPs (see Mezura-Montes & Coello, 2011).

It is important to note that the opposite direction can also be followed. A MOOP with K objectives can be formulated as a COP by just keeping one of the objectives and restricting the rest of the objectives within user-specified values. This approach is known in the multi-objective optimization literature as the $\boldsymbol{\varepsilon}$-constraint method. The reformulated problem is as follows:

$$\begin{array}{rll} \min & f_j(\mathbf{x}) & \text{where } \mathbf{x} \in \mathbb{R}^D \\ \text{subject to} & f_k(\mathbf{x}) \leq \varepsilon_k & k = 1, \ldots, K \text{ and } k \neq j \\ & L_d \leq x_d \leq U_d & d = 1, \ldots, D \end{array} \tag{C.36}$$

where the parameter ε_k represents an upper bound of the value of f_k. The solution of the reformulated problem is Pareto optimal for any upper bound vector $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_{j-1}, \varepsilon_{j+1}, \ldots, \varepsilon_K)'$. The main advantage of this method is that it can be used for any problem independent of whether the objective space is convex or nonconvex. However, the solution to the reformulated problem largely depends on the chosen $\boldsymbol{\varepsilon}$-vector. In addition, as the number of objectives increases, there exist more elements in the $\boldsymbol{\varepsilon}$-vector, thereby requiring more information from the user.

Appendix D

Previous Results for the G24 set of DCOPs

This appendix presents the full set of results obtained by Nguyen (2010, p. 181) for 18 algorithms tested on 18 problems of the G24 benchmark set of DCOPs along with the results obtained by the algorithm GSARepair proposed by Pal et al. (2013). Fig. D.1 shows a graphical representation of these results that are presented in Tables D.1 and D.2.

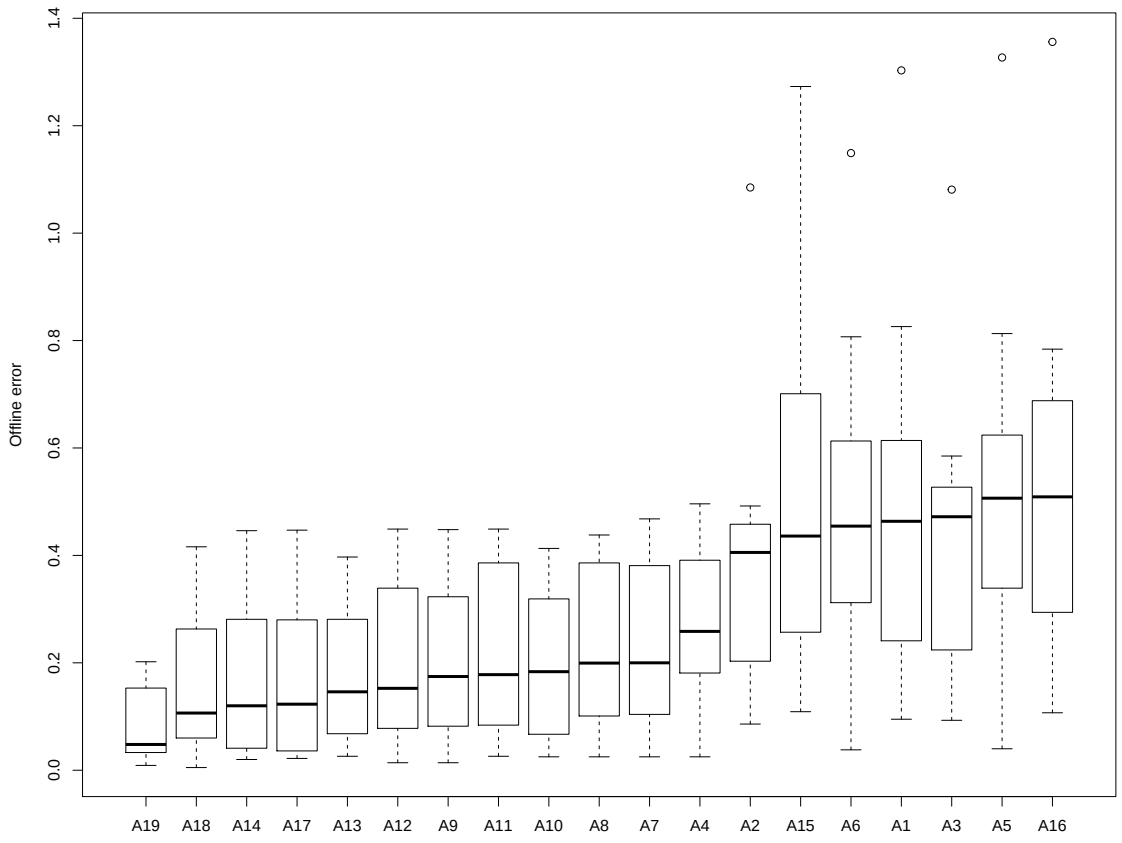

Figure D.1: Box-plots of offline errors for each tested algorithm.

Table D.1: Offline errors for different algorithms in the medium settings. Part I.

Probl. Alg.	G24-u Mean(SD)	G24-1 Mean(SD)	G24-f Mean(SD)	G24-uf Mean(SD)	G24-2 Mean(SD)	G24-2u Mean(SD)	G24-3 Mean(SD)	G24-3b Mean(SD)	G24-3f Mean(SD)
A1	0.214(0.037)	0.587(0.085)	0.227(0.065)	0.095(0.044)	0.329(0.074)	0.103(0.022)	0.384(0.092)	0.637(0.101)	0.241(0.051)
A2	0.131(0.034)	0.401(0.046)	0.171(0.100)	0.086(0.028)	0.283(0.021)	0.110(0.030)	0.340(0.045)	0.472(0.053)	0.203(0.042)
A3	0.173(0.042)	0.475(0.060)	0.224(0.052)	0.093(0.032)	0.376(0.055)	0.111(0.030)	0.561(0.104)	0.511(0.115)	0.142(0.058)
A4	0.468(0.059)	0.226(0.024)	0.041(0.011)	0.218(0.018)	0.281(0.036)	0.294(0.029)	0.156(0.008)	0.171(0.019)	0.025(0.008)
A5	0.602(0.211)	0.624(0.202)	0.238(0.159)	0.813(0.224)	0.497(0.066)	0.573(0.092)	0.239(0.128)	0.421(0.142)	0.040(0.047)
A6	0.577(0.082)	0.620(0.153)	0.238(0.137)	0.807(0.226)	0.455(0.109)	0.550(0.128)	0.200(0.098)	0.397(0.105)	0.038(0.008)
A7	0.306(0.084)	0.104(0.025)	0.041(0.009)	0.218(0.030)	0.202(0.027)	0.198(0.015)	0.038(0.007)	0.075(0.013)	0.025(0.004)
A8	0.362(0.063)	0.101(0.022)	0.042(0.011)	0.219(0.073)	0.198(0.030)	0.201(0.023)	0.034(0.005)	0.079(0.012)	0.025(0.002)
A9	0.254(0.048)	0.082(0.015)	0.028(0.006)	0.194(0.043)	0.162(0.021)	0.187(0.011)	0.029(0.004)	0.058(0.007)	0.014(0.002)
A10	0.319(0.034)	0.093(0.023)	0.045(0.010)	0.218(0.050)	0.171(0.026)	0.196(0.024)	0.027(0.005)	0.071(0.014)	0.025(0.005)
A11	0.156(0.018)	0.104(0.025)	0.041(0.010)	0.248(0.080)	0.196(0.035)	0.084(0.030)	0.035(0.007)	0.075(0.012)	0.026(0.004)
A12	0.152(0.026)	0.078(0.014)	0.029(0.009)	0.151(0.032)	0.171(0.021)	0.082(0.010)	0.029(0.005)	0.059(0.013)	0.014(0.002)
A13	0.175(0.040)	0.091(0.016)	0.043(0.010)	0.249(0.072)	0.161(0.029)	0.096(0.022)	0.026(0.005)	0.074(0.014)	0.027(0.006)
A14	0.120(0.028)	0.099(0.034)	0.020(0.008)	0.030(0.008)	0.177(0.031)	0.120(0.028)	0.099(0.034)	0.020(0.008)	0.030(0.008)
A15	0.302(0.101)	0.494(0.167)	0.109(0.072)	0.170(0.057)	0.701(0.100)	0.302(0.101)	0.494(0.167)	0.109(0.072)	0.170(0.057)
A16	0.412(0.195)	0.719(0.185)	0.107(0.083)	0.170(0.053)	0.638(0.178)	0.412(0.195)	0.719(0.185)	0.107(0.083)	0.170(0.053)
A17	0.123(0.029)	0.103(0.024)	0.022(0.014)	0.029(0.016)	0.138(0.030)	0.123(0.029)	0.103(0.024)	0.022(0.014)	0.029(0.016)
A18	0.091(0.035)	0.085(0.024)	0.021(0.014)	0.030(0.030)	0.099(0.028)	0.060(0.033)	0.028(0.007)	0.068(0.022)	0.005(0.003)
A19	0.049(0.004)	0.132(0.015)	0.029(0.012)	0.047(0.009)	0.182(0.019)	0.196(0.012)	0.028(0.004)	0.076(0.009)	0.009(0.007)
(i)	A1: GAelit			A7: GARepairwUPCwNRR			A13: dRepairHyperM-OOR		
	A2: RIGAelit			A8: dRepairGA			A14: Genocop		
	A3: HyperMelit			A9: dRepairRIGA			A15: GenocopwUPGwNRR		
	A4: GARepair			A10: dRepairHyperM			A16: GenocopwUPGwRR		
	A5: GARepairwUPGwNRR			A11: dRepairGA-OOR			A17: GenocopwUPCwNRR		
	A6: GARepairwUPGwRR			A12: dRepairRIGA-OOR			A18: dGenocop		
(ii)	Medium settings: change frequency $= 1000$, $\kappa = 1/2$, and $S = 20$						A19: GSARepair		

Table D.2: Offline errors for different algorithms in the medium settings. Part II.

Probl. Alg.	G24-4 Mean(SD)	G24-5 Mean(SD)	G24-6a Mean(SD)	G24-6b Mean(SD)	G24-6c Mean(SD)	G24-6d Mean(SD)	G24-7 Mean(SD)	G24-8a Mean(SD)	G24-8b Mean(SD)
A1	0.627(0.045)	0.373(0.031)	0.826(0.154)	0.571(0.071)	0.563(0.062)	0.614(0.108)	0.518(0.095)	0.409(0.027)	1.303(0.083)
A2	0.492(0.071)	0.259(0.031)	0.458(0.050)	0.426(0.045)	0.413(0.040)	0.427(0.028)	0.459(0.057)	0.410(0.019)	1.085(0.111)
A3	0.494(0.039)	0.297(0.047)	0.575(0.074)	0.469(0.074)	0.527(0.039)	0.585(0.057)	0.478(0.072)	0.406(0.046)	1.081(0.084)
A4	0.211(0.015)	0.236(0.024)	0.431(0.074)	0.427(0.048)	0.390(0.038)	0.354(0.038)	0.181(0.017)	0.496(0.032)	0.391(0.068)
A5	0.339(0.137)	0.286(0.059)	0.744(0.439)	0.708(0.185)	0.620(0.136)	0.516(0.208)	0.351(0.163)	0.448(0.100)	1.327(0.116)
A6	0.356(0.101)	0.281(0.084)	0.454(0.154)	0.656(0.151)	0.613(0.103)	0.588(0.173)	0.312(0.136)	0.415(0.092)	1.149(0.209)
A7	0.178(0.018)	0.181(0.023)	0.408(0.058)	0.381(0.048)	0.388(0.037)	0.341(0.029)	0.172(0.025)	0.468(0.053)	0.428(0.086)
A8	0.170(0.026)	0.181(0.032)	0.422(0.059)	0.393(0.038)	0.386(0.045)	0.356(0.037)	0.181(0.043)	0.438(0.031)	0.418(0.047)
A9	0.140(0.028)	0.152(0.017)	0.366(0.033)	0.346(0.028)	0.323(0.037)	0.315(0.029)	0.154(0.031)	0.448(0.020)	0.341(0.053)
A10	0.059(0.010)	0.131(0.019)	0.358(0.049)	0.341(0.039)	0.326(0.047)	0.286(0.035)	0.067(0.014)	0.413(0.032)	0.257(0.042)
A11	0.164(0.031)	0.177(0.034)	0.395(0.048)	0.391(0.045)	0.386(0.037)	0.352(0.035)	0.179(0.047)	0.422(0.037)	0.449(0.075)
A12	0.143(0.024)	0.154(0.028)	0.361(0.051)	0.352(0.035)	0.350(0.032)	0.302(0.022)	0.153(0.034)	0.449(0.017)	0.339(0.051)
A13	0.062(0.011)	0.131(0.019)	0.339(0.038)	0.342(0.040)	0.330(0.034)	0.281(0.036)	0.068(0.015)	0.397(0.038)	0.242(0.038)
A14	0.177(0.031)	0.059(0.039)	0.041(0.009)	0.407(0.073)	0.296(0.050)	0.281(0.050)	0.230(0.052)	0.408(0.043)	0.446(0.095)
A15	0.701(0.100)	0.378(0.121)	0.293(0.105)	0.809(0.175)	0.557(0.076)	0.725(0.397)	0.257(0.092)	0.682(0.141)	1.273(0.160)
A16	0.638(0.178)	0.440(0.279)	0.294(0.116)	0.688(0.187)	0.593(0.176)	0.578(0.189)	0.440(0.123)	0.784(0.093)	1.356(0.193)
A17	0.138(0.030)	0.074(0.025)	0.036(0.008)	0.319(0.074)	0.280(0.049)	0.291(0.061)	0.171(0.033)	0.427(0.039)	0.447(0.101)
A18	0.140(0.043)	0.114(0.025)	0.315(0.063)	0.334(0.085)	0.263(0.042)	0.242(0.041)	0.192(0.054)	0.415(0.039)	0.416(0.085)
A19	0.073(0.012)	0.153(0.013)	0.033(0.003)	0.047(0.003)	0.045(0.004)	0.037(0.007)	0.018(0.002)	0.202(0.041)	0.192(0.034)
(i)	A1: GAelit			A7: GARepairwUPCwNRR			A13: dRepairHyperM-OOR		
	A2: RIGAelit			A8: dRepairGA			A14: Genocop		
	A3: HyperMelit			A9: dRepairRIGA			A15: GenocopwUPGwNRR		
	A4: GARepair			A10: dRepairHyperM			A16: GenocopwUPGwRR		
	A5: GARepairwUPGwNRR			A11: dRepairGA-OOR			A17: GenocopwUPCwNRR		
	A6: GARepairwUPGwRR			A12: dRepairRIGA-OOR			A18: dGenocop		
(ii)	Medium settings: change frequency $= 1000$, $\kappa = 1/2$, and $S = 20$						A19: GSARepair		