UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

New Advances in Estimation of Distribution Algorithms for Quantum Machine Learning and Industrial Scenarios

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Vicente Pérez Soloviev
MSc Artificial Intelligence

Madrid, 2024


UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

Doctoral Degree in Artificial Intelligence

New Advances in Estimation of Distribution Algorithms for Quantum Machine Learning and Industrial Scenarios

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Vicente Pérez Soloviev
MSc Artificial Intelligence

Under the supervision of:
Dr. Concha Bielza
Dr. Pedro Larrañaga

Madrid, 2024

Title: New Advances in Estimation of Distribution Algorithms for Quantum Machine Learning and Industrial Scenarios

Author: Vicente Pérez Soloviev

Doctoral Programme: Artificial Intelligence

Thesis Supervision:

Dr. Concha Bielza, Full Professor, Universidad Politécnica de Madrid (Supervisor)

Dr. Pedro Larrañaga, Full Professor, Universidad Politécnica de Madrid

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

This thesis was partially supported by the Spanish Ministry of Science and Innovation through the PID2019-109247GB-I00, RTC2019-006871-7, PID2022-139977NB-I00 and TED2021-131310B-I00 projects, the BBVA Foundation (2019 Call) through the “Score-based non-stationary temporal Bayesian networks. Applications in climate and neuroscience" project. This thesis has also been also partially supported by Repsol through Agreement N.º6 UPM-Repsol “Self-driving lab project" and "Batch Reinforcement Learning" project, and by the Autonomous Community of Madrid within the ELLIS Unit Madrid framework Vicente P. Soloviev was supported by the predoctoral grant FPI PRE2020-094828 from the Spanish Ministry of Science and Innovation.

A Babushka y Dedushka

Acknowledgement

En primer lugar mi más sincero agradecimiento a mis directores de tesis que me han acompañado desde el trabajo de fin de máster hasta esta defensa de mi tesis doctoral. Pedro, Concha, muchas gracias por los sabios consejos y directrices en esta primera etapa de mi trayectoria como investigador.

Me gustaría también agradecer a aquellas personas que han colaborado conmigo durante estos años a través de convenios con empresa. En mi opinión es muy interesante este tipo de contacto entre la industria y la academia, en la que la transferencia tecnológica se ve reflejada en problemas reales de la sociedad. Gracias por tanto a Javi, Marco, Pedro, Marina y Gustavo por hacerme sentir parte de Repsol en nuestros proyectos en conjunto. Gracias también a Pelayo, David y Alessandro por una corta pero intensa colaboración entre la UPM e Idealista. Gracias Mayowa y Matthieu por otra corta pero intensa colaboración entre la academia y Fujitsu Research.

Una mención especial también para el grupo entero en la Universidad de Leiden que me acogió como un miembro más de la familia |aQa> por cuatro intensos y lluviosos meses en Países Bajos. En concreto a Vedran Dunjko por dirigir mi proyecto durante mi estancia en el grupo, y a Onur, Patrick, Yash, Adrian y Xavi por las muy interesantes charlas sobre computación cuántica.

Hacer una tesis es un proceso largo e intenso, pero gracias a todos los compañeros del laboratorio con los que he compartido risas, charlas, cabreos, viajes e incluso fiestas, estos años han pasado hasta rápido. Muchas gracias a Alaia, Bojan, Carlos Li, Carlos Villa, Dani, David Atienza, David Quesada, Enrique, Esteban, Fernando, Gabriel, Irene, Jorge, Kevin, Laura, Marta, Rafa, Sebastiaan y Víctor.

Otra mención especial a Andrés Gómez por enseñarme tantas cosas sobre computación cuántica que desconocía cuando empecé a estudiar y dedicarme tanto tiempo. Gracias por enseñarme y espero seguir en contacto contigo. También me gustaría agrader en general al CESGA, CITIUS y al *IBM Quantum Researchers Program* por el acceso a los superordenadores para llevar a cabo parte de la experimentación de esta tesis doctoral.

Me gustaría agradecer también a aquellas personas que me facilitaron la llegada a Madrid. Son así mis excompañeros de piso, Carmen, Blas y Carlos, que sin ellos la adaptación a las tierras madrileñas habría sido menos amena.

Los últimos meses de esta tesis han sido compatibilizados con mi trabajo como investigador en Fujitsu Research Europa. Ellos me han facilitado poder paralelizar la última etapa de la tesis con mis labores en la empresa. Me gustaría agradecer a todas aquellas personas en el equipo que de una forma u otra lo han hecho posible.

Me gustaría destacar a mi pareja Carla en estos agradecimientos por apoyarme día a día durante estos años. Por esos días en los que he estado frustrado, pero también por aquellos en los que había algo que celebrar. Independientemente, siempre me has apoyado y has estado a mi lado.

Por último, pero no menos importante, un enorme agradecimiento a mi padre, por sus sabios consejos como gran investigador que es; a mi madre y a mi hermana, por sus enormes apoyos durante estos años; y a mis abuelos, tíos, primos, y amigos que siempre han estado orgullosos de mi.

Abstract

Evolutionary algorithms span a wide range of domains, including Industry 4.0, machine learning, and quantum machine learning (QML) due to their ease of implementation, scalability, and interpretability. This PhD dissertation focuses on estimation of distribution algorithms (EDAs), which use probabilistic models (PMs) to sample new solutions. Bayesian networks (BNs) graphically represent conditional dependencies between variables, and their embedding into EDAs can reveal patterns and enhance solution quality.

In this PhD thesis we first focus on the application of these approaches in industrial problems. First, we extend an EDA with Gaussian BNs to optimize solvent mixtures for cost-effective dissolution in a laboratory. The EDA explores an initial search space provided by the experts in which several constraints are required (expert knowledge) and includes a parameter to balance exploration. Results indicate that maximizing exploration replicates expert proposals, while decreasing leads to more cost-effective solutions. This methodology was later extended to a multi-objective (MO) problem in designing experiments for fuel fabrication in a laboratory. To reduce resource use, we developed a framework based on EDAs to propose experimental formulas, where some objectives cannot be evaluated analytically and are predicted with surrogate models. A posteriori analysis revealed previously unknown dependencies between fuel components.

Traditional EDAs for continuous domains usually involve assuming Gaussians. Moreover, the EDA baseline updates a PM based on the top individuals from the previous iteration, risking local optima convergence. We introduce a semiparametric EDA, relaxing the Gaussianity assumption by using semiparametric BNs, where kernel estimated and Gaussian nodes coexist, determined dynamically by the algorithm. Our approach incorporates information from multiple past iterations to learn the PM. Empirical results show statistically significant improvements over other methods.

EDAs share similarities with some QML algorithms. This dissertation secondly combines EDAs with quantum computing (QC) for various optimization tasks.

On the one hand, quantum-inspired methods lead to classical algorithms assisted by QC. We introduce a quantum-inspired EDA, replacing the PM with a quantum circuit. Results indicate that quantum noise improves the results. We also compare computational costs across different topologies and identify the ideal one for our EDA.

On the other hand, variational quantum algorithms (VQAs) are QC methodologies assisted by classical optimizers. VQAs aim to minimize a given energy function by optimizing the parameters of the quantum parametric circuit (*ansatz*). Our first contribution in this field applies VQAs for BN structure learning. We analyze the algorithm's noise resilience across different types of noises, showing competitive results. Due to current quantum device limitations, we employed the Digital Annealer for larger BNs.

We propose the use of EDAs for the parameter optimization of the *ansatz* by benchmarking different EDAs in different scenarios. Our results show that EDAs offer competitive results compared to other gradient-based / -free approaches in terms of CPU time and accuracy, with

and without quantum noise simulation. Quantum architecture search involves optimizing not only the parameters but also the architecture of the *ansatz*. To manage this complex multi-level optimization, we use an EDA assisted by a surrogate model to rank solutions by expected energy. We also address both accuracy and barren plateau occurrence by framing it as a MO problem. Results show that our approach, effectively enhances the trainability of known architectures while maintaining strong performance.

Finally, we introduce `EDAspy`, a Python open-source library designed to advance EDA research. Noting the lack of EDA implementations, we offer over ten variants and supports creating custom EDAs with various components.

Resumen

Los algoritmos evolutivos abarcan muchos dominios, como la Industria 4.0, aprendizaje automático y automático cuántico (QML) debido a su fácil implementación, escalabilidad e interpretabilidad. Esta tesis se centra en los algoritmos de estimación de distribución (EDA), que usan modelos probabilísticos (PM) para generar nuevas soluciones. Las redes bayesianas (BN) representan dependencias condicionales entre variables, y su integración en los EDA puede revelar patrones y mejorar las soluciones.

En esta tesis nos centramos en la aplicación de los EDAs a problemas industriales. Primero, extendemos un EDA con BNs Gaussianas para optimizar mezclas de disolventes en una disolución en laboratorio. El EDA explora un espacio de búsqueda inicial con restricciones (conocimiento experto) proporcionado por los expertos, e incluye un parámetro que equilibra la exploración. Los resultados indican que maximizar la exploración imita a los expertos, y reducirla conduce a mayor rentabilidad. Esta metodología se amplió después para abordar un problema multiobjetivo (MO) en el diseño de experimentos para la fabricación de combustible. Para reducir el uso de recursos desarrollamos una herramienta basada en EDAs para proponer fórmulas experimentales, en las que algunos objetivos no pueden evaluarse analíticamente y se predicen con modelos subrogados. El análisis a posteriori reveló patrones desconocidos anteriormente entre los componentes del combustible.

Los EDAs tradicionales para dominios continuos asumen una distribución paramétrica. Además, el EDA canónico actualiza un PM basándose en los mejores individuos de la iteración anterior, con riesgo de convergencia prematura. Introducimos un EDA semiparamétrico, relajando la asunción de gaussianidad usando BNs semiparamétricas, donde nodos estimados por kernels coexisten con nodos gaussianos, determinados dinámicamente por el EDA. Nuestra propuesta incorpora información de múltiples iteraciones pasadas para aprender el PM. Se muestran mejoras estadísticamente significativas respecto a otros métodos.

El EDA se asemeja a algunos algoritmos de QML. Esta tesis combina EDAs con computación cuántica (QC) para optimización. Por un lado, los métodos cuántico-inspirados son algoritmos clásicos asistidos por QC. Introducimos un EDA cuántico-inspirado, sustituyendo el PM por un circuito cuántico. Los resultados indican que el ruido cuántico mejora los resultados. También comparamos el coste computacional de distintas topologías e identificamos la ideal para nuestro EDA.

Por otro lado, los algoritmos cuánticos variacionales (VQA) son metodologías asistidas por optimizadores clásicos. Los VQAs minimizan una función de energía tuneando los parámetros del circuito paramétrico (*ansatz*). Nuestra primera contribución al campo aplica VQAs para aprender BNs. Analizamos la resiliencia al ruido para diferentes tipos, encontrado resultados competitivos. El Digital Annealer es usado para BNs grandes por las limitaciones actuales de hardware.

Proponemos el uso de EDAs para la optimización de parámetros del *ansatz* comparando diferentes EDAs y escenarios. Los resultados muestran que los EDAs son competitivos comparado con otras metodologías, basadas o no en gradientes, en términos de CPU y calidad

de soluciones, con/sin ruido cuántico. Buscar la arquitectura cuántica implica optimizar la arquitectura y parámetros del *ansatz*. Para gestionar esta optimización multinivel, utilizamos un EDA asistido por un modelo subrogado que ordena las soluciones por energía esperada. También abordamos la calidad de la solución y aparición de la meseta barren como un problema MO. Mostramos que nuestro EDA mejora la capacidad de entrenamiento de arquitecturas manteniendo alto rendimiento.

Por último, se presenta `EDAspy`, una librería en Python, gratuita y de código abierto, diseñada para avanzar en la investigación de EDAs. Dada la falta de implementaciones se ofrecen más de diez variantes y permiten crear EDAs personalizados por componentes.

Contents

List of Figures

List of Tables

Abbreviations and acronyms

The most relevant abbreviations used during this PhD thesis are listed below:

AIC	Akaike's Information criterion
BP	Barren plateaus
BN	Bayesian network
BIC	Bayesian information criterion
BD	Bayesian Dirichlet
BNSL	Bayesian network structure learning
CPD	Conditional probability distribution
CD	Crowding distance
CKDE	Conditional kernel density estimation
CMA-ES	Covariance matrix evolutionary strategy
CVaR	Conditional value at a risk
DAG	Directed acyclic graph
DE	Differential evolution
DoE	Design of experiments
DM	Diversity metric
EA	Evolutionary algorithm
EDA	Estimation of distribution algorithm
EGNA	Estimation of Gaussian network algorithm
EMNA	Estimation of multivariate normal algorithm
ES	Evolution strategy
GBN	Gaussian Bayesian network
GA	Genetic algorithm
HV	Hypervolume

IC Information content

KDE Kernel density function

KDEBN Kernel density estimation-based Bayesian network

LHS Latin hypercube sampling

MAE Mean absolute error

MLE Maximum likelihood estimation

MMHC Max-min hill climbing

MSE Mean squared error

PDF Probability density function

PLS Probabilistic logic sampling

PSO Particle swarm optimization

QAOA Quantum approximate optimization algorithm

QC Quantum computing

RMSE Root mean squared error

SPBN Semiparametric Bayesian network

UMDA$_C$ Univariate marginal distribution algorithm for continuous domain

u_KEDA Univariate kernel density estimation of distribution algorithm

m_KEDA Multivariate kernel density estimation of distribution algorithm

UPM Universidad Politécnica de Madrid

VQA Variational quantum algorithm

VQE Variational quantum eigensolver

Notation

The following most relevant notation is used along the PhD thesis and is formalized as:

n	Number of variables in a model
d	Dimension of the optimization problem
N	Population-size in evolutionary algorithms
l	Archive-length
$\boldsymbol{X} = (X_1, \ldots, X_n)$	Set of variables
$\boldsymbol{x} = (x_1, \ldots, x_n)$	Instance of the variables set $\boldsymbol{X}$
$\boldsymbol{G_i}$	Set of solutions corresponding to the generation i
A^t	Archive at iteration t
$\boldsymbol{G}_i^S$	Subset of solutions selected from $\boldsymbol{G_i}$ according to some criteria
$p(\boldsymbol{X})$	Probability distribution of $\boldsymbol{X}$
$f(\boldsymbol{X})$	Density function of $\boldsymbol{X}$
$\mathcal{L}(\cdot)$	Log-likelihood function
$p(\boldsymbol{X}\|\boldsymbol{X})$	Probability distribution of $\boldsymbol{X}$ given $\boldsymbol{Y}$
$\mathcal{N}$	Normal distribution
$g(\cdot)$	Cost function
$\boldsymbol{x}^*$	Optimal solution found for an optimization problem
$\boldsymbol{x}^*$	Global optima of an optimization problem
$\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$	Vector of means
$\boldsymbol{\Sigma}$	Covariance matrix
σ	Standard deviation
$\lvert \cdot \rvert$	Cardinality of a set
$\boldsymbol{H}$	Hamiltonian
$\lvert \cdot \rangle$	Quantum state represented with Ket notation

$|+\rangle$ Superposition state

H Hamiltonian

p Number of layers of an *ansatz*

$\epsilon(\psi)$ Quantum channel applied to a quantum state

$\boldsymbol{\Theta}$ BN parameters

$\boldsymbol{\Theta}^*$ Optimal parameters of a BN model

$\mathcal{G}$ DAG representation of the BN structure

$\mathcal{D}$ Dataset with $\{\boldsymbol{X_1}, \ldots, \boldsymbol{X_n}\}$ instances, and n variables

$\mathcal{O}$ Set of operators in the BN learning algorithm

Part I

INTRODUCTION

Chapter 1

Introduction

Evolutionary algorithms (EAs) [Dasgupta and Michalewicz, 2014] mimic the principles of natural evolution to solve complex optimization and search problems. Rooted in the mechanisms of natural selection and genetics, EAs employ a population-based approach, iteratively evolving solutions to optimize a given objective function. In the context of machine learning, EAs provide powerful tools for hyper parameter optimization [Alibrahim and Ludwig, 2021], feature selection [Wang et al., 2015], and neural network architecture search [Lu et al., 2019], among others [Larrañaga and Bielza, 2024]. Unlike traditional optimization techniques that may become trapped in local optima, EAs have shown to maintain a diverse population of potential solutions, enabling more robust exploration of the search space.

Traditional EAs: genetic algorithms (GAs) ([Holland, 1975]), evolutionary strategies ([Rechenberg, 1971]), evolutionary programming ([Fogel et al., 1966]) or genetic programming ([Cramer, 1985]). The main disadvantage found in traditional EAs is that they do not explicitly consider dependencies among the variables involved, and hence, new sampled solutions are not able to exploit the information found in the data. This limitation is overcome by using a probabilistic graphical models [Koller and Friedman, 2009] to reproduce new solutions. This type of EAs are the estimation of distribution algorithms (EDAs) [Larrañaga and Lozano, 2001].

Industrial optimization tasks are often characterized by (i) the large amount of decision variables, (ii) the uncertainty of the inter-dependencies between the variables, (iii) the need of fast computation due to the existence of dynamic scenarios, (iv) the necessity of interpretability of the implemented models, and (v) the need of a set of optimal solutions, among others. In this sense, we state that EDAs make a perfect match for this necessity, because of the following facts:

- EDAs are able to efficiently tackle large scale optimization tasks [Hong et al., 2021].
- Complex probabilistic models, such as Bayesian networks (BNs) [Koller and Friedman, 2009], can be embedded into the EDAs framework which are able to detect conditional dependencies between the decision variables of the problem [Larrañaga et al., 2012].
- EDAs maintain a population of potential solutions, allowing simultaneous exploration of multiple areas of the search space. Moreover, the evaluation of these solutions can be

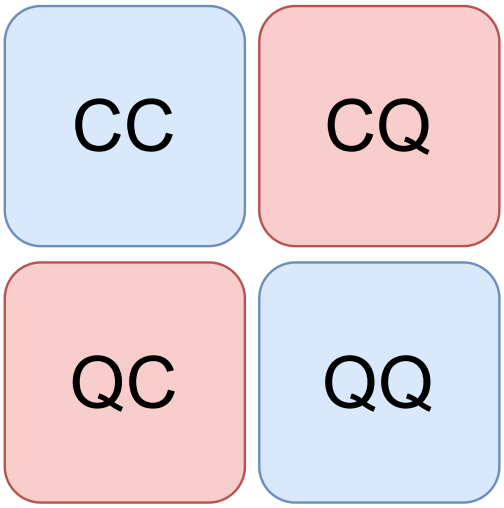


Figure 1.1: Context of the PhD thesis in the topics of classical and quantum computation. CC, CQ, QC, QQ refer to classical computation, classical computation assisted by quantum computing, quantum computing assisted by classical computing, and quantum computing, respectively.

easily parallelized to speed the computation [Falcón-Cardona et al., 2021].

- The ease of implementation of the EDAs approaches makes the framework to be easily understood. Moreover, the BNs learned in each iteration represent the dependencies encountered in several regions of the search space discovering uncertain knowledge that can be analyzed a posteriori by the industrial crew [Mihaljević et al., 2021].
- EDAs not only return the best solution found so far, but also a ranking with the best ones discovered during runtime [Larrañaga and Lozano, 2001].
- EDAs are usually resilient to noisy or imprecise cost function evaluations [Rakshit et al., 2017].
- EDAs have been successfully applied to a wide range of industrial applications [Larrañaga and Bielza, 2024].

EAs have also been combined with quantum computing (QC) technologies [Nielsen and Chuang, 2002, Gyongyosi and Imre, 2019] in the last decade. QC has demonstrated to be a way of saving energy consumption and outperforming classical computation in some specific problems, such as optimization or chemistry simulation. This is leading to the noisy intermediate-scale quantum (NISQ) era, which is characterized by being limited by the number of qubits of the devices and the presence of quantum noise in the systems. The intersection between QC and machine learning [Murphy, 2022] is quantum machine learning (QML).

Previous studies on QML [Schuld, 2018, Lau et al., 2017, Wiebe et al., 2012] have focused on exploiting the benefits of the QC to improve the performance of the optimization algorithms. With respect to EAs, QML revolves around three main research areas [Zhang, 2011]: (i) quantum-inspired evolutionary algorithms, which take advantage of the concepts and principles of quantum mechanics to improve the classic EAs; (ii) evolutionary-designed quantum

algorithms, which develop new quantum algorithms implemented by EAs; and (iii) quantum EAs, which develop new EAs to be executed in quantum devices.

In this thesis we address two general aspects of the topic of EDAs. On the methodological side, we aim to advance the state of the art of EDAs by proposing new approaches that improve its current state. On the application side, we aim to introduce the use of these approaches to real industrial problems, as well as in the field of QML. Figure 1.1 summarizes the main focus of the thesis, and will be systematically referred to during this document. It depicts the four ways of combining classical (C) and quantum (Q) computation. CC regards the new methods introduced to the state of the art of the EDAs and the application of the methods to industrial problems. CQ regards to EDAs assisted by QC, whereas QC regards to quantum algorithms assisted by EDAs. Finally, QQ regards to those contributions in the area of QC, in which EDAs are not present.

Chapter outline

The outline of the chapter is organized as follows. Section 1.1 enumerates the hypotheses and objectives of this thesis, and Section 1.2 explains the document organization.

1.1 Hypotheses and objectives

The hypotheses of this thesis are as follows:

- **H1**. Allowing the EDA to decide itself whether to use parametric, non-parametric or a combination of both probability distributions will improve traditional EDA approaches performance for continuous optimization domains.
- **H2**. The combination of classical with quantum computing applied for Bayesian network structure learning will achieve competitive results with the state of the art.
- **H3**. EDAs will provide competitive results for optimizing the architecture and parameters of quantum parametric circuits for variational quantum algorithms.
- **H4**. Quantum-inspired evolutionary algorithms will outperform classical evolutionary algorithms for solving complex optimization problems by harnessing principles of quantum mechanics.
- **H5**. EDAs will provide interpretable results and useful tools for facing single- and multi-objective real problems in the Industry 4.0.

To prove these hypotheses, the following objectives are proposed:

- **O1**. To develop an EDA variant in which parametric and non-parametric probability distributions can co-exist together for continuous optimization.
- **O2**. To develop a hybrid method between classical and quantum computations for Bayesian network structure learning, and prove its resilience to quantum noise.
- **O3**. To compare the performance of EDAs with the state of the art for parameter

optimization in quantum parametric circuits.

- **O4**. To develop an efficient method for optimizing the architecture of quantum parametric circuits.
- **O5**. To develop an EDA variant where new solutions are sampled from a quantum system.
- **O6**. To develop an open-source library in which several EDA approaches are available and custom implementations can be easily extended.
- **O7**. To solve a single-objective real problem in the Industry 4.0 using EDAs.
- **O8**. To solve a multi-objective real problem in the Industry 4.0 using EDAs.

1.2 Document organization

The thesis is organized in five parts that include fourteen chapters and three appendices, with the following contents:

Part I. Introduction

This part introduces the topic of the thesis and details its hypotheses and objectives.

- Chapter 1 enumerates the hypotheses and objectives of the work and describes the document outline.

Part II. Background

- Chapter 2 provides an introductory background of Bayesian networks. The chapter formally explains the representation of these models, and afterwards, the learning and inference methods.
- Chapter 3 explains the main differences of EDAs compared to the rest of EAs, and reviews the different research lines in the topic.
- Chapter 4 provides an introductory background for quantum computation. First the general foundations of quantum mechanics and its representation are explained. Then, variational quantum algorithms are introduced as a methodology widely used during this thesis. The chapter finishes with a soft introduction to quantum noises.

Part III. Contributions

This part contains the nine chapters associated to each of the contributions of the thesis.

- Chapter 5 assesses the performance of traditional EDAs to solve a real optimization problem in the industry. The algorithm searches over the landscape of initial solutions, where the user is able to tune how different are the new samples compared to those

provided initially by the experts, but also specifies the desired values for a subset of variables.

- Chapter 6 extends previous research done in Chapter 5 for multi-objective optimization applied to a real industrial optimization problem which aims to design an optimal formulation for fuel fabrication. Surrogate modelling is used for the approximation of part of the restrictions set.
- Chapter 7 overcomes a major limitation of traditional EDAs in continuous optimization proposing a novel methodology in which a coexistence of both parametric and non-parametric probability distributions is allowed. The algorithm decides itself in each iteration the type of distributions to use, and moreover, the algorithm uses the information retrieved from more than one past iterations.
- Chapter 8 introduces a new methodology in which a quantum system is proposed as the sampler engine of the EDA. The quantum system is iteratively updated based on the best solutions selected in the previous iteration for the traveling salesman problem.
- Chapter 9 faces the problem of learning the structure of a Bayesian network using different quantum approaches. On the one hand, variational quantum algorithms are used, where a noise resilience analysis is performed. On the other hand, a quantum-inspired device is used to learn larger problem instances.
- Chapter 10 proposes the usage of EDAs for the optimization of quantum parametric circuits, and compares different variants of the algorithm to the state-of-the art approaches.
- Chapter 11 presents a methodology for quantum architecture search. That is, going one step beyond Chapter 10 in which, not only the parameters are optimized, but also the quantum circuit composition. An EDA-based approach assisted by surrogate modelling faces the multi-objective optimization, where avoiding barren plateaus is one of the objectives while improving the performance of the variational quantum algorithm.
- Chapter 12 presents `EDAspy` library, where many EDA variants are implemented. Moreover, visualization tools and modules to easily implement a custom EDA version are also provided.

Part IV. Conclusions

This part rounds the thesis off with some conclusions and future work.

- Chapter 13 first summarizes the main conclusions agreed during this thesis, and then proposes new open research lines.

Part V. Appendices

Supplementary materials are included in this part.

- Appendix A describes the benchmark suites used for the validation of the methodology proposed in Chapter 7, and are publicly available in the library presented in Chapter 12.

- Appendix B describes the dataset provided by the experts from the industry for the developed methodology presented in Chapter 6.
- Appendix C shows the configuration used for the algorithms to which our approach is compared in Chapter 6.
- Appendix D shows the main results described as part of Chapter 9, in which large Bayesian networks are learned using quantum-inspired technologies.
- Appendix E shows the complementary materials for the results shown in Chapter 11, in which a novel approach is proposed for quantum architecture design.
- Appendix F presents the library required metadata for `EDAspy`, described in Chapter 12.

Part II

BACKGROUND

Chapter 2

Bayesian Networks

2.1 Introduction

Complex problems usually involve dealing with variables in which the relationships among them are unknown. Discovering the dependencies between each other is highly interesting for the experts in the case of industrial settings, but also in other fields.

Bayesian networks (BNs) [Koller and Friedman, 2009] are useful tools founded on probability theory for dealing with uncertainty in a given training dataset. Their representation allows the combination of automatic learning with the incorporation of expert knowledge added by the user. Once this model is trained, it allows to perform different types of reasoning for discovering relationships between the variables, and to be implemented for decision-making tasks.

Chapter outline

The outline of this chapter is organized as follows. Section 2.2 introduces the formal representation of BNs. Section 2.3 reviews the three main ways of learning the model from some given data. Section 2.4 introduces the concept of inference for BNs and how it is performed in continuous domains.

2.2 Representation

Bayesian networks (BNs) [Koller and Friedman, 2009] are a type of probabilistic graphical model that compactly represents the joint probability distribution of a set of random variables. BNs are widely used in machine learning [Murphy, 2022] for different applications [Bielza and Larrañaga, 2014, Puerto-Santana et al., 2021] due to their capability of representing the uncertain knowledge contained in the data and the possibility of adding expertise.

BNs are defined as a pair $(\mathcal{G}, \boldsymbol{\Theta})$ over a set of random variables $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$, where n is the number of variables. They are composed of: (i) a directed acyclic graph (DAG) $\mathcal{G} = \{V, A\}$, where V denotes the variables in $\boldsymbol{X}$ represented as nodes in the DAG

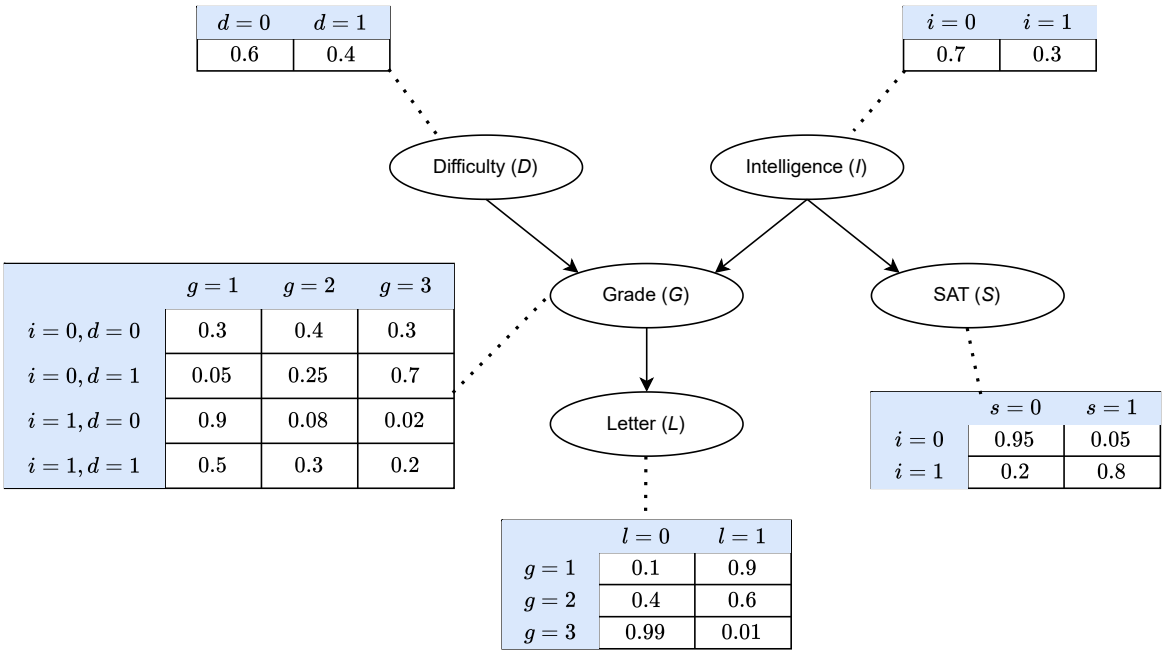


Figure 2.1: Student BN in which nodes are the variables of the problem, arcs represent conditional dependencies between the variables, and the tables are the CPDs.

and A contains the arcs between the nodes, which encode the conditional (in)dependence relationships among the variables; and (ii) a set of parameters $\boldsymbol{\Theta}$ that define the conditional probability distribution (CPD) of each variable X_i given its parents $\boldsymbol{Pa}_i$ in $\mathcal{G}$, where the parents of a variable X_i are the variables directly pointing at X_i in the DAG.

Considering this notation, the joint probability distribution $p(\boldsymbol{X})$ over a set of random variables $\boldsymbol{X}$ is obtained as a product of all the CPDs of each variable given its parents $(p(X_i|\boldsymbol{Pa}_i))$ in the graph:

$$p(\boldsymbol{X}) = p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|\boldsymbol{Pa}_i) \tag{2.1}$$

Regarding the parameters $\boldsymbol{\Theta} = (\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \ldots, \boldsymbol{\Theta}_n)$, each parameter $\boldsymbol{\Theta}_i$ defines the conditional probability of a variable X_i to its parents $\boldsymbol{Pa}_i$. If a variable does not have parents in the graph, then $\boldsymbol{\Theta}_i$ corresponds to its marginal probability $(p(X_i))$. On the other hand, if X_i has parents in the graph, then its parameters correspond to the effects of its parents on its CPD.

Figure 2.1 shows an example of a BN, in which five random variables are represented: the difficulty of an exam (D), the intelligence of a student (I), the grade that the student archives (G), the student's SAT score (S) and the quality of the recommendation letter (L). All the variables are binary $(0, 1)$ except G which can have three different values $(1, 2, 3)$. Arcs between the nodes represent conditional dependencies, and the tables represent the CPDs of the model.

Regarding the BN structure $(\mathcal{G})$, the grade of the student directly depends on both the

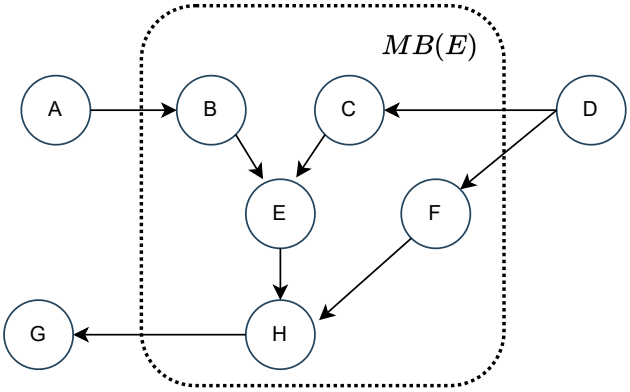


Figure 2.2: Illustrative example of the set of variables included in the Markov blanket of variable E, which includes the parents, children, and parents of children of E.

difficulty of the exam and the intelligence of the student. However, the student's SAT score only depends on the intelligence of the student. The quality of the recommendation letter only depends on the grade that the student achieves in the exam.

Regarding the BN parameters ($\boldsymbol{\Theta}$), the CPDs associated to D represent the marginal probability $p(D)$ of intelligent ($d = 0$) versus non intelligent ($d = 1$). Another example is the CPD associated to I. The distribution over the student's grade, SAT score S and recommendation letter L, are conditional probabilities $p(G|D, I)$, $p(S|I)$ and $p(L|G)$, respectively. For example, $p(G|D, I)$ represents the probability of the student's grade depending on the difficulty of the exam and the intelligence of the student. The CPD assigns a specific value for all possible variables configuration.

Following Equation 2.1 the joint probability distribution of the BN represented in Figure 2.1 is computed as

$$p(D, I, G, S, L) = p(D)p(I)p(G|D, I)p(S|I)p(L|G). \tag{2.2}$$

The Markov blanket $\mathbf{MB}(X_i)$ of a variable X_i in the BN is the minimal set of variables such that X_i is conditionally independent of all the other variables given $\mathbf{MB}(X_i)$. $\mathbf{MB}(X_i)$ is considered the strongly related variables to X_i when data follow a stable distribution [Tsamardinos and Aliferis, 2003], and includes parents, children and spouses (parents of children) of X_i found in $\mathcal{G}$. Figure 2.2 depicts an illustrative example of the Markov blanket of the variable E in a BN structure.

Depending on the nature of the variables in the BN, we can distinguish among discrete, continuous and hybrid BNs, which hold discrete, continuous and a combination of both types of variables, respectively.

2.2.1 Discrete BNs

When dealing with discrete variables, the CPD is a tabular representation, where $p(X_i|\boldsymbol{Pa}_i)$ is encoded as a table in which each entry is a joint assignment to X_i and each variable in $\boldsymbol{Pa}_i$. Note that each entry in the CPD table is nonnegative and is restricted to $\sum_{x_i} p(x_i|\boldsymbol{pa}_i) = 1 \ \ \forall i$, where $\boldsymbol{pa}_i$ denotes a value assignment for $\boldsymbol{Pa}_i$.

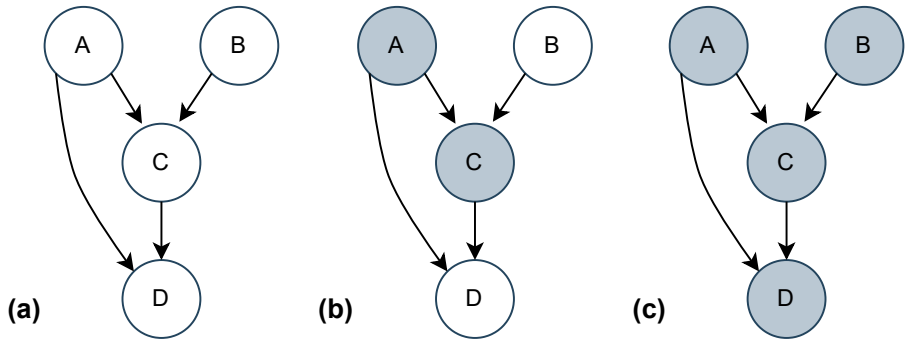


Figure 2.3: Parametric (a), semiparametric (b) and non-parametric (c) BN examples, where white nodes and grey nodes represent parametric and non-parametric conditional distributions, respectively.

2.2.2 Continuous BNs

When dealing with continuous BNs, a tabular representation is unfeasible, and thus, each CPD is factorized as a conditional probability density function (PDF). Then, Equation 2.1 is redefined as

$$f(\boldsymbol{X}) = f(X_1, \dots, X_n) = \prod_{i=1}^{n} f(X_i|\boldsymbol{Pa}_i). \tag{2.3}$$

Depending on the CPD used for the variables in the BN, we classify the BNs for continuous domains in parametric, non-parametric and semiparametric BNs. Figure 2.3 graphically represents the models, in which white and grey nodes represent the nodes in which a parametric and non-parametric CPD is used, respectively. On the one hand, parametric BNs (Figure 2.3(a)) assume a parametric probability distribution over all the variables in the model. On the other hand, non-parametric BNs (Figure 2.3(c)) overcome this assumption by using non-parametric models over the variables. Semiparametric BNs (Figure 2.3(b)) are a combination of both types of models in which parametric and non-parametric distributions are considered in the BN and dependencies between both types are allowed [Atienza et al., 2022b].

2.2.2.1 Parametric BNs

When a BN assumes a parametric distribution for each of its variables, such as Gaussian, then the BN is a Gaussian Bayesian network (GBN), where all CPDs are defined using a linear Gaussian CPD:

$$f(X_i|\boldsymbol{pa}_i) = \mathcal{N}(\beta_{i0} + \beta_{i1} pa_{i_1} + \cdots + \beta_{ik} pa_{i_k}; \sigma_i^2) \tag{2.4}$$

where $\beta_{i1}, \dots, \beta_{ik}$ are the weights associated to each of the k parents of X_i, σ_i^2 is the respective variance, and β_{i0} is the intercept coefficient.

GBNs are the most widely used models in the BN area when dealing with continuous data, as they provide many advantages, such as ease of implementation, the speed of fitting a dataset to a Gaussian distribution, and the existence of closed formulas for performing inference in such models.

However, when the data do not fit Gaussians, then the distribution is poorly modelled. Moreover, as GBNs assume linear interactions between X_i and $\boldsymbol{Pa}_i$, they are not applicable in representing nonlinear relationships among variables.

Other approaches related to the assumption of probabilistic parametric distributions include mixtures of multivariate Gaussian distributions [Thiesson et al., 1998], mixtures of Gaussians [Dasgupta, 1999], mixtures of truncated exponentials [Moral et al., 2001], mixtures of polynomials [Shenoy and West, 2011], and mixtures of truncated basis functions [Langseth et al., 2012]. However, because of factors such as the existence of closed formulas for inference and the fact that learning the parameters is less expensive, the usage of GBNs is more flexible and widespread than these alternatives.

2.2.2.2 Non-parametric BNs

The alternative to assuming a parametric distribution over a dataset is to use non-parametric models. Non-parametric models have the advantage of better fitting the distribution function that do not fit parametric models, but they sacrifice speed and simplicity.

An example of this type of non-parametric model is the kernel density estimation (KDE) [Silverman, 2018], which can be considered as a mixture model in which the number of components is equal to the number of data samples. Consequently, the KDE demands much more memory compared to parametric models, and this demand grows according to the size of the data used for its training. The KDE joint probability density is defined as:

$$f(\boldsymbol{x}) = \frac{1}{N}\sum_{j=1}^{N} K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{x}_j), \tag{2.5}$$

where $K(\cdot)$ is the n-variate kernel function, $K_{\boldsymbol{H}}(\boldsymbol{x}) = |\boldsymbol{H}|^{-1/2}K(\boldsymbol{H}^{-1/2}\boldsymbol{x})$ is the kernel function, $\boldsymbol{x}_j$ is the j-th sample among the N samples used to train the KDE and $\boldsymbol{H}$ is the bandwidth matrix: a square $n \times n$ matrix that defines properties of the KDE such as the smoothness of the density estimation. If a Gaussian kernel is used, then $K(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}} exp(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x})$, where $\boldsymbol{x}^T$ is the transpose vector of $\boldsymbol{x}$.

A BN where all the CPDs are estimated with KDEs is named a kernel density estimation-based Bayesian network (KDEBN). This type of model is less common than GBNs but is still widely used in the literature [Pérez et al., 2009, Wang et al., 2016] because of the goodness of fit provided by the KDEs used. However, KDEBNs inherit the disadvantages of KDEs, such as a high memory demand that grows linearly with N, the parameters to be tuned (such as $\boldsymbol{H}$), and the complexity of these models.

Other approaches that avoid parametric models include the combination of GBNs with non-parametric Bayesian mixture models [Ickstadt et al., 2010] and the use of Gaussian processes to learn functional relations between variables [Friedman and Nachman, 2000]. However, the use of KDEs is more widespread due to the existence of different types of easily implementable kernels.

2.2.2.3 Semiparametric BNs

The combination of both non-parametric and parametric CPDs results in semiparametric BNs (SPBNs) [Atienza et al., 2022b]. This type of model allows the coexistence of Gaussian and KDE nodes in the same models, and the connections among them. Figure 2.3(b) shows an example of an SPBN structure where all possible dependencies between both types of nodes are depicted. The seminal paper proposed different algorithms for learning the parameters and the structure of SPBNs. During the learning process these learning algorithms decide between the following two possibilities: (i) the dependency between a variable X_i and its parents $\boldsymbol{Pa}_i$ is linear Gaussian,

$$f(X_i|\boldsymbol{Pa}_i) = \mathcal{N}\left(\beta_{i0} + \sum_{X_j \in \boldsymbol{Pa}_i} \beta_{ij} X_j, \sigma_i^2\right)$$

where $\beta_{i1}, \ldots, \beta_{ik}$ are the weights associated with each of the k parents of X_i, β_{i0} is the intercept coefficient and σ_i^2 the variance of X_i; or (ii) the dependency is given by a conditional kernel density estimation (CKDE) CPD:

$$f(X_i|\boldsymbol{Pa}_i) = \frac{f(X_i, \boldsymbol{Pa}_i)}{f(\boldsymbol{Pa}_i)},$$

where $f(X_i, \boldsymbol{Pa}_i)$ and $f(\boldsymbol{Pa}_i)$ are KDE models as defined in Equation (2.5).

Note that if all the CPDs in the SPBN are linear Gaussian, then the learned SPBN is equivalent to a GBN, and if all the CPDs in the SPBN are CKDEs, the learned SPBN is equivalent to a KDEBN. For a more exhaustive mathematical formalization, see Atienza et al. [2022b].

2.2.3 Hybrid BNs

When considering both continuous and discrete variables in the same model, two types of conditional dependencies must be addressed: discrete variables are parents of continuous ones, and viceversa. In the first case, we can store a linear Gaussian CPD for each possible discrete parents configuration (Equation 2.4). In the second case, this approach cannot be used and other alternatives are considered, such as mixtures of Gaussians in which each Gaussian component corresponds to a discrete instantiation of the discrete variables [Koller and Friedman, 2009].

2.3 Learning

Learning a BN involves finding both the optimal structure ($\mathcal{G}$) representation and the set of parameters ($\boldsymbol{\Theta}$) that best fit some given data ($\mathcal{D}$). Learning by hand involves an expert specifying the relationships between variables in order to build the structure. Subsequently, the underlying CPDs can be described by the expert. This entails extensive domain knowledge, and becomes infeasible as the size of the model grows with the number of variables. Additionally, uncertainty about the data being handled may lead to wrong domain knowledge.

An alternative to these limitations is using automatic tools to learn the entire model from data, being possible to add expert knowledge during the learning process.

In this section we describe the general methodologies for learning both the parameters and the BN structure automatically. This way, the inputs for the learning procedure are:

- Expert knowledge
- Training data $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where N is the size of the dataset.

2.3.1 Parameter learning

A standard approach for parameter estimation is employing the maximum likelihood criterion, which aims to select the set of parameters that maximize the likelihood function. The likelihood function is defined as the probability of the training dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ given the BN model:

$$p(\mathcal{D}|\boldsymbol{\Theta}, \mathcal{G}) = \prod_{j=1}^{N} p(\boldsymbol{x}_j|\boldsymbol{\Theta}, \mathcal{G}) = \prod_{j=1}^{N} \prod_{i=1}^{n} p(x_{ji}|\boldsymbol{\Theta}_i, \mathcal{G}), \tag{2.6}$$

where $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jn})$ represents the j-th sample of the dataset $\mathcal{D}$, and $\boldsymbol{\Theta}_i$ is the set of parameters for the CPDs of node i. Commonly, the logarithm of the likelihood (log-likelihood) is optimized, as it is considered to provide better numerical precision, and it is defined as:

$$\mathcal{L}(\mathcal{G}, \boldsymbol{\Theta} : \mathcal{D}) = \sum_{j=1}^{N} \sum_{i=1}^{n} log\ p(x_{ji}|\boldsymbol{\Theta}_i, \mathcal{G}). \tag{2.7}$$

Once the likelihood function is defined, the maximum likelihood estimation (MLE) is stated as choosing the parameter configuration $\boldsymbol{\Theta}^*$ that maximizes Equation 2.6 or Equation 2.7.

For cases in which the variables and the relationships between them are defined as linear Gaussian (Equation (2.4)), the MLE for the mean is obtained using an ordinary least squares estimator [Fox, 1997].

The CKDE CPDs are composed of two non-parametric distributions, $f(X_i, \boldsymbol{Pa}_i)$ and $f(\boldsymbol{Pa}_i)$, which involve the estimation of bandwidth matrices $\boldsymbol{H}_i$ (for $f(x_i, \boldsymbol{Pa}_i)$) and $\boldsymbol{H}_i^-$ (for $f(\boldsymbol{Pa}_i)$). However, a CKDE CPD can be fitted by estimating only the bandwidth matrix $\boldsymbol{H}_i$ [Atienza et al., 2022b]. For the estimation of $\boldsymbol{H}_i$, and due to the impossibility of using MLE, the KDE models are trained using other error criterion models, such as the normal reference rule [Scott, 2015], widely used for Gaussian KDE models where the mean integrated squared error is minimized. It defines $\boldsymbol{H}_i = N^{-2/(|\boldsymbol{Pa}_i|+5)}\hat{\boldsymbol{\Sigma}}$, where $\hat{\boldsymbol{\Sigma}}$ is the sample covariance matrix of random variables X_i and $\boldsymbol{Pa}_i$.

2.3.2 Structure learning

The BN structure learning (BNSL) problem is an NP-hard problem [Chickering, 1996] that is well-known in the state-of-the-art research due to the combinatorial explosion of possible DAGs which can represent the relationships among the variables in $\boldsymbol{X}$. Given a dataset $\mathcal{D}$

with n variables and as many examples as variable observations, the objective is to determine the DAG that better depicts the relationships among the variables $\boldsymbol{X}$ found in $\mathcal{D}$. Please, see Scanagatta et al. [2019] and Kitson et al. [2023] for a more extended and recent review on the approaches.

Three main BNSL approaches are identified in the literature: (i) score-based approach, whose objective is to optimize a function that evaluates the quality of the structure given the data; (ii) constraint-based approach, which performs some statistical tests to check conditional independences among the variables; and (iii) hybrid methods that combine both approaches.

2.3.2.1 Score-based approaches

Score-based methods approach the BNSL as an optimization task, and are characterized by:

- Score function. Measures the fitness of each candidate structure.
- Space of structures. All possible structures in which the search is carried out.
- Search method. Optimizer that explores the spaces of structures guided by the score function.

Score function A score function is defined which measures the fitness of a candidate structure to a given dataset. Optimizing it will lead to finding the optimal structure as,

$$\max \ \mathcal{S}(\mathcal{G}, \mathcal{D}),$$

where $\mathcal{G}$ can be all the possible graphs that represent a set of variables $\boldsymbol{X}$, $\mathcal{D}$ is the training data, and $\mathcal{S}$ the defined score function.

Some well-known scores have been used for the BNSL problem. The decomposability property is desirable for computational reasons, which means that the score of a structure given some data is computed as the sum of the local scores of the subgraphs formed by each variable X_i and its parents $\boldsymbol{Pa}_i$,

$$\mathcal{S}(\mathcal{G}, \mathcal{D}) = \sum_{i=1}^{n} \mathcal{S}_i(\mathcal{G}_{X_i, \boldsymbol{Pa}_i}, \mathcal{D}), \tag{2.8}$$

The log-likelihood score (Equation 2.7) can be used as a score function to be optimized. However, this metric favors the complexity of the BN graph. The more complex the structure is, the better the metric value is. To avoid this structural overfitting, a penalization term is added to the formula as,

$$\mathcal{L}(\mathcal{G}, \boldsymbol{\Theta} : \mathcal{D}) - \dim(G)\text{pen}(N), \tag{2.9}$$

where $\dim(G)$ and $\text{pen}(N)$ are the dimension of the model and a non-negative penalization term, respectively.

Depending on the penalization term definition, different metrics are defined. If $\text{pen}(N) = 1$, then Akaike's information criterion (AIC) metric [Akaike, 1974] is obtained. Nevertheless, Bayesian Information Criterion (BIC) [Schwarz, 1978] term is $\text{pen}(N) = \frac{1}{2} log(N)$.

A different strategy is trying to obtain the maximum a posteriori probability of the given structure given the data. Using Bayes formula,

$$p(\mathcal{G}|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{G})p(\mathcal{G}), \tag{2.10}$$

where $p(\mathcal{G})$ denotes the prior distribution over all possible structures and $p(\mathcal{D}|\mathcal{G})$ is the marginal likelihood of the data.

Depending on the probability distribution assumed over the prior information we identify the K2 metric [Cooper and Herskovits, 1992] and the Bayesian Dirichlet (BD) scores [Heckerman et al., 1995, Buntine, 1991], in which uniform and Dirichlet distributions are assumed, respectively.

Space of structures The BNSL problem is known to be NP-hard [Chickering, 1996] because the number of possible structures for a BN with n nodes ($h(n)$) increases more than exponentially with the number of variables n in the given data [Robinson, 1977]:

$$\begin{aligned} h(1) &= 1 \\ h(n) &= \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} h(n-i), \end{aligned} \tag{2.11}$$

and thus, heuristic search algorithms are commonly used.

The space of structures can be reduced by different alternatives. One option that involves adding expertise to the learning procedure, is fixing (*white list*) or forbidding (*black list*) some of the arcs in the graph.

Search methods Once the score metric to be optimized and the search space are defined, different methods can be applied to explore the landscape of solutions.

A well-known method is the Hill Climbing (HC) algorithm. The algorithm starts from an initial graph $\mathcal{G}_0$, in which there are no edges between nodes, and small changes are added iteratively over the structure. The set of available changes are defined in the set of operators $\mathcal{O}$ and usually include adding, removing or reversing arcs in the graph. In each iteration, the modification that better improves the chosen score is elected to be performed over $\mathcal{G}$.

Algorithm 1 describes the HC baseline, which receives as input some training data $\mathcal{D}$, an empty structure $\mathcal{G}_0$, a set of available operators $\mathcal{O}$ and a score metric $\mathcal{S}$. Iteratively, the algorithm finds the best operator that better improves $\mathcal{S}$ (Line 5), applies it (Line 6), and compares to the best one found so far (Line 7). If no improvement is found, then the algorithm is considered to have converged.

Other approaches include a tabu list in order to avoid re-evaluating already analyzed structures in the search space. Regarding other types of heuristics a wide range of approaches, such as particle swarm [Aouay et al., 2013, Quesada et al., 2021], evolutionary algorithms [Blanco et al., 2003, Larrañaga et al., 1996] and simulated annealing [Lee and Kim, 2019] have been applied to solve the BNSL problem in recent decades.

Algorithm 1 Hill-climbing

Input: Training data $\mathcal{D}$, starting structure $\mathcal{G}_0$, set of operators $\mathcal{O}$, score metric $\mathcal{S}$
Output: Optimal structure $\mathcal{G}_{best}$

1: flag $\leftarrow$ **false**
2: $\mathcal{G}_{best} \leftarrow G_0$
3: $\mathcal{G}_{new} \leftarrow G_0$

4: **repeat**
5: $o \leftarrow$ FIND_BEST_OP $(\mathcal{G}_{best}, \mathcal{O})$
6: $\mathcal{G}_{new} \leftarrow o(\mathcal{G}_{best})$
7: **if** $\mathcal{S}(\mathcal{G}_{new}, \mathcal{D}) \geq \mathcal{S}(\mathcal{G}_{best}, \mathcal{D})$ **then**
8: $\mathcal{G}_{best} \leftarrow \mathcal{G}_{new}$
9: **else**
10: flag $\leftarrow$ **true**
11: **end if**
12: **until** flag == **true**

13: **return** $\mathcal{G}_{best}$

Regarding SPBNs, the original paper [Atienza et al., 2022b] proposes a modified version of HC algorithm in which a new operator is added to $\mathcal{O}$ in order to allow changing the type of node (KDE or Gaussian fitted). The authors determined that optimizing traditional scores such as BIC leads to overfitting by adding too many arcs in the structure, so they propose using the K-fold cross-validated log-likelihood over the training set $\mathcal{D}_{trn}$ as the score to be optimized by the greedy algorithm,

$$\mathcal{S}_{CV}^{K}(\mathcal{D}, \mathcal{G}) = \sum_{m=1}^{K} \mathcal{L}(\mathcal{G}, \boldsymbol{\Theta}_{\mathcal{I}_{trn}^{m}} : \mathcal{D}_{\mathcal{I}_{test}^{m}}), \tag{2.12}$$

where $\mathcal{L}(G, \boldsymbol{\Theta}_{\mathcal{I}_{trn}^{m}} : \mathcal{D}_{\mathcal{I}_{test}^{m}})$ is the log-likelihood (Equation (2.7)) of the m-th test fold dataset element in an SPBN composed of parameters $\boldsymbol{\Theta}_{\mathcal{I}_{trn}^{m}}$ and DAG $\mathcal{G}$, K is the number of folds for cross-validation, and $\mathcal{I}^m$ refers to the disjoint sets of indices used in the cross-validation technique.

The overfitting is controlled using the validation set $\mathcal{D}_{val} = \mathcal{D} \backslash \mathcal{D}_{trn}$ which measures the goodness of fit of the new structure at each iteration:

$$\mathcal{S}_{val}(\mathcal{D}_{trn}, \mathcal{D}_{val}, \mathcal{G}) = \mathcal{L}(\mathcal{G}, \boldsymbol{\Theta}_{\mathcal{D}_{trn}} : \mathcal{D}_{val}) \tag{2.13}$$

where $\boldsymbol{\Theta}_{\mathcal{D}_{trn}}$ are the parameters estimated using the training dataset $\mathcal{D}_{trn}$.

The HC proposed for SPBNs is improved by using a tabu list to constrain the search space and explore different directions to escape from local optima. Its pseudocode is presented in Algorithm 2. Lines 8-16 describe the process of searching for new structures that aims to maximize the score function (Equation (2.12)) by applying different operators from $\mathcal{O}$, and Lines 17-24 enable and disable the tabu list depending on the evaluation of Equation (2.13).

The tabu list prevents the algorithm from applying operators that may undo operations that were recently applied. The algorithm uses the hyperparameter λ as the stopping criterion. λ denotes the number of iterations with no improvement in the best score found, and once it is reached, the algorithm is considered to have converged.

Algorithm 2 Greedy hill-climbing for SPBNs

Input: Training data $\mathcal{D}$, starting structure G_0, set of operators $\mathcal{O}$, patience λ, number of folds K

Output: Optimal structure G_{best}

$\mathcal{G}_{best} \leftarrow \mathcal{G}_0$
$\mathcal{G}_{new} \leftarrow \mathcal{G}_0$
$i \leftarrow 0$
$Tabu \leftarrow \emptyset$
$\mathcal{D}_{trn}, \mathcal{D}_{val} \leftarrow$ Split($\mathcal{D}$) # training and validation
while $i < \lambda$ **do**
  $\mathcal{G} \leftarrow \mathcal{G}_{new}$
  for o in $\mathcal{O}$ **do**
    if o does not reverse $o' \in Tabu$ **then**
      $\mathcal{G}_{candidate} \leftarrow o(\mathcal{G})$
      if $\mathcal{S}^K_{CV}(\mathcal{D}_{trn}, \mathcal{G}_{candidate}) > \mathcal{S}^K_{CV}(\mathcal{D}_{trn}, \mathcal{G}_{new})$ **and** $\mathcal{S}^K_{CV}(\mathcal{D}_{trn}, \mathcal{G}_{candidate}) - \mathcal{S}^K_{CV}(\mathcal{D}_{trn}, \mathcal{G}) > 0$ **then**
        $o_{new} \leftarrow o$
        $\mathcal{G}_{new} \leftarrow \mathcal{G}_{candidate}$
      end if
    end if
  end for
  if $\mathcal{S}_{val}(\mathcal{D}_{trn}, \mathcal{D}_{val}, \mathcal{G}_{new}) > \mathcal{S}_{val}(\mathcal{D}_{trn}, \mathcal{D}_{val}, \mathcal{G}_{best})$ **then**
    $\mathcal{G}_{best} \leftarrow \mathcal{G}_{new}$
    Tabu $\leftarrow \emptyset$
    $i \leftarrow 0$
  else
    Tabu $\leftarrow$ Tabu $\cup o_{new}$
    $i \leftarrow i + 1$
  end if
  Update_best_result($\mathcal{G}, o_{new}$)
end while
return $\mathcal{G}_{best}$

2.3.2.2 Constraint-based approaches

Constraint-based approaches perform statistical hypothesis tests in order to study conditional (in)dependence relationships between the variables in the model. Iteratively, this type of approaches ask the data whether two variables X and Y are conditional independent, for all pairs of variables in the model.

The PC algorithm [Spirtes et al., 2000] is the most well-known approach in which starting from a complete undirected graph, the algorithm recursively performs conditional independence tests over pairs of variables, and outputs a completed partially DAG. Several rules are proposed for transforming a completed partially DAG, in which some of the arcs are undirected, to a complete DAG.

2.3.2.3 Hybrid approaches

Combining constraint-based with score-based approaches leads to hybrid approaches. Usually, a constraint-based strategy is used to reduce the search space by performing different statistical tests, and afterwards, a score-based approach is used to find the optimal structure.

An example of hybrid learning approach is the max-min HC [Tsamardinos et al., 2006] (MMHC) algorithm, in which the constraint- and score-based parts are fulfilled by constraint-based max-min parents and children [Tsamardinos et al., 2003] and HC (Algorithm 1) approaches, respectively.

2.4 Inference

Once the BN parameters ($\boldsymbol{\Theta}$) and structure ($\mathcal{G}$) are learned, different types of reasoning can be performed over the model. Inference involves obtaining the distribution of a set of variables $\boldsymbol{X}_1$, given some fixed values $\boldsymbol{x}_2$ for a set of variables $\boldsymbol{X}_2$ (evidences). That is,

$$p(\boldsymbol{X}_1|\boldsymbol{X}_2 = \boldsymbol{x}_2) = \frac{p(\boldsymbol{X}_1, \boldsymbol{x}_2)}{p(\boldsymbol{x}_2)}. \tag{2.14}$$

This task is known to be NP-hard, regardless of using exact [Cooper, 1990] or approximate [Dagum and Luby, 1993] inferences. Nevertheless, in the case of GBNs, there are closed formulas for efficiently performing inference.

2.4.1 Gaussian distributions

In this section we discuss how to compute the conditional probabilities of a GBN. Gaussian CPDs allow to be mathematically defined as multivariate Gaussian distributions ($\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$)[Koller and Friedman, 2009]. Both the structure ($\mathcal{G}$) and the parameters ($\boldsymbol{\Theta}$) are considered for this direct transformation.

First, the mean vector is obtained by computing the most probable value for each of the variables, considering the parameters in the GBN:

$$\mu_i = \sum_{j=1}^{|\boldsymbol{Pa}_i|} \beta_j \mu_j, \tag{2.15}$$

where $|\boldsymbol{Pa}_i|$ denotes the number of parents of X_i in the graph representation $\mathcal{G}$. Root nodes compute the respective mean value without considering any parent.

Second, the covariance matrix is computed in two parts. The first one involves the diagonal elements of the matrix, while the second one involves the off-diagonal elements. Similarly to the mean vector, the diagonal elements are defined as

$$\Sigma_{ii} = \sigma_i^2 + \sum_{j=1}^{|\boldsymbol{Pa}_i|} \beta_j^2 \sigma_j^2. \tag{2.16}$$

Again, the diagonal elements corresponding to root nodes are computed without considering any parent nodes. Off-diagonal elements are computed as

$$\Sigma_{ij} = \Sigma_{ji} = \sum_{k=1}^{|\boldsymbol{Pa}_j|} \beta_k \sum_i k, \tag{2.17}$$

where Σ_{ij} denotes the covariance element between variables X_i and X_j, respectively.

To infer a conditional probability, the probability distribution of a set of variables ($\boldsymbol{X_1}$) is calculated given a fixed value of one or more variables ($\boldsymbol{X_2}$). This is $p(\boldsymbol{X_1}|\boldsymbol{X_2})$. To work with Gaussian probability distributions, some closed formulas are given [Murphy, 2022]. The conditional probabilities are computed as follows: if $\boldsymbol{X} = (\boldsymbol{X_1}, \boldsymbol{X_2})$ is jointly Gaussian with parameters

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}, \quad \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{pmatrix} \tag{2.18}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denote the vector of means and the covariance matrix, respectively, computed in previous steps and split now depending on the conditional probability to be computed. Then marginal probabilities are given by,

$$\begin{aligned} f(\boldsymbol{x}_1) &= \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) \\ f(\boldsymbol{x}_2) &= \mathcal{N}(\boldsymbol{x}_2|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \end{aligned}$$

and the posterior conditional is given by

$$\begin{aligned} f(\boldsymbol{x}_1|\boldsymbol{x}_2) &= \mathcal{N}\left(\boldsymbol{x}_1|\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}\right) \\ \boldsymbol{\mu}_{1|2} &= \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \\ &= \boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \\ &= \boldsymbol{\Sigma}_{1|2}(\boldsymbol{\Lambda}_{11}\boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{12}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)) \\ \boldsymbol{\Sigma}_{1|2} &= \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} = \boldsymbol{\Lambda}_{11}^{-1} \end{aligned} \tag{2.19}$$

Chapter 3

Estimation of Distribution Algorithms

3.1 Introduction

Evolutionary algorithms (EAs) are stochastic approaches inspired in Darwin's theory of evolution, where the concept of natural selection is introduced as the driving force behind the adaptation and evolution of species. Natural selection is based on the idea that in any population, its degree of adaptation of the environment is biased by the characteristics of the individuals. Those individuals with traits that better suit the environment are more likely to survive and reproduce, passing those favorable traits to their offspring. Over generations, this leads to the accumulation of advantageous traits in a population.

Estimation of distribution algorithms (EDAs) are a type of EA in which probabilistic models are used during runtime for modelling the search space and sample new solutions. In this chapter we deeply explain this type of approach, detail the main advantages and review the state of the art in the literature.

Chapter outline

The chapter is organized as follows. Section 3.2 introduces the general outline of traditional EAs and reviews the different research lines. Section 3.3 describes the EDA baseline and its difference of this approach compared to other EAs. Section 3.4 reviews the different variants of EDA found in the literature.

3.2 Evolutionary algorithms

EAs imitate the behavior of the natural evolution, where in each iteration of the algorithm a population is evaluated according to a cost function $g(\cdot)$ to be optimized, and only the best solutions of the generation survive and are used to generate the next population. The general outline of the algorithm is described in Algorithm 3. The degree of adaptation of an individual $\boldsymbol{x}$ to the environment is measured by the cost function to be optimized $g(\boldsymbol{x})$, and the new populations are obtained by applying recombination and mutation operators to the

Algorithm 3 Evolutionary algorithm baseline

Input: Cost function $g(\boldsymbol{x})$
Output: Best individual $\boldsymbol{x}^*$ and cost found $g(\boldsymbol{x}^*)$

```
GENERATE Initial population
EVALUATE population according to g(x)
while not stopping criteria do
    SELECT survivals from population
    GENERATE new population according to selection
    EVALUATE new population according to g(x)
end while
```

best proposed solutions. The process is iteratively repeated until the stopping criteria is met. Note that due to the stochasticity of the algorithm, EAs are non-deterministic approaches, and thus, different executions of the algorithm may converge to different solutions.

Depending on the criteria used to select the survival individuals, and how the next population is generated, different sub-categories are identified in the literature [Dasgupta and Michalewicz, 2014].

- Genetic algorithms (GAs) [Holland, 1975]. GAs are the earliest and most extended type of EA. They involve *crossover* and *mutation* operators during runtime.
- Evolution strategies (ESs) [Rechenberg, 1971]. ESs are a variant of traditional GAs implemented for continuous optimization where the main difference regards the self-adaptation and that the modification of candidate solutions is limited to mutation operators.
- Evolutionary programming (EP) [Fogel et al., 1966]. EP extends the principles of GAs to evolve computer programs or symbolic expressions. In EP, the individuals in the population are represented as trees, and genetic operators are applied to these tree structures.
- Differential evolution (DE) [Storn, 1995]. DE is a variant of traditional GAs for continuous optimization in which new individuals are perturbations of the solutions with a scaled difference between two randomly selected individuals.
- Estimation of distribution algorithms (EDAs) [Larrañaga and Lozano, 2001]. Unlike the previously mentioned EAs, EDAs iteratively use probabilistic models (PMs) to learn the traits of a given population and generate the next one. Depending on the complexity of the probabilistic model, different variants are identified in the literature.

Formally, a single-objective optimization problem is defined as a tuple $(\Omega,\ g)$, where Ω represents the domain of the cost function g as,

$$g : \Omega \to \mathbb{R}, \tag{3.1}$$

and the objective is finding the element $\boldsymbol{x}^* \in \Omega$ such as,

$$g(\boldsymbol{x}^*) \geq g(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega, \tag{3.2}$$

in the case of maximization problems and $\boldsymbol{x}$ is a vector solution with size d. When Ω is graphically represented, its representation is named optimization landscape.

3.3 Estimation of distribution algorithms

EDAs [Larrañaga and Lozano, 2001] differ from traditional GAs in the sense that new solutions are not sampled as a result of *mutation* and *crossover* operators. Instead, EDAs use PMs estimated from a dataset containing the best solutions of the previous iterations; and model the search space for posteriorly sample new solutions from promising subspaces in the landscape.

In addition to the possibility of contemplating dependencies between variables, EDAs have more advantages compared to other EAs. Regarding the explainability of the model [Mihaljević et al., 2021], due to its ease of implementation, introducing modifications or restrictions is very simple. Additionally, the PM of each iteration can be analyzed in order to extract patterns in the heuristic search of the algorithm, being possible in some variants to represent the PM graphically.

Algorithm 4 EDA baseline

Input: Population size N, selection ratio α, cost function g
Output: Best individual $\boldsymbol{x}^*$ and cost found $g(\boldsymbol{x}^*)$

1: $G_0 \leftarrow N$ individuals randomly sampled
2: **for** $t = 1, 2, ...$ until stopping criterion is met **do**
3: Evaluate G_{t-1} according to $g(\cdot)$
4: $G^S_{t-1} \leftarrow$ Select top $\lfloor \alpha N \rfloor$ individuals from G_{t-1}
5: $p_{t-1}(\cdot) \leftarrow$ Learn a probabilistic model from G^S_{t-1}
6: $G_t \leftarrow$ Sample N individuals from $p_{t-1}(\cdot)$
7: **end for**

Algorithm 4 shows the basic EDA approach pseudocode. Only two parameters are required, the size N of the population and the ratio of the population $\alpha \in (0, 1)$, which is selected to update the probabilistic model (Line 4) according to a cost function $g(\boldsymbol{x})$ (Line 3). Each generation is denoted as G_t and is sampled (Line 6) from the probabilistic model $p_{t-1}(\boldsymbol{x})$ estimated with the top $\lfloor \alpha N \rfloor$ individuals from the selected individuals of the previous generation G^S_{t-1} (Line 5). The initial generation G_0 is sampled randomly (Line 1) considering the bounds of the search space and with different sampling methods and assumptions about the density of the solutions.

3.3.1 Toy example

Figure 3.1 depicts a toy example in which a simple EDA approach is applied to solve the OneMax [Eshelman, 1991] optimization problem as,

$$g(\boldsymbol{x}) = \sum_{i=1}^{d} x_i \tag{3.3}$$

where $x_i = \{0, 1\}$ and the objective is to maximize the cost function.

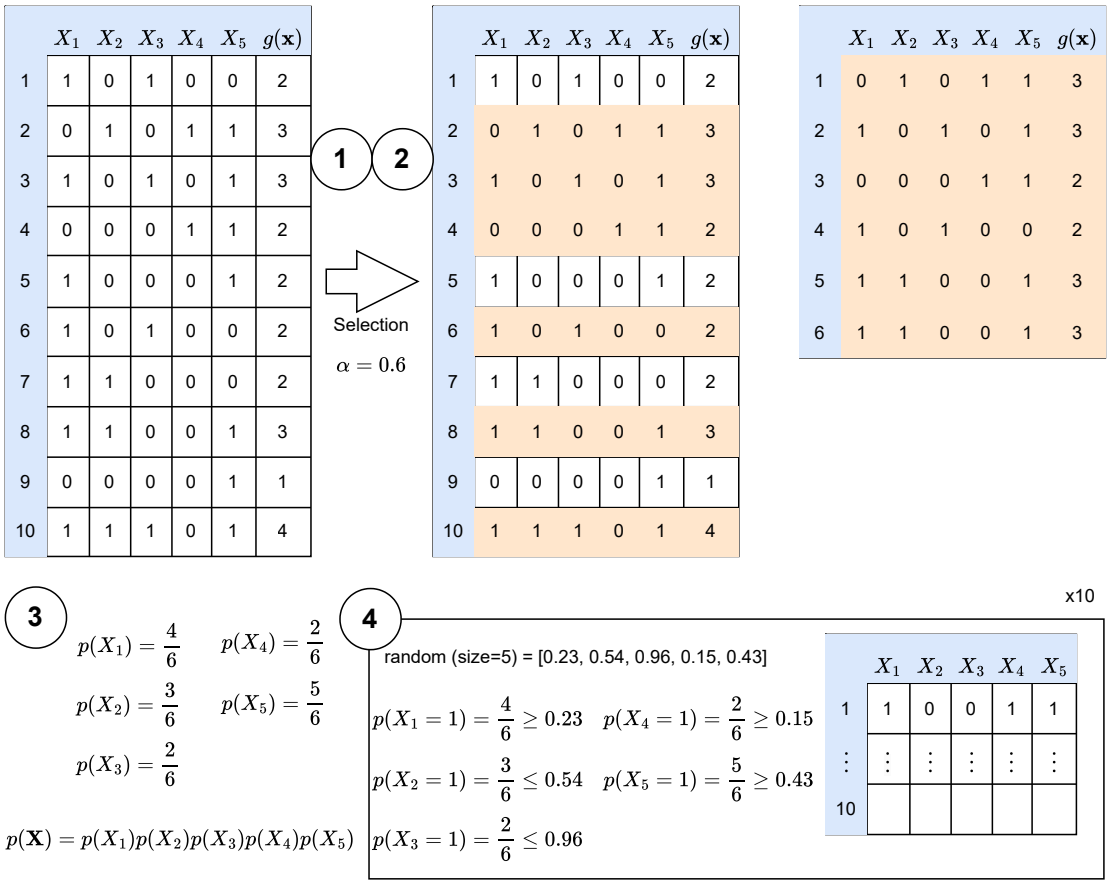


Figure 3.1: Toy example of EDA for the OneMax optimization problem.

The example shows one iteration of the algorithm in which the population size $N = 10$, the selection ratio $\alpha = 0.6$, and the dimension $d = 5$. Initially, (1) the population is evaluated according to the cost function $g(\boldsymbol{x})$ and (2) the top $\lfloor \alpha N \rfloor$ individuals are selected. From this dataset, (3) the marginal probabilities of each variable, $p(X_i)$, are computed. Finally, (4) using the computed probabilities a new population is sampled. This process is repeated iteratively until the stopping criterion is met.

3.4 State of the art

Different PMs have been proposed in the literature to be embedded into the EDA approach [Larrañaga et al., 2012]. Depending on the dependencies considered into the PM we can distinguish between (i) univariate approaches, in which no dependencies are considered, (ii) bivariate approaches, in which pairwise dependencies are considered, and (iii) multivariate approaches, in which no restrictions are considered between the dependencies among the

Name	Probabilistic model	Domain
UMDA_D, UMDA_C, UMDA_C^G	univariate	continuous & discrete
PBIL, PBIL_C	univariate	continuous & discrete
cGA	univariate	continuous & discrete
u_KEDA	univariate	continuous
MIMIC, MIMIC_c^G	bivariate	continuous & discrete
COMIT, TREE	bivariate	discrete
BMDA	bivariate	discrete
ECGA	multivariate	discrete
EBNA_{PC}, EBNA_{K2+pen}, EBNA_{BIC}	multivariate	discrete
BOA	multivariate	discrete
EMNA	multivariate	continuous
EGNA_{BDe}, EGNA_{BIC}	multivariate	continuous
IDEA	multivariate	continuous
LFDA	multivariate	continuous
DEUM	multivariate	continuous
AM	multivariate	continuous

Table 3.1: Classification of the EDAs state of the art according to their type of probabilistic model and domain of application

variables. Some examples of these PMs are shown in Figure 3.2, which are referenced in the following sections.

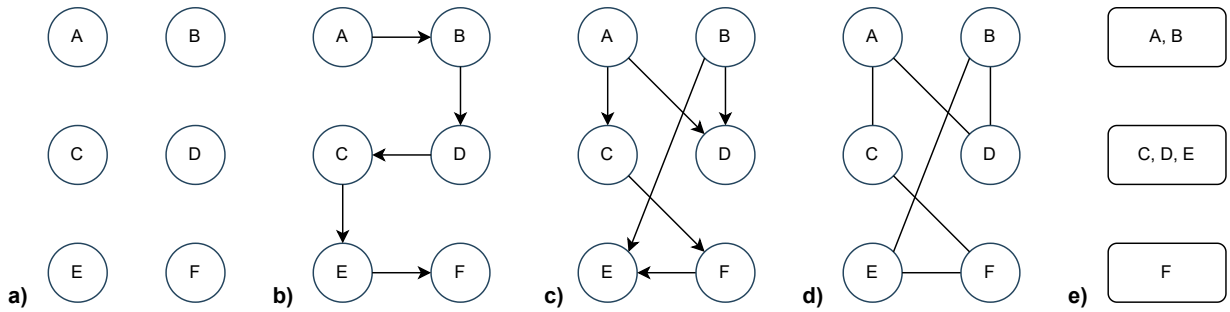


Figure 3.2: Examples of different types of PM structures used in EDAs.

Table 3.1 classifies some single-objective approaches according to their type of probabilistic model and the domain (continuous/discrete) in which they are assigned.

3.4.1 Univariate approaches

Univariate EDAs are the first variant found in the field, in which all variables are assumed to be independent. Thus, the joint probability distribution is computed as the product of the marginal probabilities:

$$p(\boldsymbol{X}) = p(X_1, \ldots, X_d) = \prod_{i=1}^{d} p(X_i), \tag{3.4}$$

where d is the number of dimensions in the optimization problem, and $p(X_i)$ is the marginal probability distribution of variable X_i. Figure 3.2(a) shows an example of the probabilistic model, graphically with six variables.

3.4.1.1 Discrete EDAs

Some examples are the univariate marginal distribution algorithm (UMDA$_D$) [Mühlenbein and Paass, 1996] and population-based incremental learning algorithm (PBIL) [Baluja, 1994]. The latter is a particular case of the former, in which only the best and the worse solutions are considered to update the PM. The compact genetic algorithm (CGa) [Harik et al., 1999] is another example characterized by sampling too few solutions from the PM.

Although the above mentioned approaches were initially proposed for binary and discrete tasks, they were also adapted for continuous optimization.

3.4.1.2 Continuous EDAs

Some examples are the continuous univariate marginal distribution algorithm (UMDA$_C$) [Mühlenbein et al., 1996]), its adaptation using univariate Gaussian density (UMDA$_C^G$), [Larrañaga et al., 2000] and the continuous population-based incremental learning algorithm (PBIL$_C$) [Baluja and Davies, 1997]. While both algorithms use a population to update the probabilistic model, other univariate approaches, such as compact genetic algorithm (cGA) [Harik et al., 1999], update it using just few individuals. Similarly, the stochastic hill climbing with learning by vector of normal distributions (SHCLVND) [Rudlof and Koppen, 1996] uses the Hebbian rule to update the PM.

The adaptation from the discrete approaches to continuous domains consisted in using independent Gaussian distributions approximated for each variable. Recently, a univariate EDA was proposed in which an independent kernel density estimation is used for each variable [Luo and Qian, 2009] (u_KEDA).

Regarding other types of PM, it is worth to mention the marginal histogram model, in which histograms are estimated and sampled in each iteration [Tsutsui et al., 2001]

Due to the simplicity and ease of implementation of these algorithms, these approaches have been widely analyzed for theoretical results [Krejca and Witt, 2017, Zheng and Doerr, 2023].

3.4.2 Bivariate approaches

Bivariate EDAs restrict each variable to depend at most on one parent. Figure 3.2(b) depicts an example of the probabilistic model, in which a chain structure is shown, graphically. Formally, the maximum-in-degree of each node in the graph, is restricted to be at most one. Then, assuming a set of variables, ordered following an ancestral order in the graph, the joint probability distribution over all the variables in the graph is

$$p(\boldsymbol{X}) = p(X_1, \ldots, X_d) = \prod_{i=1}^{d} p(X_i|X_j), \tag{3.5}$$

where $p(X_i|X_j) = p(X_i|X_{i-1})$ in the case that there is an arc between X_i and X_{i-1}, and $p(X_i|X_j) = p(X_i)$ in the case that X_i has no parents in the graph (root node).

3.4.2.1 Discrete EDAs

An example of this variant for discrete domains is the Mutual-Information-Maximizing Input Clustering (MIMIC) [De Bonet et al., 1997], which in each iteration uses the mutual information between pairs of variables to build the chain structure. From an initial chain structure, the algorithm changes the edges in the graph guided by the mutual information, and ends with a pseudo optimal chain structure.

A PM with a tree structure was proposed by [Baluja and Davies, 1997] with combining optimizers with mutual information trees (COMIT) algorithm where a local optimizer is used to each generated individual in the algorithm. An upgrade of this algorithm was TREE [Larrañaga and Lozano, 2001] in which this step was removed.

Other examples of this variant for discrete domains include bivariate marginal distribution algorithm (BMDA) [Pelikan and Mühlenbein, 1999] in which an undirected acyclic graph is assumed.

3.4.2.2 Continuous EDAs

MIMIC was posteriorly adapted to continuous domains (MIMIC$_c^G$) [Larrañaga et al., 1999, Larrañaga et al., 2000] by using bivariate Gaussians for the edges represented in the chain structure.

3.4.3 Multivariate approaches

When the number of dependencies considered between the variables involved in the optimization problem is low, or there are none at all, the univariate and bivariate approaches fit properly. Nevertheless, when a high number of dependencies or a more complex structure is assumed, these approximations are too simple. This is covered by multivariate models, where there are no restrictions on the maximum-in-degree of the nodes in the network.

3.4.3.1 Discrete EDAs

Regarding extensions of previously explained approaches, it is worth to mention extended cGA [Harik, 1999] for discrete domains, in which those variables that are dependent in a given iteration, are integrated together in a cluster, and each cluster is considered as an independent variable in the model. An example of this structure is shown in Figure 3.2(e).

More complex probabilistic models include the use of BNs embedded in the approach. Figure 3.2(c) shows an example of the probabilistic model. Regarding discrete variables, we find estimation of Bayesian network algorithm (EBNA) [Etxeberria and Larrañaga, 1999], which learns a discrete BN in each iteration of the algorithm. Depending on the BN structure learning approach used in each iteration, different variants are identified such as EBNA$_{PC}$, EBNA$_{K2+pen}$ and EBNA$_{BIC}$, in which the PC [Spirtes et al., 2000] constraint-based algorithm,

the K2 approach with penalization term [Cooper and Herskovits, 1992] and a score-based approach with BIC score [Schwarz, 1978] are used, respectively. A similar approach is Bayesian optimization algorithm (BOA) [Pelikan et al., 1999] in which a BN is learned using BDe score [Heckerman et al., 1995, Buntine, 1991].

3.4.3.2 Continuous EDAs

Estimation of multivariate normal algorithm (EMNA) [Larrañaga and Lozano, 2001] estimates the vector of means and the covariance matrix of a multivariate Gaussian in each iteration, from which the next generation is sampled.

If a GBN is learned in each iteration, we find the estimation of Gaussian Bayesian network algorithm (EGNA) [Larrañaga et al., 2000, Larrañaga et al., 1999]. Again EGNA_{BDe} and EGNA_{BIC} variants are identified depending on the score used in the score-based BN learning algorithm used. The former uses BDe [Schwarz, 1978], while the latter uses BIC score [Schwarz, 1978]. Similar approaches are iterated density estimation algorithm (IDEA) [Bosman and Thierens, 2000] which uses mixtures of Gaussians, and learned factorized distribution algorithm (LFDA) [Mühlenbein and Mahnig, 1999], which restricts the number of parents in the GBN.

Other approaches include the use of Markov networks in each iteration (DEUM) [Brownlee, 2009]; mixtures of models, such as adaptive Gaussian mixture model (AM) [Gallagher et al., 1999], in which a mixture of Gaussian distributions is estimated in each iteration; copulas and vines [Soto et al., 2012]; reinforcement learning assistance [Paul and Iba, 2003]; Boltzmann machines [Shim et al., 2013], auto encoders [Probst and Rothlauf, 2020] or generative adversarial networks [Probst, 2015].

3.4.4 Multi-objective approaches

3.4.4.1 Problem formulation

Multi-objective optimization deals with the simultaneous optimization of multiple objectives $(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})$ in the context of decision variables, where some of them might be conflicting. Multi-objective optimization aims at identifying a set of solutions that represent the trade-offs between different objectives, i.e. the best Pareto frontier approximation. Then, a multi-objective optimization problem is defined as

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) \\ & \text{subject to } \boldsymbol{x} \in \mathbb{R}^d \end{aligned} \tag{3.6}$$

where m and d are the number of objectives and variables involved in the problem, respectively, and the optimization criterion is maximization.

For a maximization problem and a set $\mathcal{X}$ of solutions $\boldsymbol{x'}$, a solution $\boldsymbol{x}^*$ is part of the Pareto frontier $\mathcal{X}_{PF}$ if,

$$g_i(\boldsymbol{x}^*) \geq g_i(\boldsymbol{x}) \text{ for all } i = 1, \ldots, m \text{ and for all } \boldsymbol{x} \in \mathcal{X} - \mathcal{X}_{PF}.$$

where $g_i(\boldsymbol{x})$ is the evaluation of solution $\boldsymbol{x}$ in the objective function i. Thus, the condition essentially states that no other solution from $\mathcal{X}$ can make all objectives better than those of Pareto frontier solutions.

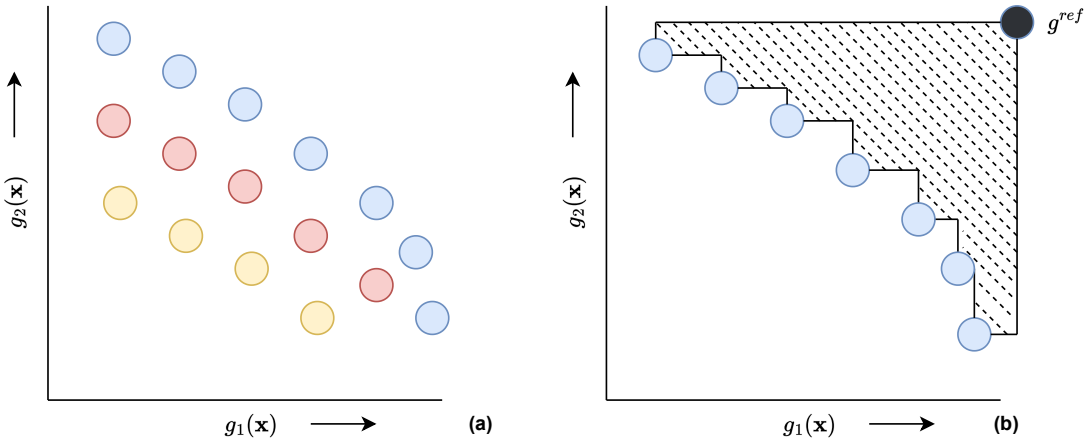


Figure 3.3: Panel (a) shows first, second and third Pareto frontiers represented as blue, red and yellow circles, respectively, for maximizing $g_1(\boldsymbol{x})$ and $g_2(\boldsymbol{x})$ objectives simultaneously. Panel (b) shows the hypervolume (HV) indicator computation for the first Pareto frontier and the reference point g^{ref}.

The Pareto frontier is identified in Figure 3.3(a) with blue circles, among all the possible solutions, where $g_1(\boldsymbol{x})$ and $g_2(\boldsymbol{x})$ are to be maximized. The second Pareto frontier is identified with red circles, and defined as the Pareto frontier if the first Pareto frontier (blue circles) was removed. None of the red samples maximizes both objectives better than any of the blue set of samples. The same happens with the third frontier, represented with yellow circles, and defined as the Pareto frontier if red and blue samples are removed.

Quality indicators in multi-objective optimization are quantitative measures used to assess the performance and characteristics of solutions generated by multi-objective optimization algorithms [Li and Yao, 2019].

The hypervolume (HV) measures the volume of the objective space that is dominated by a set of solutions. It quantifies how well a set of solutions covers the entire Pareto front. A higher hypervolume indicates a better spread of solutions. HV of a set $\boldsymbol{S}$, given a reference point $g^{ref} = (g_1^{ref}, g_2^{ref}, \ldots, g_m^{ref})$, is the volume of the union of the hypercubes determined by each of its solutions $\boldsymbol{s} \in \boldsymbol{S}$ and g^{ref},

$$HV(\boldsymbol{S}, g^{ref}) = \Lambda(\bigcup_{\boldsymbol{S}} \{ \left[g_1(\boldsymbol{s}), g_1^{ref} \right] \times \cdots \times \left[g_m(\boldsymbol{s}), g_m^{ref} \right] \}), \tag{3.7}$$

where g_i^{ref} refers to the reference ideal point for objective function g_i and $\Lambda(\cdot)$ refers to the Lebesgue measure. Figure 3.3(b) illustrates an example of the HV computation over the first Pareto frontier for $m = 2$ objectives.

The diversity metric (DM) quantifies the dissimilarity or spacing between solutions in a given set. It can be measured with different geometric distances or using the crowding distance (CD),

which measures how densely solutions are distributed along the Pareto frontier in terms of the objectives. This metric characterizes a set of solutions and, depending on the optimization task to be solved, its evolution during runtime may vary. The CD of two solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as,

$$CD(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{m} (g_k(\boldsymbol{x}_i) - g_k(\boldsymbol{x}_j))^2, \tag{3.8}$$

where $g_k(\boldsymbol{x}_i)$ and $g_k(\boldsymbol{x}_j)$ are the objective function values of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ for the objective g_k, respectively.

Then, we define DM over a set of solutions $\mathcal{X}$ as,

$$DM(\mathcal{X}) = \frac{1}{|\mathcal{X}|(|\mathcal{X}| - 1)} \sum_{\substack{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X} \\ \boldsymbol{x}_i \neq \boldsymbol{x}_j}} CD(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{3.9}$$

where $|\mathcal{X}|$ is the number of solutions in $\mathcal{X}$.

3.4.4.2 State of the art

When more than one cost function are desired to be optimized in parallel, we find multi-objective approaches. Thus, the main difficulty in this variant regards in how to rank the solutions and update the PM used in the framework.

Regarding approaches which use BNs we find Pareto BOA [Schwarz and Ocenasek, 2001], in which the ranking is based in the Pareto strength method and uses the BOA baseline. Bayesian multiobjective optimization algorithm (BMOA) [Laumanns and Ocenasek, 2002] uses a similar ranking method to select subsets of solutions for the BN update. Decision tree based multiobjective EDA (DT-MEDA) [Zhong and Li, 2007] uses GBNs an bases the selection method in the non-dominated sorting genetic algorithm (NSGA-II) [Deb et al., 2002].

Some approaches use a mixture of probabilistic models to better approximate the Pareto frontier, such as multiobjective mixture-based IDEA (MIDEA) [Bosman and Thierens, 2002]. More examples include multiobjective Parzen-based EDA (MOPEDA) [Costa and Minisci, 2003] which uses a mixture of kernel methods to reduce the risk in the estimation of the PM; multi-objective hierarchical BOA (mohBOA) [Pelikan et al., 2005], in which each component is a BN.

ECGa was extended to multiobjective approaches (meCGA) [Soh and Kirley, 2006] in which each component is a univariate PM and the Pareto dominance ranking is used in the selection step.

Some approaches have focused in keeping a certain diversity among the solutions sampled in the algorithm, such as diversity preserving multi-objective rBOA (dp-MrBOA) [Ahn and Ramakrishna, 2007] which uses adaptive sharing and dynamic crowding methods.

Is worth mentioning approaches which have been combined with other algorithms such as multi-objective Bayesian optimization algorithm [Khan et al., 2002] in which BOA and non-dominated sorting genetic algorithm (NSGA-II) [Deb et al., 2002] are combined; the multi-objective hierarchical BOA [Pelikan et al., 2005], which modifies the previous algorithm

by identifying promising solutions using clustering; and the multi-objective estimation of distribution algorithm [Karshenas et al., 2013], in which BNs are used to capture the dependencies between the decision variables and the objectives to be optimized. Other approaches have been found in which neural networks assist the EDA approach [Martí et al., 2013, 2016].

While most of the previously mentioned approaches use Pareto approximations for ranking the solutions, some other proposals have been found in the literature such as decomposition methods [Shim et al., 2012, Zhang et al., 2009].

Chapter 4

Quantum Computing

4.1 Introduction

Quantum computing (QC), a cutting-edge field at the intersection of physics and computer science, represents a paradigm shift in the way we process information. Unlike classical computers, which rely on bits as fundamental units of data, QC leverages quantum bits or qubits. QC harnesses the principles of quantum mechanics, such as superposition and entanglement, allowing quantum computers to perform complex calculations exponentially faster than their classical counterparts for certain tasks. The promise of quantum computing lies in its potential to solve problems that are currently intractable for classical computers, ranging from optimization challenges to simulating quantum systems.

Currently, the state of the art of quantum computers is the noisy intermediate-scale quantum (NISQ) era, which is characterized by quantum computers with hundreds of qubits and no error correction. Thus, there is a need to develop algorithms that do not require a large number of qubits and that offer resilience to the presence of quantum noise (which characterizes quantum devices). Fault tolerance in QC involves implementing error correction techniques that can detect and correct errors without compromising the overall quantum computation.

Variational quantum algorithms (VQAs) are well-known NISQ era approaches similar to classical neural networks, in which a parameterized quantum system is used to solve a given problem. Two examples are quantum approximate optimization algorithm (QAOA) and variational quantum eigensolver (VQE). This thesis focuses in the use, improvement and analysis of this type of algorithms.

Chapter outline

The outline of this chapter is organized as follows. Section 4.2 introduces the foundations of quantum computing and how it is represented into a quantum circuit. Section 4.3 introduces the background of VQAs. Section 4.4- 4.5 deeply explains QAOA and VQE, respectively. Section 4.6 compares different types of quantum noise embedded into NISQ-era quantum devices.

4.2 Foundations of quantum computing

4.2.1 Qubits

In classical computation the minimal piece of information is is a bit, which can represent either zeros (0) or ones (1). In contrast, in QC all the computations are carried out by the manipulation of quantum bits (qubits).

The representation of two possible qubit states following Dirac notation are the $|0\rangle$ and $|1\rangle$, which correspond to 0 and 1 classical states, respectively. However, the main difference between both types of computations rests in that qubits can represent states different from $|0\rangle$ and $|1\rangle$, by linear combinations of quantum states. This is referred in the literature as the *quantum superposition*:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{4.1}$$

where α and β are complex numbers representing the amplitudes, and $|\alpha|^2 + |\beta|^2 = 1$. Here, $|0\rangle$ and $|1\rangle$ are known as the computational basis states of our quantum system. Although a qubit can represent more states than $|0\rangle$ and $|1\rangle$, when it is measured, then it collapses to 0 and 1 classical states with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. A one-qubit system with equal probability to be measured in both computational basis states is represented as,

$$|+\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,, \tag{4.2}$$

where $P(0) = P(1) = (\frac{1}{\sqrt{2}})^2 = 0.5$.

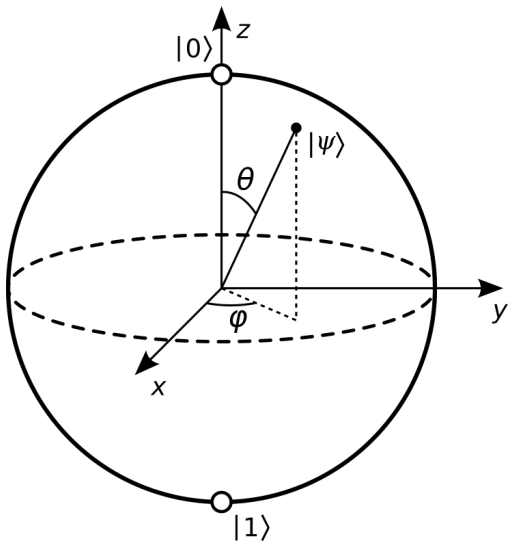


Figure 4.1: Bloch sphere representation of a qubit.

Geometrically, a qubit can be represented as a *Bloch sphere* (Figure 4.1), where θ and φ place a point in the three-dimensional space of the sphere. Then, the quantum state is represented following *Dirac notation* as,

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle\,, \tag{4.3}$$

4.2.2 Multiple qubits

The combination of two qubits in a quantum system allows to represent the quantum states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$, equivalently to classical computation using bits. The quantum state is described as

$$|\psi\rangle = \delta_{00}\,|00\rangle + \delta_{01}\,|01\rangle + \delta_{10}\,|10\rangle + \delta_{11}\,|11\rangle\,, \tag{4.4}$$

where δ_i are the corresponding amplitudes to each of the computational basis states of the two-qubit system, in which each of the basis states owns an associated probability $P(i) = |\delta_i|^2$ to be measured, and $\sum_{i\in(00,01,10,11)} |\delta_i|^2 = 1$.

The superposition over all the computational basis states of a two-qubit system is represented as

$$|+\rangle = \frac{1}{\sqrt{4}}\,|00\rangle + \frac{1}{\sqrt{4}}\,|01\rangle + \frac{1}{\sqrt{4}}\,|10\rangle + \frac{1}{\sqrt{4}}\,|11\rangle\,, \tag{4.5}$$

where $P(00) = P(01) = P(10) = P(11) = (\frac{1}{\sqrt{4}})^2 = 0.25$.

This notation is easily extended to an n-dimensional qubit system, where n is the number of qubits and 2^n the number of computational basis states.

4.2.3 Quantum gates

In classical computation, logical gates can be applied over a set of classical bits to perform manipulations over the information initially represented. Analogously, QC defines a set of quantum operations to be applied over a quantum state.

A quantum gate over n qubits is defined as a unitary matrix of size $n \times n$, which is applied over the quantum system.

4.2.3.1 One-qubit gates

A one-qubit quantum system (Equation 4.1) is represented in a vector notation as

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix}, \tag{4.6}$$

where α and β are the amplitudes of $|0\rangle$ and $|1\rangle$ the computational basis states, respectively.

The set of Pauli operators (X, Y, Z) includes the universal one-qubit quantum gates which perform a rotation of π radians in the X, Y and Z-axis of the qubit Bloch sphere, respectively. Table 4.1 shows the gate and matrix representations of each of the Pauli operators.

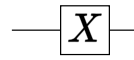
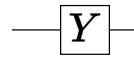
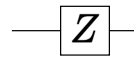
Operator	Gate representation	Matrix representation
Pauli-X		$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y		$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z		$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Table 4.1: Gate and matrix representations of one-qubit X, Y and Z Pauli operators.

Note than Pauli-X is equivalent to the classical NOT gate. Applying Pauli-X over the $|0\rangle$ state, computes the $|1\rangle$ quantum state:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix},$$

where the probabilities of $|0\rangle$ and $|1\rangle$ have been exchanged after the computation.

Alternatively, the set of rotation operators $(R_X(\theta), R_Y(\theta), R_Z(\theta))$ includes the universal one-qubit quantum gates which perform a parametric rotation over the X, Y and Z-axis of the qubit Bloch sphere, respectively, where $\theta \in [0, 2\pi]$ refers to the angle of rotation (in radians). Table 4.2 shows the gate and matrix representations of each of the operators.

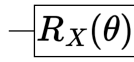
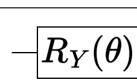
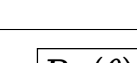
Operator	Gate representation	Matrix representation
$R_X(\theta)$		$R_X(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$
$R_Y(\theta)$		$R_Y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$
$R_Z(\theta)$		$R_Z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$

Table 4.2: Gate and matrix representations of one-qubit $R_X(\theta), R_Y(\theta)$ and $R_Z(\theta)$ operators.

Additionally, the Hadamard gate imposes a linear superposition among all the computational basis states of the quantum system. It is often used as an initial state of a quantum system. Table 4.3 shows the gate and matrix representations of Hadamard operator.

4.2.3.2 Multi-qubit gates

Analogous to classical computation, in QC there exist some two-qubit gates in which two qubits are considered for the computation. The use of two-qubit gates allows the system to have quantum entanglement.

Operator	Gate representation	Matrix representation
Hadamard		$H = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

Table 4.3: Gate and matrix representation of the Hadamard gate.

In the two-qubits case, the quantum system (Equation 4.4) is represented in a vector notation as

$$\begin{bmatrix} \delta_{00} \\ \delta_{01} \\ \delta_{10} \\ \delta_{11} \end{bmatrix}, \tag{4.7}$$

where δ_{00}, δ_{01}, δ_{10} and δ_{11} are the amplitudes of the computational basis states.

Table 4.4 shows the gate and matrix representation of controlled-NOT (CNOT), controlled-Z (CZ) and swap gates. CNOT gate executes a Pauli-X gate over the target qubit (q_1) depending on the control qubit (q_0). If control qubit is $|1\rangle$, then a Pauli-X gate is executed over the target qubit, and none operation is executed otherwise. Equivalently happens in the case of CZ, where the control qubit is q_0 and the target qubit is q_1. SWAP gate swaps the state between two qubits q_0 and q_1.

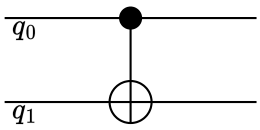
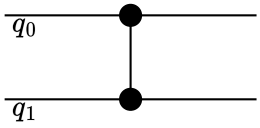
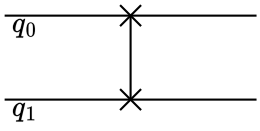
Operator	Gate representation	Matrix representation
Controlled-NOT		$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled-Z		$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 4.4: Gate and matrix representation of controlled-NOT (CNOT), controlled-Z (CZ) and swap (SWAP) gates.

As an example, applying a CNOT gate over the quantum state $|01\rangle$ does not modify the

quantum system:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

where the amplitude of the computational basis states remains constant. However, applying a CNOT gate over the quantum state $|10\rangle$ modifies it as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

where the second qubit flips the state resulting to the quantum state $|11\rangle$.

4.2.4 Quantum measurement

Quantum measurement involves the collapse of the qubit into a classical bit. Although there exists the measurement in the three axis, the z-axis is commonly used in the literature. The computational basis states are probabilistically obtained according to the amplitudes represented in the quantum system. Figure 4.2 shows the gate representation for the measurement.

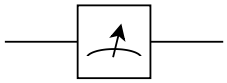


Figure 4.2: Qubit measurement gate in the z-axis.

4.2.5 Quantum circuits

A quantum circuit is a computational routine in which sequential operators (Section 4.2.3) are applied over a set of qubits. The circuit is designed to be read from left to right, where each line refers to a different qubit. Note that any quantum computation can be expressed in a quantum circuit as a combination of one-qubit and two-qubit gates [Nielsen and Chuang, 2002].

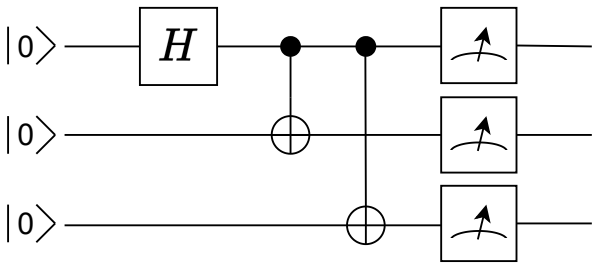


Figure 4.3: GHZ quantum circuit.

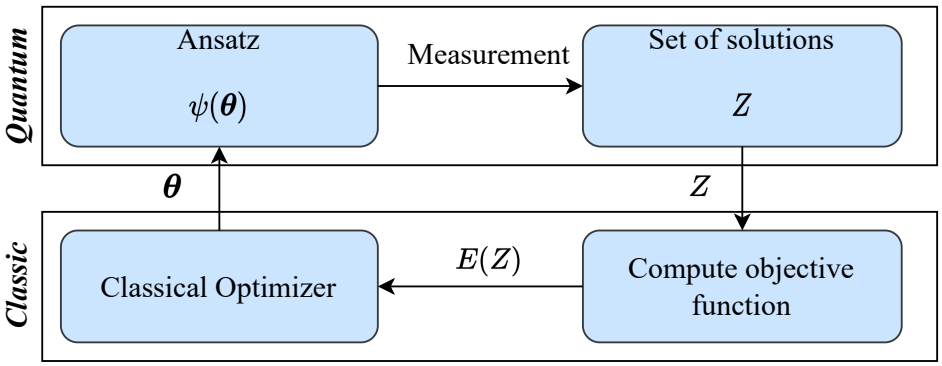


Figure 4.4: VQA workflow where, iteratively, the classical optimizer proposes a new set of parameters ($\boldsymbol{\theta}$) for the *ansatz* $\psi(\boldsymbol{\theta})$, which is then measured (Z), and the objective function is computed ($E(Z)$).

An example of a three qubit system is shown in Figure 4.3 which represents the following quantum state, also known in the literature as GHZ state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle\,, \tag{4.8}$$

where $P(000) = P(111) = 0.5$.

4.3 Variational quantum algorithms

Variational quantum algorithms (VQA) [Cerezo et al., 2021a] are classical-quantum hybrid approaches that have been continuously adapted and modified in the literature. The three main modules are (i) an objective cost function to be minimized, (ii) a quantum parametric circuit (henceforth called as *ansatz*), and (iii) a classical optimization technique that manipulates the *ansatz*.

In the following sections, we deeply explain each of the modules, which are represented in the workflow in Figure 4.4.

4.3.1 Objective cost function

A Hamiltonian (H) is an Hermitian operator that describes a physical system, yielding the energy of a quantum state, which is often used as the objective cost function to be minimized in VQAs. Finding the global minima of the Hamiltonian (ground state) implies finding the ground state of the quantum system.

There exist a wide range of objective functions in the literature [Bharti et al., 2022], such as expectation value or conditional value at a risk [Barkoutsos et al., 2020].

Expectation value is simplified as

$$\min_{\boldsymbol{\theta}} E(\boldsymbol{\theta}, \{\langle H\rangle_{U(\boldsymbol{\theta})}\}). \tag{4.9}$$

where $\boldsymbol{\theta}$ is the set of parameters to be manipulated classically, and $\langle H \rangle_{U(\boldsymbol{\theta})}$ describes the measurements of a quantum system as

$$\langle H \rangle_{U(\boldsymbol{\theta})} = \langle 0 | U^T(\boldsymbol{\theta}) H U(\boldsymbol{\theta}) | 0 \rangle , \tag{4.10}$$

where $U(\boldsymbol{\theta})$ is the unitary state generated by an *ansatz*, parameterized by $\boldsymbol{\theta} \in [0, 2\pi]$ parameters.

Conditional value at a risk (CVaR) is a modification of expectation value (Equation 4.9), in which only part of the measurements are considered to compute the objective function. Given a set of energy basis measurements $\{E_1, \ldots, E_M\}$, sorted in an ascending order, CVaR is simplified as

$$\text{CVaR}(\alpha) = \frac{1}{\lceil \alpha M \rceil} \sum_{k=1}^{\lceil \alpha M \rceil} E_k, \tag{4.11}$$

where only the tail of the distribution of energy measures are considered and $\alpha \in (0, 1]$ is the portion of measures considered.

4.3.2 Quantum parametric circuit

An *ansatz* is a quantum circuit which is parameterized by a set of parameters $\boldsymbol{\theta}$, and its quantum state is denoted as,

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta}) |\psi_0\rangle , \tag{4.12}$$

where $|\psi_0\rangle$ is the given initial state, typically set to the $|0\rangle$ state, i.e., $|00\cdots0\rangle^{\otimes n}$ state, where n is the number of qubits of the system.

The election of the *ansatz* influences in both, (i) the quality of solutions for a given problem, and (ii) the possibility of being executed in a given quantum device. Given both limitations, we can classify the *ansatz* in problem-inspired and hardware-efficient.

On the one hand, the problem-inspired ansatz involve the analysis of the underlying physics of the Hamiltonian to be solved, and the decomposition of it into Pauli operators. An example of this type is quantum approximate optimization algorithm (QAOA) [Farhi et al., 2014].

On the other hand, there exist several device constraints which lead to a specific optimal design of an *ansatz* for each architecture. Some of these constraints are:

- Set of gates. Each quantum device allows a different set of operators to be executed in the *ansatz*.
- Qubit connectivity. Each quantum device assumes a connectivity map (topology) which describes which qubits are connected to which. Thus, executing a two-qubit operation between qubits is only possible if both qubits are connected in the topology. If they are not, then a combination of SWAP gates is carried out to place both qubits together, which involves a deeper (longer) *ansatz*.
- Coherence times. The total *ansatz* execution time must be shorter than the coherence time to ensure the results to be coherent.

4.3.3 Parameter optimization

The *ansatz* parameter tuning consists of a continuous optimization where the parameters represent angle rotations of the qubits (in radians), and thus, are restricted to $[0, 2\pi]$. However, depending on the type of VQA to be used, and the number of layers that compose the circuit, the number of parameters vary substantially. Additionally, it is necessary to decide whether to prioritize the algorithm runtime, or the quality of the solutions given by the optimizer.

Considering the previous optimization requirements, we list the following state-of-the-art optimizers grouped according to whether they are gradient-based or not [Bharti et al., 2022], which will be analyzed in the comparison performed in this work (Chapter 10).

A common technique is to compute the gradients over the landscape of solutions given the d dimensions, where d is the number of parameters to be tuned in the *ansatz*. This computation is carried out in each step of the algorithm given the direction towards finding better solutions. Some gradient-based optimizers include:

- Conjugate gradient method (CG) [Hestenes and Stiefel, 1952]: designed for systems of linear equations whose matrices are symmetric and positive-definite.
- Limited-memory Broyden–Fletcher–Goldfarb-Shanno Bound algorithm (L-BFGS-B) [Byrd et al., 1995]: limited memory-based and designed for non-linear optimization problems.
- Sequential least squares programming (SLSQP) [Boggs and Tolle, 1995]: finds a local search direction by solving the second-order local approximation of the objective function.
- ADAM [Kingma and Ba, 2014]: stochastic version of the gradient descent.
- Gradient descent [Ruder, 2016]: a first-order optimization algorithm, commonly used in deep learning to optimize the loss function.

Alternatively, some gradient-free optimizers have been included in the literature, such as:

- Constrained optimization by linear approximation (COBYLA) [Powell, 1998]: uses linear approximations of the objective functions, and is mostly used when the derivative of the objective function is unknown.
- Simultaneous perturbation stochastic approximation (SPSA) [Spall, 1992]: suited for large-scale optimizations.
- Other approaches include EAs [Anand et al., 2021] or reinforcement learning [Garcia-Saez and Riu, 2019].

4.3.3.1 Barren Plateaus

Increasing the complexity of the *ansatz* in terms of number of qubits and circuit depth leads to theoretical challenges such as Barren plateaus (BPs) [McClean et al., 2018, Ragone et al., 2023, Cerezo et al., 2021b, Wang et al., 2021]. BPs are known to be exponential vanishing gradients, leading to flat landscapes in which gradient-based optimizers get stuck and can no

longer improve the results. Formally, BPs are characterized by the following properties,

$$\mathbb{E}(\partial_k E(\boldsymbol{\theta})) = 0, \tag{4.13}$$

$$\mathrm{Var}(\partial_k E(\boldsymbol{\theta})) \in \mathcal{O}(\exp(-n)), \tag{4.14}$$

where $\mathbb{E}(\partial_k E(\boldsymbol{\theta}))$, and $\mathrm{Var}(\partial_k E(\boldsymbol{\theta}))$ are the expectation and variance of the gradients for the objective cost function, respectively, $\boldsymbol{\theta}$ is the set of parameters of the unitary representing the *ansatz*, and n is the number of qubits.

Recently, Pérez-Salinas et al. [2023] have stated that the norm of the gradients can be bounded efficiently with a small number of quantum measurements (which grows linearly with the number of parameters), without the need of optimizing the *ansatz* parameters. This method performs a random walk in the parameter space and measures the entropy of fluctuations of cost values along the walk. The measured entropy value can be used to analytically bound the gradient of the cost function along the walk. We notice that the average of the gradient field (henceforth named as IC) can be approximated by the average along the random walk (due to Monte Carlo integration):

$$\| \nabla E \| \approx \mathbb{E}_W \left(\sum_{k=1}^{m} (\partial_k E(\boldsymbol{\theta}))^2 \right) = \sum_{k=1}^{m} \mathrm{Var}_W(\partial_k E(\boldsymbol{\theta})), \tag{4.15}$$

where Var_W denotes the variance found in the objective cost function using m different $\boldsymbol{\theta}$ parameters generated from a random walk W.

4.4 Quantum approximate optimization algorithm

QAOA was originally proposed in Farhi et al. [2014] for solving combinatorial optimization problems. The QAOA *ansatz* is composed of $p \in \mathbb{N}$ layers, which internally builds two sequential operators: (i) the cost operator $U(H_C, \gamma)$ parameterized by $\gamma \in [0, 2\pi]$, which is built specifically for each problem instance and encodes the classical cost function through a combination of single and two-qubit rotation gates,

$$U(H_C, \gamma) = e^{-i\gamma H_C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}, \tag{4.16}$$

where C_α is the cost function to be minimized, and m the number of clauses that define the classical cost function to be optimized; and (ii) the mixed operator $U(H_B, \beta)$ parameterized by $\beta \in [0, 2\pi]$, which represents a rotation in the X-axis in each qubit (σ_j^x),

$$U(H_B, \beta) = e^{-i\beta H_B} = \prod_{j=1}^{n} e^{-i\beta\sigma_j^x}, \tag{4.17}$$

where n is the number of qubits of the quantum system.

Thus, an *ansatz* composed by p layers has $2p$ parameters to be optimized $\boldsymbol{\theta} = (\gamma_1, \beta_1, \ldots, \gamma_p, \beta_p)$, as shown in the *ansatz* example in Figure 4.5. The quantum state represented by the QAOA *ansatz* is,

$$\psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) = U(H_B, \beta_p)U(H_C, \gamma_p) \cdots U(H_B, \beta_1)U(H_C, \gamma_1) \left| s \right\rangle, \tag{4.18}$$

where $p \geq 1$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, and $\langle s \rangle$ is the superposition state over the computational quantum states.

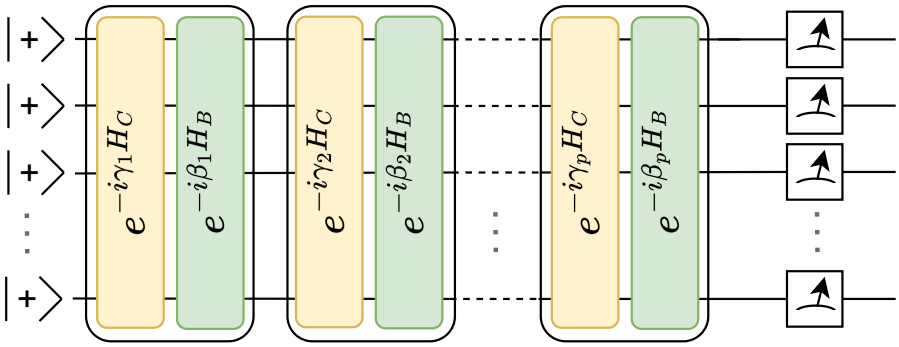


Figure 4.5: An *ansatz* with p layers and $2p$ parameters to be tuned. The quantum circuit starts from a superposition state over all the possible computational states, then applies p layers, and measures the qubits in the Z-axis.

4.5 Variational quantum eigensolver

VQE was originally proposed by Peruzzo et al. [2014]. In contrast to QAOA, the VQE *ansatz* is not designed specifically for each problem instance. There exist several pre-designed *ansatz* which are used independently of the optimization problem to be solved, and usually involve a larger number of parameters to be optimized. The parameters are tuned following the workflow defined in Figure 4.4. An *ansatz* example is the TwoLocal *ansatz*[1], shown in Figure 4.6.

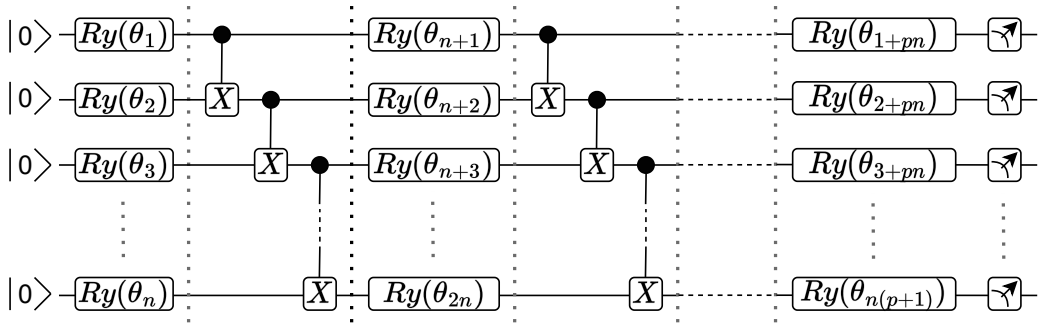


Figure 4.6: TwoLocal *ansatz* used for the VQE with $p \in \mathbb{N}$ layers, n qubits and pn parameters. Each layer applied a rotation gate in the Y-axis over each qubit. Between each layer, a linear entanglement is applied in the system, where each qubit i is connected through a two-qubit rotation gate in the X-axis to the qubit $i + 1$.

[1] https://qiskit.org/documentation/stubs/qiskit.circuit.library.TwoLocal.html

4.6 Quantum noise

Two main disadvantages of NISQ computers are their limited numbers of qubits and the presence of quantum noise. Thus, there is a need to implement approaches that offer resilience to quantum noise and to optimize the number of qubits used to solve the given problem. It has been shown that VQAs can compensate for quantum errors such as those in over-/under-rotations [McClean et al., 2016].

Following the formalism of Kraus operators [Nielsen and Chuang, 2002], each quantum channel ε is defined given a set of matrices E_i, which are applied to a quantum state, where i runs through all the operators considered for a given channel and k is the number of operators that define the noise channel. Thus, given a state ψ, the resulting state after applying a quantum channel is

$$\varepsilon(\psi) = \sum_{i=1}^{k} E_i \psi E_i^\dagger, \tag{4.19}$$

where the Kraus operators must meet the restriction $\sum_{i=1}^{k} E_i E_i^\dagger = 1$. Note that all the operators defined in this section are one-qubit operations ($E_i^{(1)}$), and that all the channels are parameterized by $\omega \in [0, 1]$, which regulates the probability of occurrence of the respective noise.

4.6.1 Amplitude damping error

The amplitude damping channel (ε_{AD}) describes the energy dissipation of a quantum system. This channel involves a parameter $\omega \in [0, 1]$ that tunes the probability of a quantum state to decay from state $|1\rangle$ to $|0\rangle$. Then, $\omega = 0$ represents no amplitude damping error, while $\omega = 1$ is the maximal noise, which in this case means the state $|0\rangle\langle 0|$.

This quantum operation describes the energy dissipation of the quantum states. The operation over a one-qubit system is defined as

$$\varepsilon_{AD}(\psi) = E_1 \psi E_1^\dagger + E_2 \psi E_2^\dagger,$$

where E_1 and E_2 are defined as

$$E_1^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\omega} \end{pmatrix}, \quad E_2^{(1)} = \begin{pmatrix} 0 & \sqrt{\omega} \\ 0 & 0 \end{pmatrix}.$$

The E_2 operation converts a $|1\rangle$ to a $|0\rangle$, representing the physical process of the environment energy lost. E_1 leaves $|0\rangle$ unaltered but decreases the amplitude of a $|1\rangle$ state. Physically, this occurs because some energy was not lost to the environment, and thus the quantum state is more likely to be in the $|0\rangle$ state, rather than in the $|1\rangle$ state.

4.6.2 Phase damping channel

The phase damping channel (ε_P) describes the loss of information of a quantum state without loss of energy. It can be due to the iteration of the system with the environment, which can cause random phase shifts in the quantum states. This channel involves a parameter $\omega \in [0, 1]$

that tunes the probability of a quantum state to lose information. Then, $\omega = 0$ represents no phase damping error, while $\omega = 1$ is the maximal noise, which in this case means any linear combination of $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$. The longitudinal relaxation time (T1) and the dephasing time (T2) are two quality metrics related to the amplitude and phase damping channels, respectively. After T1, the quantum behaviour is no longer that predictable, and T2 accounts for the phase loss of a qubit, after which the qubit can perform different phase rotations from those it was required to perform.

This quantum operation describes the loss of quantum information without loss of energy. The operation over a one-qubit system is defined as

$$\varepsilon_{PD}(\psi) = E_1 \psi E_1^\dagger + E_2 \psi E_2^\dagger,$$

where E_1 and E_2 are defined as

$$E_1^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\omega} \end{pmatrix}, \quad E_2^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\omega} \end{pmatrix}.$$

In this case, the E_1 operator acts in the same way as in the case of the amplitude damping noise channel, leaving $|0\rangle$ unchanged, but reducing the amplitude of $|1\rangle$. However, in this case, E_2 also reduces the amplitude of the $|1\rangle$ state, but does not change it to $|0\rangle$.

4.6.3 Depolarizing channel

The depolarizing channel (ε_D) describes the probability of a qubit to be depolarized. The depolarization of a qubit is the replacement of the current state with the mixed state $I/2$. This event occurs with probability $\omega \in [0, 1]$, and the qubits leave untouched with probability $1 - \omega$. Then, $\omega = 0$ represents no depolarizing error, while $\omega = 1$ is the maximal noise, which in this case means the mixed state $I/2$.

This quantum operation describes the depolarization of a qubit. That is, with certain probability ω a quantum state is replaced by the mixed state $I/2$. The operation over a one-qubit system is defined as

$$\varepsilon_D(\psi) = \omega \frac{I}{2} + (1 - \omega)\psi. \tag{4.20}$$

Despite the fact that Equation 4.20 does not involve any Kraus operators, it is possible to define the depolarizing channel with the following Kraus operators [Nielsen and Chuang, 2002],

$$\begin{aligned} E_1^{(1)} &= \sqrt{1 - \frac{3\omega}{4}} I_2, & E_2^{(1)} &= \sqrt{\frac{\omega}{4}} \sigma^x, \\ E_3^{(1)} &= \sqrt{\frac{\omega}{4}} \sigma^y, & E_4^{(1)} &= \sqrt{\frac{\omega}{4}} \sigma^z, \end{aligned} \tag{4.21}$$

where σ^x, σ^y and σ^z are the Pauli operators, and $\omega = 1$ implies the output state $\varepsilon_D(\psi)$ to be the mixed state $I/2$.

4.6.4 Multi-qubit quantum error

The application of the Kraus operators defined in Equation 4.21 for a two-qubits quantum system is defined as:

$$E_1^{(2)} = E_1^{(1)} \bigotimes E_1^{(1)}, \quad E_2^{(2)} = E_2^{(1)} \bigotimes E_2^{(1)},$$
$$E_3^{(2)} = E_3^{(1)} \bigotimes E_3^{(1)}, \quad E_4^{(2)} = E_4^{(1)} \bigotimes E_4^{(1)}.$$

4.6.5 Measurement error

Quantum measurement error regards to the precision error during the qubit measurement process. It is usually emulated as a white Gaussian noise ($\mathcal{N}(0, 1)$) over the samplings obtained from the quantum system.

Part III

CONTRIBUTIONS

Chapter 5

Industrial Problems Constrained by Environment Variables

5.1 Introduction

In this chapter we analyze the performance of traditional EDAs. Concretely, the implemented approach shares characteristics with EGNA variant, and introduces the *environment variables* concept, widely used in the following chapters, for restricting the search space during the optimization runtime.

The typical optimization problems that are usually referenced in the literature involve optimizing a cost function with a specific number of variables that are introduced to the algorithm in order to find the best solution. However, there are many real-world optimization problems in which variables that are not present in the cost function, influence the behaviour of the algorithm during the optimization process. Therefore, the total set of variables ($\boldsymbol{X}$) is decomposed into two subsets: those that are present in the cost function and can change their value during the optimization process ($\boldsymbol{Y} = \{Y_1, \ldots, Y_p\}$) which will be henceforth called as decision variables, and others, which are the inputs of the algorithm, and remain constant during the optimization ($\boldsymbol{Z} = \{Z_1, \ldots, Z_c\}$), henceforth called as *environment variables*.

$$\boldsymbol{X} = \{\boldsymbol{Y}, \boldsymbol{Z}\} = \{Y_1, \ldots, Y_p, Z_1, \ldots, Z_c\}$$

The relations within and between both subsets of variables must be considered in order to find coherent solutions to the problem.

In this chapter, we aim to solve one of these problems. The motivation for the problem comes from a real situation in an industry: a chemical process is carried out for the dissolution of a solid substance. Depending on the properties of the substance, the solvents to be used may change. Moreover, the final product after dissolving is stored in a tank and must comply with some further restrictions. Thus, the combination of solvents depends not only on the substance properties, but also on the final product restrictions. The industry aims to optimize the dissolution process and relies on a system to decide which is the optimal combination of solvents ($\boldsymbol{Y}$) for the specific *environment variables* ($\boldsymbol{Z}$): some substance properties and final

product restrictions. In this problem, optimization is evaluated from an economic point of view: each solvent is associated with a price, and it is desired to minimize the total cost. The optimization process uses the historic records of the laboratory technicians that dissolved the substance so far. Figure 5.1 shows a sketch of the process.

Depending on the properties of the substance and the restrictions imposed on the final tank, the solvent combinations vary, since the dependencies between the variables are different. If the dependencies between and within both subsets of variables are not considered, only the p variables involved in the cost function will be observed during the optimization process. Thus, the cheapest combination of solvents found by the algorithm will be to use 100% of the cheapest solvent and not using the others, as they would increase the cost. Optimizing only the solvent variables could be solved using a simpler optimizer, such as a gradient descent. The aim of this paper is to find the cheapest solvent combination, but keeping the patterns identified in the historical data between and within both subsets of variables. These patterns constrain the cost, and, for this reason, the implemented algorithm must identify the patterns in the initial data, and generate valid solutions according to them.

The cost function is then given by,

$$g(\boldsymbol{y}) = \sum_{i=1}^{p} y_i c_i \tag{5.1}$$

where y_i and c_i are the percentage of each solvent in the mixture ($\sum_{i=1}^{p} y_i = 1$) and its cost, respectively, and p is the total number of solvents. The optimizer should search for the optimum values of y_i such that the dependencies with the *environment variables* hold.

The problem consists of $p + c$ continuous variables: p solvents where each one specifies the percentage of the total mix of solvents, and c substance properties and restrictions. The substance properties specifically describe the substance, such as thermodynamic properties or chemical elements present in the substance, and some final product restrictions are for example the density, volatility or viscosity of the final liquid or a quality index calculated by the industry. Thus, the algorithm receives as input a total of c constant values $b_1, \ldots, b_c$ for $Z_1, \ldots, Z_c$, and must output the optimum combination of the p solvents. In this particular problem, the available historic records of how laboratory technicians have combined the solvents has a total of 1056 instances.

The application of most population-based algorithms and conventional solvers does not consider explicitly the relationship between the variables or does not allow the existence of a subset of constant inputs. In this chapter, a novel technique of EDAs is prepared to overcome this limitation. It includes a PM to identify conditional independence relationships between and within both $\boldsymbol{Y}$ and $\boldsymbol{Z}$ subsets of variables. The PM allows to set a subset of constant variables as observed evidence ($\boldsymbol{Z}$) of the model and then sample from the rest of the variables. Moreover, a new hyper-parameter is added to the implementation to control the influence of the historical data in the algorithm's decision making.

In this chapter, a PM learned in each generation of the algorithm builds an abstract representation of the relationships among the variables (solvents percentages, substance properties, and restrictions) of the promising solutions. In this representation, the patterns among variables

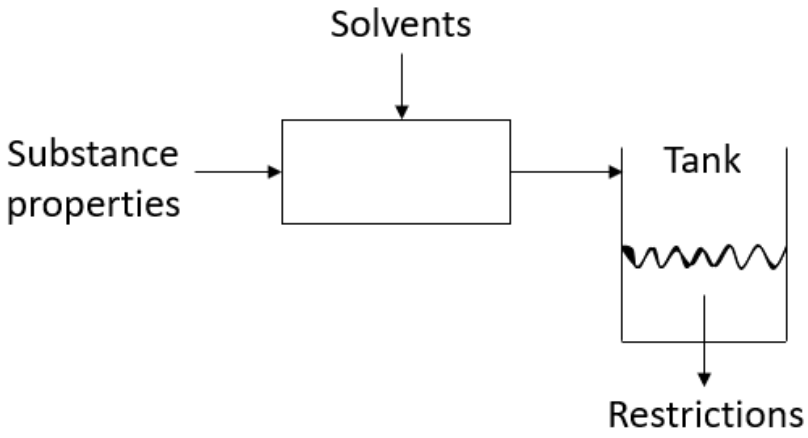


Figure 5.1: Sketch of the chemical process to optimize. The combination of solvents is influenced by the substance properties and the quality that the final product must attain.

can be identified in order to find better solutions. The use of PGMs provides the algorithm with the ability to deal with problems that involve variables with strong dependencies among them.

This chapter includes the developed methodology and results included in Soloviev et al. [2022b]. The dataset used for the evaluation is not publicly available due to confidentiality, although the implemented methodology is available in `EDAspy` library[1].

Chapter outline

The chapter outline is organized as follows. Section 5.2 details the proposed methodology. Section 5.3 presents some results of the evolution of the optimization process and the behaviour of the algorithm for different problem instances and different parameter tuning. Section 5.4 rounds the paper off with the conclusions and future work.

5.2 Proposed solution

As mentioned above, an EDA using a GBN is used to solve the optimization problem. The pseudocode in Algorithm 5 shows how the proposed EDA is implemented.

In the pseudocode, the main functions of the algorithm are shown. Initially, G_0 is an input to the code. Then, in each iteration, the algorithm selects the top individuals of the generation considering the cost function. The algorithm learns a GBN, a multivariate Gaussian distribution is then derived from the GBN, and new individuals are sampled from joint distribution of it. Each set of new individuals constitutes a generation that is the input for the next iteration of the algorithm. The stopping criteria may be based on a maximum number of iterations, a state in which no better solutions are found by the algorithm, or a state in which the algorithm is not able to generate valid individuals. In this implementation

[1]https://github.com/VicentePerezSoloviev/EDAspy

Algorithm 5 EDA

1: $G_0 \leftarrow N$ individuals from the historic records
2: $G_{aux} \leftarrow$ Select similar situation individuals from the historic records
3: **for** $t = 1, 2, ...$ until stopping criterion is met **do**
4: $G_{t-1} \leftarrow$ Select E individuals from G_{t-1}
5: $G_{t-1} \leftarrow$ Append G_{aux} to G_{t-1}
6: $GBN_t \leftarrow$ Learn GBN from G_{t-1}
7: $f(\boldsymbol{X}) \leftarrow$ Compute the joint density function from GBN_t
8: $G_t \leftarrow$ Sample N individuals from $f(\boldsymbol{X})$
9: **end for**

the stopping criterion is a number of iterations after which the algorithm cannot improve the best global cost. The main functions used in the algorithm are described below.

The GBN learned in each generation is an abstract representation of the region of the search space explored, which guides the algorithm to promising areas. The individuals sampled from the GBN keep the patterns identified in the data used to learn the multivariate model. The use of GBNs allows generating solutions that were not contemplated in the learning of the GBN.

The use of some algorithms is restricted to the amount of data availability. For example, using neural networks demands a large amount of data to be trained. However, building a BN does not require such a large amount of data. The EDA can be run correctly with the 1056 instances we dispose of for the initialization step. Another advantage of the use of GBNs is that the DAG can be visualized in each generation in order to show patterns in data, or to easily analyse the algorithm behaviour.

5.2.1 Initialization

The initialization of the algorithm has a large influence on how the algorithm moves through the search space during execution. The aim of this chapter is to implement an algorithm which finds the optimum solution of the problem keeping the patterns identified in the data. A random initialization would generate some random patterns that would be kept through iterations and this is not desired. This could be addressed implementing a *whitelist* in the GBN in order to fix the arcs that are desired in the generated samples, but this would over-bias the algorithm, as exploration would be limited.

In our approach, the initialization is based on the historic records of how technicians of the laboratory have been manipulating the solvents. In the chemical laboratory, different sensors record values of the variables used in the problem. First, some sensors measure the substance to be dissolved providing the properties variables. Second, the tank where the final product is to be stored has some further restrictions that must be accomplished, and are also sensorized. Also, the amount of each solvent introduced in the solvent mixture is recorded. These values are the historic records. We assume that the staff has been manipulating the solvents to obtain the best results, considering the restrictions and the final cost. As these historic records are considered nearly optimum, the algorithm is expected to converge soon.

5.2.2 Truncation

When a generation is obtained in each iteration, the individuals whose values in the subset $\boldsymbol{Y}$ of variables minimizes the cost function, are interesting and must therefore be selected. Once they are selected, all the individuals chosen have the same relevance regardless of the cost of the individual. Moreover, those selected individuals may have a very low probability of occurrence in the historic records. This can lead the algorithm to learn confusing patterns, and future generations will be sampled incorrectly. For this reason, two modifications are made in the baseline of the algorithm.

First, the cost is added as a new node of the GBN. By adding this new variable, a certain coherence is ensured among all the samplings. If the cost is considered as a new variable, the majority of the individuals in a generation will be sampled around the mean cost according to the shape of the normal distributions, and thus, anomalous individuals are less likely to be obtained when sampling. This does not avoid anomalous individuals to be sampled, because a small part of the samples will be in the tails of the normal distributions, but maximum dispersion among individuals is reduced. This new variable is calculated with the cost function given in Equation 5.1. In this way, the GBNs learned in each iteration are more coherent. Other multi-objective EDAs has also added the cost as nodes of the BN, such as [Karshenas et al., 2013].

Second, the likelihood of each individual in the historic records is considered. If only the individuals that minimize the cost function were selected, it would be possible that very anomalous individuals would be selected, as individuals from different promising spaces, even located at the tails of the distribution, may be found. To avoid this, both, the cost of the individual and the likelihood of the individual in the historic records must be considered in the truncation ranking. An individual that costs very cheap is very attractive, but if its likelihood is low in the historic records, it might not be worth choosing. Therefore, those individuals whose cost is not as cheap as the so attractive ones of the generation, but their likelihood is higher, are preferable. For this reason, the likelihood in the historic records must be considered.

The ideal is to find a balance between the likelihood in the historic and the randomness of the samples of the individuals in the multivariate model. Consequently, a new hyper-parameter α is added to the implementation, which controls the influence of the likelihood in minimizing the cost function. We define the parameter as a value in the range $[0, 1]$. If the likelihood is not taken into account ($\alpha = 0$), the solutions can be dispersed, and different executions of the EDA can provide very diverse solutions that may be very cheap. However, if the likelihood is totally considered ($\alpha = 1$), the optimizer can nearly not optimize the solutions in the historic records, as the EDA will try to find a similar solution to those that can be found in the historic records, and not a solution based on the patterns found in the data. An analysis of this parameter is provided below. Thus, the function used in the truncation is,

$$C_\alpha(\boldsymbol{x}) = \frac{\sum_{i=1}^{p} y_i c_i}{\sum_{i=1}^{p} c_i} - \alpha\, p(\boldsymbol{x}|\boldsymbol{\Theta}_0, \mathcal{G}_0), \tag{5.2}$$

where $p(\boldsymbol{x}|\boldsymbol{\Theta}, \mathcal{G})$ is the likelihood (Equation 2.6) of an individual $\boldsymbol{x}$ in the Gaussian multivariate model estimated from the initial generation (the historic records) represented by a DAG

$\mathcal{G}_0$ and some parameters $\boldsymbol{\Theta_0}$; and y_i and c_i are the solvent amounts percentages and costs, respectively. The first term of the equation has been normalized so that a comparison with the second term is possible, as likelihood is represented as a probability. For $\alpha \to 1$ the historic relevance tends to have the same relevance as the cost of the individual. For $\alpha = 1$ the cost and the historic influence have the same relevance in Equation 5.2.

It is expected that the likelihood of the individuals of each generation will increase. Because of this, Equation 5.2 is not the cost function that should be minimized. It is only a function used to evaluate the individuals in the truncation step, and establish a ranking, to guide the selection of the top individuals.

With these two modifications it may happen that some very cheap individuals are rejected from a generation due to their anomalous costs in comparison with those of the rest of the population. In this case, these rejected individuals will appear in future generations, where their likelihoods and costs are more consistent with those of the other individuals in the population. This favours the building of coherent GBNs in each iteration, as more coherence among individuals is obtained.

5.2.3 Problem formulation

The mathematical formulation of the problem is presented in this section. The problem involves $p + c$ variables, $\boldsymbol{X} = \{\boldsymbol{Y}, \boldsymbol{Z}\} = \{y_1, \ldots, y_p, z_1, \ldots, z_c\}$, where $\boldsymbol{Y}$ are the decision variables and $\boldsymbol{Z}$ are the *environment variables* fixed as constant.

The optimization problem is,

$$\text{minimize} \qquad \min_{\boldsymbol{x}} \left(\frac{\sum_{i=1}^{p} y_i c_i}{\sum_{i=1}^{p} c_i} - \alpha \, p(\boldsymbol{x}|\boldsymbol{\Theta}_0, \mathcal{G}_0) \right) \tag{5.3}$$

$$\text{subject to} \qquad g_i(\boldsymbol{x}) = b_i, \text{ for } i = 1, \ldots, c, \tag{5.4}$$

where $c_i \in \mathbb{R}^+$ are constant terms predefined in the problem by $\boldsymbol{C} = \{c_1, \ldots, c_p\}$, the parameter $\alpha \in [0, 1]$, and $\sum_{i=1}^{p} y_i = 1$.

Here, the vector $\boldsymbol{x} = (y_1, \ldots, y_p, z_1, \ldots, z_c)$ is the optimization variable of the problem, the function $C_\alpha(\boldsymbol{x}) : \boldsymbol{R}^{p+c} \to \boldsymbol{R}$ is the objective function defined in Equation 5.2, the functions $g_i : \boldsymbol{R}^c \to \boldsymbol{R}, i = 1, \ldots, c$ are the constraint functions and the constants $b_1, \ldots, b_c$ are the constraint values that the constraint functions should meet. A vector $\boldsymbol{x}^*$ is optimal if it has the lowest objective value among all vectors that satisfy the constraints. Thus, for any $\boldsymbol{x}$ with $g_1(\boldsymbol{x}) = b_1, \ldots, g_c(\boldsymbol{x}) = b_c$ we have $C_\alpha(\boldsymbol{x}) \geq C_\alpha(\boldsymbol{x}^*)$.

Despite the fact that the cost function is defined in Equation 5.1 in terms of the decision variables y_i, the *environment variables* $\boldsymbol{Z}$ in the problem statement must be considered using Equation 5.2 during the optimization process. However, as this approach has an industrial perspective, the results shown in Section 5.3 represent the economic cost calculated with Equation 5.1 for each solution $\boldsymbol{x}$ provided by the algorithm.

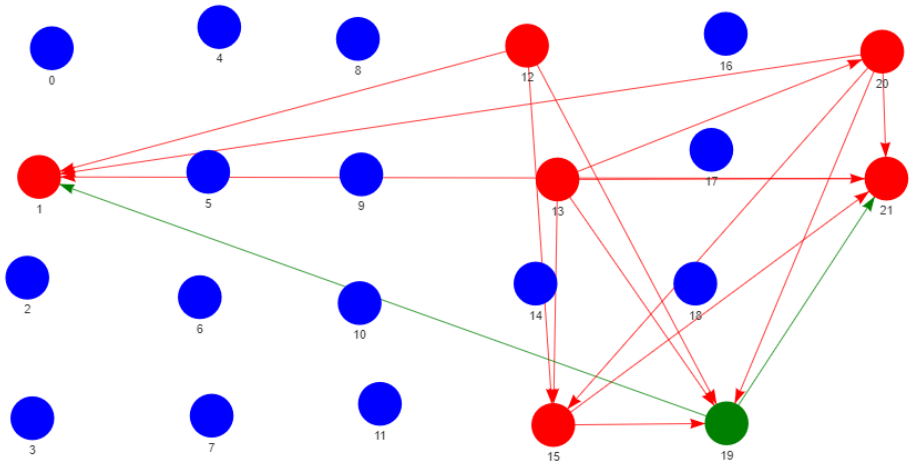


Figure 5.2: DAG learned in an EDA execution where the input variables are fixed as evidence of the GBN. Only the y_i variables (red nodes) are connected in the DAG, and the rest are independent. The blue nodes represent the *environment variables* and the green node is the cost.

5.2.4 Estimation of the density function

When the top individuals are selected using Equation 5.2, the structure of the GBN can be learned. As the only variables that the EDA can optimize are the subset $\boldsymbol{Y}$ of variables, the *environment variables* are inputs of the problem and hence, evidences of the multivariate Gaussian distribution. Thus, in each generation the subset of *environment variables* will assign to each individual the same value. When the learning algorithm tries to find dependencies among variables, all the fixed *environment variables* are found to be independent, and if some dependence is found it would be spurious. This means that the variables included in the subset $\boldsymbol{Y}$ of variables would only influence and be influenced by each other, without considering the subset $\boldsymbol{Z}$ of variables. This would be a situation in which the algorithm finds the optimum combination of solvent amounts, without considering the rest of the variables: $\boldsymbol{Y} \perp \boldsymbol{Z}$. Figure 5.2 shows this type of situation where the red nodes represent the subset $\boldsymbol{Y}$ of variables, the green node represents the cost function, and the blue nodes are the *environment variables* $\boldsymbol{Z}$.

The patterns identified in the historic records must be translated to the GBN learned from the top individuals of each generation. Therefore, some individuals from the historic records are added to the selection made in the truncation phase of the algorithm. These added individuals G_{aux} (see Algorithm 5) are selected from the historic records in such a way that individuals similar to the problem situation are selected, that is, similar *environment variables*. If a random set of individuals is added from the historic records, a lot of dispersion may be introduced, or solutions from different zones of the search space may be selected, and hence, confusing patterns could be learned by the algorithm. For this reason, clustering is implemented in the historic records considering the subset of *environment variables*. In this way, G_{aux} is incorporated into the selection of the E (see Algorithm 5) individuals made in the truncation. G_{aux} will provide to the E selection of individuals, the patterns of the search

space explored by the algorithm in order to identify the dependencies among the variables.

Once G_{aux} is added to the selection made in the truncation phase, the GBN can be learned. The score-based hill climbing algorithm is used. The subset of *environment variables* are the inputs of the problem, and thus, evidences of the GBN. If z are the input values that the subset $\boldsymbol{Z}$ of variables take, the multivariate Gaussian distribution built is $f(Y|Z = z)$ with the evidences as fixed values in the multivariate Gaussian distribution. Expert knowledge is added by specifying the *black list* and *white list*.

When the optimal combination of y_i is found, an optimal structure of the GBN is learned for specific substance properties and further restrictions. Thus, this GBN can be used to perform inferences and analyze different combinations of y_i. The structure can be also used by experts to infer patterns among variables of the system. By introducing the cost as a node of the GBN, it can be used by the experts to calculate some posterior probabilities of the cost $\boldsymbol{C}$ given some variables values $x_1, \ldots, x_t$, for example, $f(C|x_1, \ldots, x_t)$ for classification tasks.

5.2.5 Sampling

Once the GBN is built, the next generation can be sampled. It is expected that the dispersion among the individuals in the initial iterations is higher than in later iterations. The behaviour of the algorithm is designed to make the space search move towards the optimal space and to reduce the dispersion. The more iterations of the algorithm, the lower dispersion between individuals, until convergence of the algorithm is reached and the dispersion among individuals is minimum. Thus, the individuals in the last iterations should be similar and centered in the optimal solution found.

Despite the fact that the multivariate Gaussian distribution is estimated from real data, it is possible that non-real data are sampled. These samplings must be discarded. Otherwise, the algorithm will tend towards a minus infinite cost.

Depending on the input values the number of samplings removed may be different. In some situations, the optimizer may find solutions in which some of the y_i variables are reduced to nearly zero, so due to the Gaussian distribution, some of the samplings will lead to solutions with values less than zero. However, other input variables will make the optimizer stabilize the amounts at a realistic percentage, and the number of removed samplings will be smaller. The frequency of appearance of these invalid individuals is small, so it does not imply a significant computational cost.

5.3 Results

As the system was implemented to optimize a process in an industry, the model was run considering expert knowledge. The optimizer results were validated with expert technicians of the chemical laboratory.

In the problem we aim to solve in this chapter, the substance properties and the final liquid restrictions in the tank are given as inputs (*environment variables*), and the optimal combination of the six solvent amounts are returned as output. In this section, some real

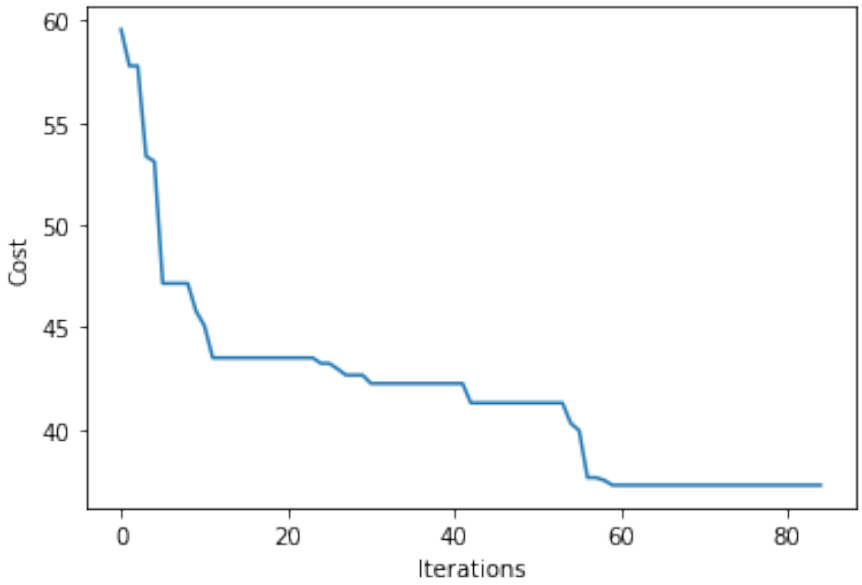


Figure 5.3: Best cost (Equation 5.1) found in each iteration of an EDA execution for a specific substance with restrictions.

problem results are provided. To validate the optimizer, real historic records situations have been selected. The values of the subset of *environment variables* are selected and the process and final results of the optimizer are analyzed by expert knowledge in the field. Expert knowledge is of great importance in this validation process as it must corroborate that the existing relationships that are identified between the data are correct.

Figure 5.3 shows the best cost evolution with an increasing number of iterations. In each iteration, note that the algorithm tries to converge towards a better solution than it has so far. When the algorithm cannot improve the best solution found after an specific number of iterations, the algorithm converges and returns the best solution found. When the curve flattens, the EDA has found the optimal area of the initial search space (iteration 60). One of the advantages of this algorithm is that not only a single solution can be returned. The algorithm can return a set of solutions with similar costs, in order to select one preferable by laboratory technicians.

The expected behaviour of the algorithm evolution is that the dispersion among individuals is reduced as the number of iterations increases. Figure 5.4 shows the evolution of the mean cost and the dispersion among individuals along an EDA execution. The dispersion does not have to decrease in a linear way, but a decreasing trend is appreciated in its evolution. The way to measure this dispersion is the distance from the mean cost of the generation.

The ability to return a set of optimal solutions to the problem is not the only notable advantage of this algorithm. Another one is that once the algorithm finds the optimal area of the search space, it is possible to save the GBN of this area as the optimal structure for the concrete problem. This GBN can be used with different purposes. The GBN found for an EDA execution is shown in Figure 5.5, which is a representation of the optimal space that the EDA found in the initial search space. Note the difference between both learned structures in

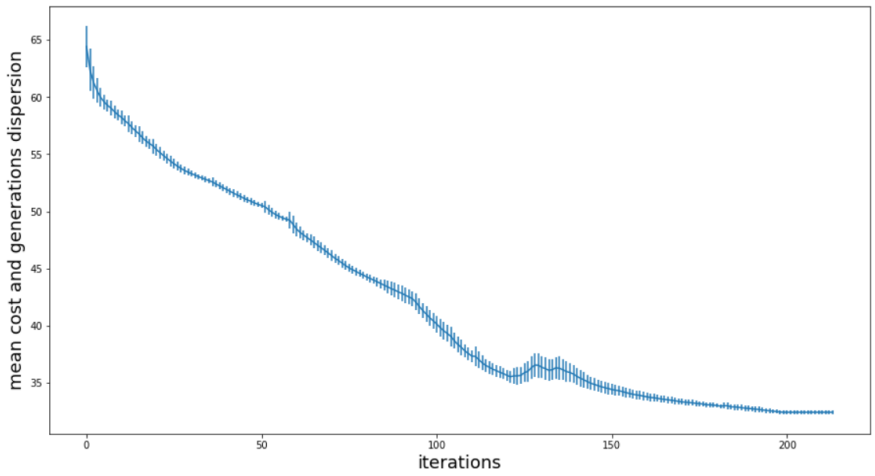


Figure 5.4: Mean cost (Equation 5.1) and dispersion among individuals in each generation for a specific substance with restrictions.

Figure 5.2 and Figure 5.5. Figure 5.2 shows a situation in which *environment variables* ($\boldsymbol{Z}$) are independent nodes and only the solvent variables ($\boldsymbol{Y}$) are influenced by each other as they are continuously being optimized during the process: $\boldsymbol{Y} \perp \boldsymbol{Z}$. However, this is solved by adding G_{aux} in each generation of the algorithm (see Algorithm 5). This way, we ensure that dependencies between both $\boldsymbol{Y}$ and $\boldsymbol{Z}$ subsets of variables are considered and the algorithm reaches a realistic situation as shown in Figure 5.5, where *environment variables* influence the decisions in the solvent variables.

Notions of expert knowledge in the field will not be explained, but different aspects of the DAG shown in Figure 5.5 must be analyzed. In a BN a variable depends on those in its MB. All y_i variables are dependent on each other. This relation is obvious as are percentages of a total amount of solvent mix. If some of the solvents amount are reduced, others must compensate this reduction ($\sum_{i=1}^{p} y_i = 1$). Thus, all the solvents are dependent on each other. The cost node depends on all the solvents, and other *environment variables*. The modification of some of the solvent amount is directly related to the cost variation. Other relations can be appreciated, such as some *environment variables* which are directly related to the cost variable. This GBN can be used to perform different inferences, to try different combinations of solvents amounts, and infer the most probable solvent combination always keeping in mind that the GBN represents an optimal area in the initial search space.

Figure 5.6 shows the value of the cost function C as a function of the α parameter (Equation 5.2). For the same problem (same inputs), the EDA was run 20 times for each value of α. C_α (Equation 5.2) was used to perform the truncation, and find the minimum cost function (Equation 5.1). Note that the mean C value increases with α while its dispersion among different solutions decreases. As $\alpha \to 1$, C tends toward the value determined by expert knowledge ($C \approx 58.5$). However, although the dispersion increases for $\alpha = 0$, for all the simulations the optimal costs found is below that determined by theoretical expertise. The figure also shows that the cost converges to a constant value (dashed line) for $\alpha > 0.5$, which represents the cost based on expert knowledge.

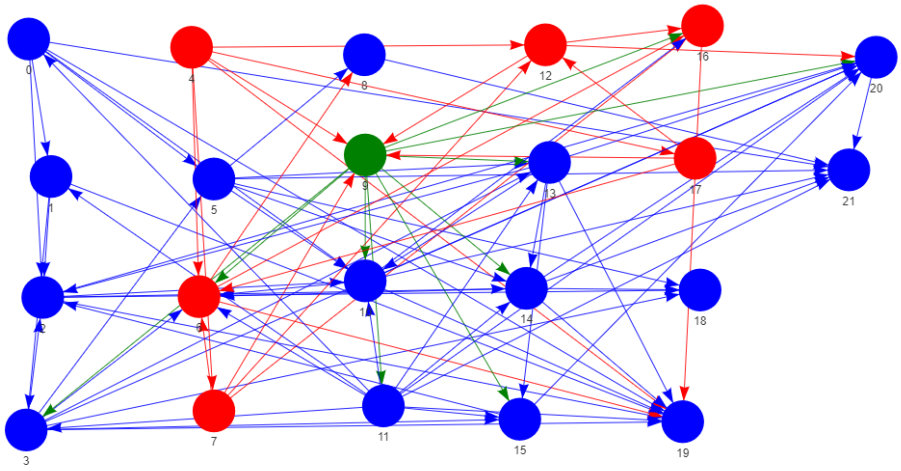


Figure 5.5: DAG of the GBN resulting from an EDA execution. The optimal GBN is obtained for specific substance properties and restrictions. No nodes are independent. The blue nodes represent the substance properties and restrictions, the green node is the cost, and the red nodes are the solvent amounts.

For $\alpha > 0.5$, the optimizer obtained similar costs and small dispersion among solutions. In Equation 5.2 the cost of the solvent combination tends to have the same relevance as the likelihood in the historic records. Thus, the algorithm will not provide solutions that do not fit well with the historic records. The solution provided by the algorithm will be the same or nearly the same as that of the historic records, as the laboratory records are assumed to be nearly optimum. For highest values of α the optimizer performs as a predictive model, as the solutions provided are the ones that can be found in the historic records.

For low values of the α parameter, the solutions provided have a larger dispersion. The algorithm is therefore more stochastic than it is for high values of the α parameter. For each iteration, only the cost of the solvent combination is considered, and thus, the individuals that have a lower density in the historic Gaussian multivariate data will not be removed and can lead to new solutions not contemplated. As the likelihood is nearly not considered, the solutions provided by the algorithm are more stochastic, and thus, in different executions, different GBNs can be learned. This explains the dispersion among the different executions shown in Figure 5.6 for low values of α.

Facing an industrial application, it must be considered the α hyper-parameter tunning. Despite the fact that the theoretical basis on which the implementation of the algorithm is based is correct, using those solutions found by the algorithm for α close to zero may carry a risk; the algorithm may have learned wrong patterns during runtime so not reliable solutions may have been found, or more constraints should be added to translate the *savoir-faire* of the technicians.

Thus, it is necessary to find a balance between the stochasticity of the algorithm and the solutions already found in the historic records. This balance must be found using expert knowledge, or even carrying out virtual simulations of the chemical process to properly tune

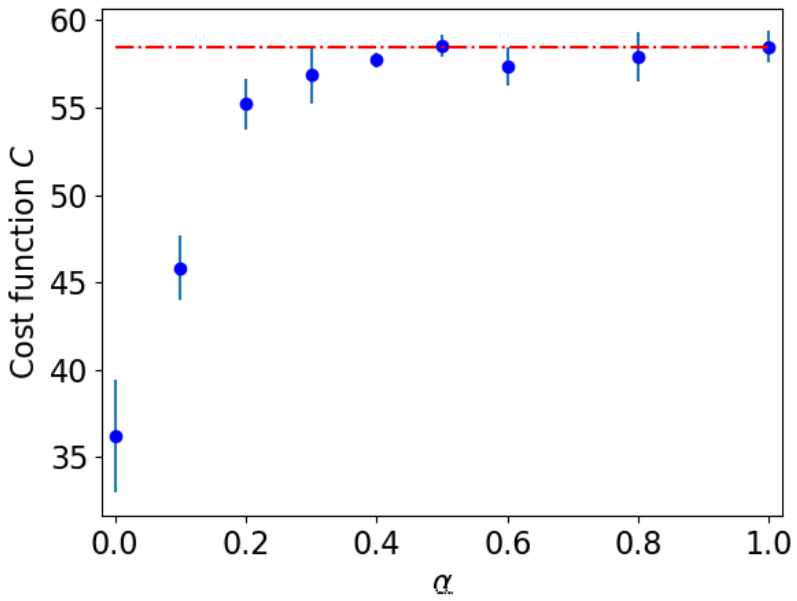


Figure 5.6: Mean cost as a function of the α parameter (Equation 5.2). For each value of α, 20 simulations were run. The dashed line corresponds to the cost of solvents traditionally used in the laboratory processes for this specific substance and set of restrictions.

this novel hyper-parameter. The new hyper-parameter can be discussed from the exploitation-exploration point of view. For high values of the parameter, the exploration of the search is minimum while the exploitation of the already existent solutions in the historic data is maximum. Though, low values of the parameter leads to explore the search space.

In order to perform a deeper analysis, the proposed approach has been executed for different instances of the problem. We have designed four experiments in which the substance to be dissolved is the same, but different *environment variables* are present.

- Experiment 1. The viscosity index of the dissolution result has a value of 10% below the mean found in the historical data.
- Experiment 2. The viscosity index of the dissolution result has a value of 10% above the mean found in the historical data.
- Experiment 3. The volatility index of the dissolution result and the quality index defined by the industry are decreased a 10% below the mean found in the historical data.
- Experiment 4. The volatility index of the dissolution result and the quality index defined by the industry are increased a 10% above the mean found in the historical data.

The experiments have been executed for EMNA, the Particle Swarm Optimization (PSO) [Kennedy and Eberhart, 1995] and our approach. The EMNA approach has a similar pseudo-code as our approach but instead of learning a Gaussian Bayesian network in each iteration of the evolutionary algorithm, it learns a multivariate Gaussian distribution from where it is

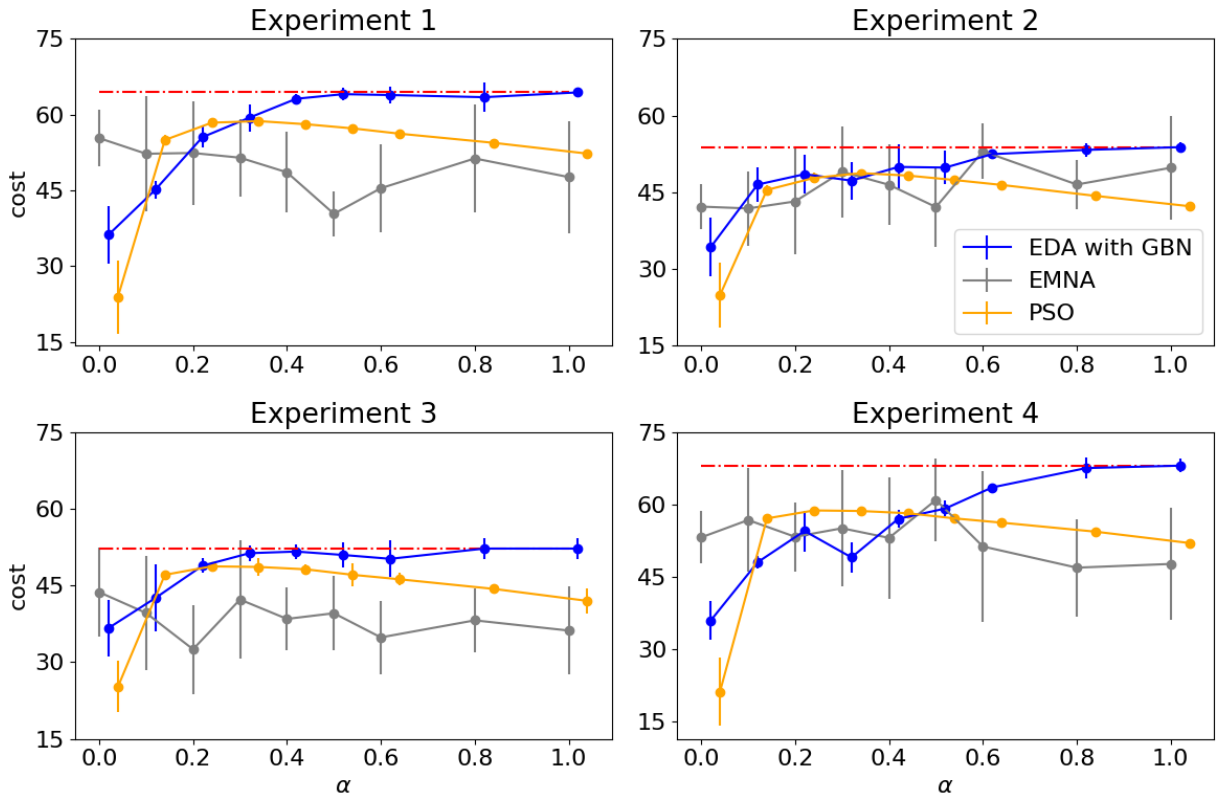


Figure 5.7: Mean cost and standard deviation of EDA with a Gaussian Bayesian network, and the EMNA and PSO algorithms as a function of the α parameter (Equation 5.2) for the different experiments. For each value of α, 20 simulations were run. The dashed line corresponds to the cost of solvents traditionally used in the laboratory processes for this specific substance and set of restrictions.

sampled (lines 6-7 of Algorithm 5). PSO in each iteration updates its parameters in order to minimize the cost function but not considering dependencies among variables. This way, we compare the difference between using Gaussian Bayesian networks and not using them during the optimization process.

The results are shown in Figure 5.7 and in Table 5.1, Table 5.2 and Table 5.3. If we analyze each of the experiments individually, it can be seen that the behavior of our approach is as expected. For low values of α it finds very cheap solutions, and the cost increases with α, until the cost settles to the value determined by expert knowledge for $\alpha \to 1$, shaping a curve with a logarithmic profile. However, the behaviour of the EMNA and PSO is not as predictable as our approach for different values of α. For $\alpha \to 0$ our approach seems to find better solutions than EMNA. However, for $\alpha \to 1$, EMNA and PSO find better solutions than our approach, but always very far from the costs provided by the experts and with high values of standard deviation. EMNA and PSO are not learning the patterns found in data and thus, they are converging to solutions that are cheaper but are not of interest for the company when fully considering the likelihood of solutions in the historical data (Equation 5.2). Facing

α	Experiment 1	Experiment 2	Experiment 3	Experiment 4
0.0	36.22 ± 5.66	34.27 ± 5.76	36.67 ± 5.55	37.46 ± 3.99
0.1	45.17 ± 1.77	46.52 ± 3.45	42.55 ± 6.50	48.03 ± 1.23
0.2	55.54 ± 2.01	48.55 ± 3.72	48.84 ± 1.46	52.26 ± 4.26
0.3	59.40 ± 2.70	47.29 ± 3.68	51.29 ± 1.57	50.33 ± 3.22
0.4	63.14 ± 0.85	49.97 ± 4.42	51.61 ± 1.41	56.40 ± 1.95
0.5	64.05 ± 1.24	49.84 ± 3.26	50.95 ± 2.50	59.49 ± 1.58
0.6	63.87 ± 1.71	52.47 ± 0.77	50.20 ± 3.54	63.50 ± 0.71
0.8	63.45 ± 2.97	53.30 ± 1.34	52.20 ± 2.12	67.50 ± 2.12
1.0	64.40 ± 0.88	53.84 ± 0.91	51.18 ± 2.12	68.00 ± 1.41

Table 5.1: Mean and standard deviation for each of the four designed experiments and different values of α executed by our approach (Figure 5.7).

α	Experiment 1	Experiment 2	Experiment 3	Experiment 4
0.0	55.35 ± 5.590	42.22 ± 4.360	43.68 ± 8.727	53.15 ± 5.432
0.1	52.25 ± 11.46	41.86 ± 7.251	39.61 ± 11.08	56.74 ± 10.74
0.2	52.41 ± 10.25	43.23 ± 10.24	32.52 ± 8.702	53.26 ± 7.166
0.3	51.48 ± 7.700	49.01 ± 8.911	42.24 ± 11.57	55.03 ± 12.01
0.4	48.58 ± 7.990	46.48 ± 7.833	38.46 ± 6.135	53.04 ± 12.53
0.5	40.37 ± 4.380	42.13 ± 7.845	39.56 ± 7.312	60.90 ± 8.631
0.6	45.43 ± 8.640	53.03 ± 5.448	34.84 ± 7.151	51.26 ± 15.58
0.8	51.33 ± 10.69	46.53 ± 4.821	38.17 ± 6.170	46.87 ± 10.06
1.0	47.64 ± 11.07	49.80 ± 10.22	36.18 ± 8.651	47.64 ± 11.62

Table 5.2: Mean and standard deviation for each of the four designed experiments and different values of α executed by the EMNA approach (Figure 5.7).

an industrial solution, when our approach considers the likelihood completely, the laboratory technicians will obtain solutions which they are accustomed to, and as α is decreased, solutions that are somewhat more risky and therefore cheaper will be obtained, but they still take into account the patterns found in the data. It is interesting how the results obtained by the PSO approach when the likelihood is not considered ($\alpha = 0$) outperforms the ones obtained by EMNA and by our approach, and the small standard deviations observed in Table 5.3 for $\alpha > 0$. From this, we conjecture that PSO is falling in local optima solutions which seems to have been avoided by EMNA and our approach.

The set of experiments shows how in general experiments 2 and 3 are cheaper than the others, which makes sense because in both experiments we lead to solutions in which the final product is closer to a solid than to a liquid, and therefore it is not necessary to dissolve the substance as much as in experiments 1 and 4.

Table 5.4 shows a quantitative comparison of the EMNA, PSO and our approach analyzing the number of fitness evaluations and computation time until convergence. All the approaches have been evaluated under the same conditions. We can observe how our approach takes the larger runtime compared to the two other algorithms. This is due to the GBN learning in each

α	Experiment 1	Experiment 2	Experiment 3	Experiment 4
0.0	36.22 ± 5.66	34.27 ± 5.76	36.67 ± 5.55	35.89 ± 3.99
0.1	45.17 ± 1.77	46.52 ± 3.45	42.55 ± 6.50	47.98 ± 1.23
0.2	55.54 ± 2.01	48.55 ± 3.72	48.84 ± 1.46	54.47 ± 4.26
0.3	59.40 ± 2.70	47.29 ± 3.68	51.29 ± 1.57	49.00 ± 3.22
0.4	63.14 ± 0.85	49.97 ± 4.42	51.61 ± 1.41	57.03 ± 1.95
0.5	64.05 ± 1.24	49.84 ± 3.26	50.95 ± 2.50	59.12 ± 1.58
0.6	63.87 ± 1.71	52.47 ± 0.77	50.18 ± 3.54	63.52 ± 0.71
0.8	63.45 ± 2.97	53.31 ± 1.34	52.24 ± 2.11	67.45 ± 2.15
1.0	64.40 ± 0.88	53.84 ± 0.91	51.98 ± 2.08	68.02 ± 1.41

Table 5.3: Mean and standard deviation for each of the four designed experiments and different values of α executed by the PSO approach (Figure 5.7).

Algorithm	# iterations	CPU time
EDA with GBN	1 ± 1	1 ± 1
EMNA	1.12 ± 4.90	0.33 ± 2.98
PSO	1.43 ± 0.37	0.39 ± 0.20

Table 5.4: Mean and standard deviation of the number of iterations and runtime until convergence for our approach, EMNA and PSO after 20 simulations. This experiment has been carried out with population size $N = 100$ and $\alpha = 0.6$.The number of fitness evaluations would be NQ, where Q is the number of iterations until convergence. All the results have been normalized taking as reference the results of our approach. The experiment was conducted on a Windows 10 machine with an Intel i7-5820K processor and 16 GB of RAM.

iteration, which is a characteristic that makes our algorithm to take more than double of the runtime. Our approach seems to need a smaller number of fitness evaluations until convergence compared to its competitors. However, the main advantage of our approach is that we are able to obtain solutions that can approximate those given by experts, parameterized by α, while EMNA and PSO are not able.

Our approach seems to be a good option in those industrial situations where it is crucial to provide similar solutions to the ones offered by the experts ($\alpha \to 1$) but also when cheaper solutions are sought ($\alpha \to 0$) under relaxed computation time requisites.

5.4 Conclusions

In summary, the aim of this chapter was to propose a solution for those optimization problems in which there exist two types of subsets of variables: a subset of variables that define the cost function and can be optimized, and a subset of fixed variables that are the input of the problem. The dependencies between and within both subsets must be considered in order to keep the patterns observed in data. To solve this, an EDA is used which incorporates a GBN that is rebuilt in each iteration to find the best model for the optimum area of the search space.

As the algorithm is initialized based on the historic records, a novel hyper-parameter α was introduced to control the influence of the historic conditions on the individuals that constitute each generation. Our results show that cheaper solutions can be obtained when the algorithm is not constrained by the historic conditions ($\alpha = 0$). In this case, the dispersion of the solutions among different EDA executions is higher, but the worst solution for $\alpha = 0$ is cheaper than the solutions in which historic records are considered ($\alpha > 0$). This way, this hyper-parameter must be tuned. Our approach is compared with the EMNA and PSO algorithms in which no GBNs are used to identify the relationships among variables in each iteration. The results show that our approach can approximate the solution given by experts when $\alpha \to 1$ unlike EMNA or PSO.

Future work to this chapter is listed below:

- The EGNA approach is assuming that the given data fit Gaussian distributions. Future work should consider avoiding this limitation.
- This chapter is focused on optimization tasks in which only one objective is considered. Future implementations should extend this work to multi-objective tasks in the industry.
- The hyper-parameter tunes the influence of the historic data in the decisions made during runtime by comparing the individuals likelihood. A different heuristic could be applied, such as comparing the individuals likelihood in the previous generations, in such a way that, the decisions made during runtime would be influenced by the previous generations, and not only by the historical data.
- Future work also includes adapting this approach to dynamic environments in which *environment variables* or the cost function may vary during runtime.

Chapter 6

A Multi-objective Framework for Data-Driven Experimental Design

6.1 Introduction

Chapter 5 assesses the performance of EGNA approach for a real application in the Industry 4.0. In this chapter we extend the previous methodology for multi-objective optimization tasks in a real application in the industry, as suggested in the conclusions and future work section in Section 5. The approach is embedded into a bigger framework which explodes the capabilities and interpretability of the model, and is applied for the design of chemical experiments.

The United Nations' global goal of net zero emissions by 2050 [Deutch, 2020, Seto et al., 2021] is demanding an increase in the production of high-performance fuels from sustainable feed-stocks in order to reduce dependence on petroleum-based products and mitigate the environment footprint. Considering the small time-window available to achieve the net-zero emission goal, the design of a new generation of fuels that can improve sustainability by reducing carbon dioxide emissions, can benefit from the speed-up provided by an holistic approach that integrates artificial intelligence and laboratory experiments.

Fuel design is a black-box optimization task, where the result of the target properties of a proposed fuel formulation are most of the time unknown until experiments are performed, and can seldom be estimated (predicted or analytically imputed, depending on the problem). Thus, smart fuel design relies on methodologies that propose promising formulations, and analyze past experiments to improve future proposals, minimizing the number of experiments in the real laboratory.

Traditional experimental design methods involve modelling and sampling with the aim of exploring unknown regions of the design landscape. Some examples are factorial design [Jankovic et al., 2021], in which linearity between variables is assumed and samplings are located at the orthogonal corners of the landscape. There are more complex methods such as Latin hyper-cubes sampling with general purpose space-filling designs [Viana, 2013, Vieira Jr et al., 2013], and response surface methodologies [Myers et al., 2016], where polynomials are

used to model the search space. However, the main drawback of these approaches is that once sampled, most of the solutions are unnecessarily evaluated, and the knowledge extracted from the promising ones is not incorporated into future samplings, as these are not iterative processes.

Data-driven design of experiments (DoE) is a powerful approach that leverages statistical and machine learning techniques to optimize the design and execution of experiments. By leveraging previous data provided by experts and computational models, data-driven DoE aims at identifying the most informative and efficient experimental configurations, thus minimizing the number of experiments required to obtain a fuel with the desired properties while maximizing the insights gained. This methodology allows researchers and scientists to strategically select experimental conditions, variables, and sample sizes, leading to more accurate and robust conclusions from the available data. It has become instrumental in a wide range of scientific disciplines, including chemistry, biology, physics, and engineering, ultimately accelerating the pace of knowledge discovery and innovation. Data-driven DoE becomes even more complex when: (i) some of the variables involved in the historical data must be set to a specific value, as determined by the experts [Soloviev et al., 2022b]; (ii) there are several constraints on the variables; or (iii) there is no historical data available from which to learn.

Recent approaches include complex simulation methodologies such as adaptive sampling [Liu et al., 2018a] in which promising regions of the search space are detected and sampled iteratively. Adaptive sampling has been combined with Bayesian optimization and widely used as a data-driven DoE approach. This leverages a surrogate model that can be sampled efficiently and an acquisition function which is used to select promising solutions from the search space [Greenhill et al., 2020, Hanaoka, 2021, 2022]. Other approaches found in the literature face the task as a multi-level optimization problem, where evaluating a given solution involves solving further optimization tasks [Sun et al., 2020], although this design decision is problem dependent. However, all these approaches act as gray boxes where all variables are of equal importance and it is not possible to find knowledge of the relationships between variables.

In this paper, we present a new data-driven DoE approach aimed at obtaining an optimal multi-ingredient fuel exploiting previous experimentation provided by experts. The design of the fuel is modeled as a black-box multi-objective optimization problem with $m = 3$ objectives, in which one of the objectives ($f_1(\boldsymbol{x})$) is estimated using an embedded regression model, while the others ($f_2(\boldsymbol{x}), f_3(\boldsymbol{x})$) are computed analytically. Moreover, several bounds are imposed regarding the fuel descriptors which lead to a constrained optimization problem. Starting from an initial set of experiments, provided by the experts, the EDA approach converges to a new set of experiments to be evaluated into a laboratory, where the real properties are experimentally measured. The new data points are fed back in the framework loop and the EDA is newly executed to further explore the Pareto frontier in search of the optimal solutions. By the use of Gaussian Bayesian networks, our approach is able to restrict the search space and propose feasible solutions for the optimization problem. The level of exploration/exploitation over the initial set of experiments is regulated by the introduction of a new parameter, which is experimentally analyzed. A posteriori analysis is performed, in which relevant variables

during the optimization process are identified.

Thus, the main contributions of the paper are:

- The use of multi-objective EDAs to approach data-driven DoE.
- The probabilistic model enforcement by a regression model to improve the accuracy.
- The parametrization of the level of exploration and exploitation of the algorithm in the landscape.
- The posteriori analyses in which dependencies between the variables are detected based on the findings of the BN.

This chapter includes the developed methodology and results included in Soloviev et al. [2024c]. Data has not been publicly disposed for confidential reasons. However, a deep description of the data is provided in Appendix B. The implementation of the algorithm will be added to `EDAspy`[1] after paper acceptance.

Chapter outline

The outline of this chapter is organized as follows. Section 6.2 presents the formal formulation of the black-box multi-objective optimization problem. The proposed methodology for the optimization task and further results are presented in Section 6.3. Section 6.4 rounds the chapter off with the conclusions and future research lines.

6.2 Optimizing the design of fuel

A fuel is a physical mixture of several ingredients of different chemical nature and is normally defined in terms of the relative weight percentage of each ingredient in the formulation.

By starting from an initial pool of n available ingredients, the design of a fuel for a specific application requires to efficiently explore an n-dimensional search space in order to find optimal combinations of ingredients, whose physical properties must meet a set of constraints.

In this work we are dealing with $n = 24$ ingredients $I_1, \dots, I_{24}$ that can be grouped into six different categories $\{A, B, C, D, E, F\}$ and each ingredient is characterized by a set of physical properties $P1, \dots, P14$, as shown in Table B.1 (B). B shows a histogram for each of the properties.

Additionally, the total fuel mixture is characterized by a set of descriptors, namely $W = (W_{calc}, W_{lab})$. The former (W_{calc}) includes the percentage of some of the ingredients, and certain physical properties that can be computed analytically as weighted sums of the individual properties of the ingredients. The latter (W_{lab}) refers to physical properties that cannot be computed analytically, and have to be evaluated experimentally in the laboratory. Both types of descriptors are bounded by a lower (LB) and upper (UP) bounds, respectively, leading to optimization constraints. B.2 describes W_{calc} formulas, and bounds for all the descriptors.

[1] https://github.com/VicentePerezSoloviev/EDAspy

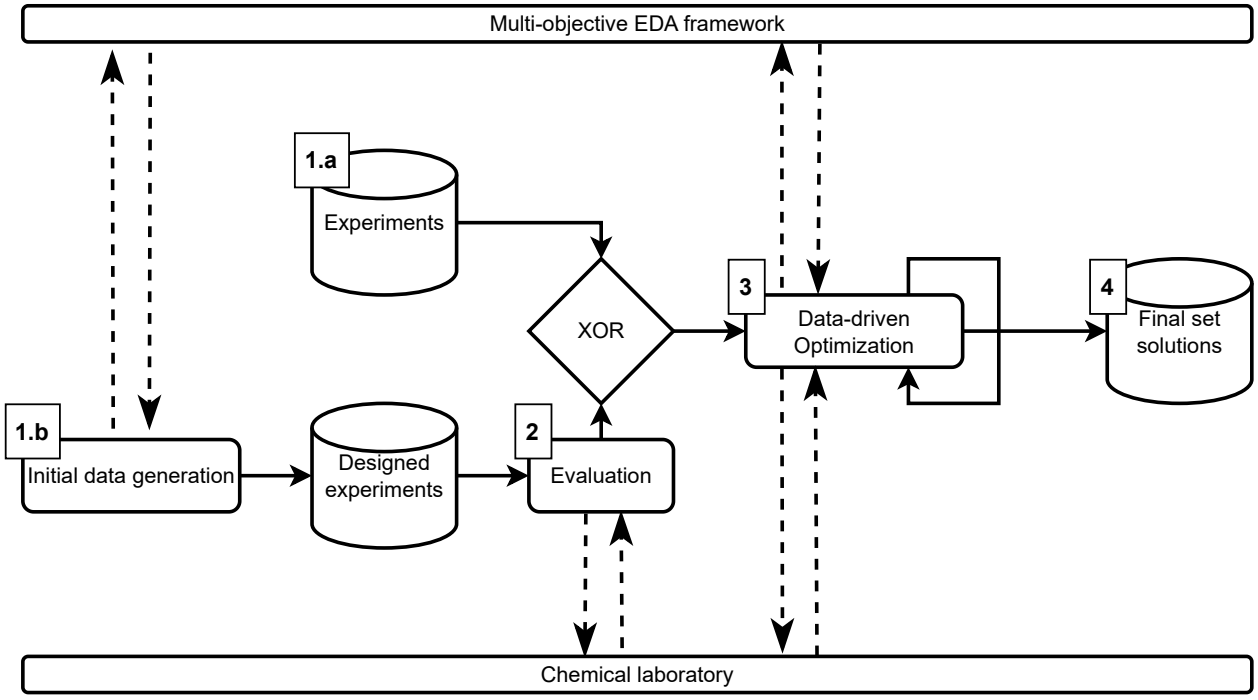


Figure 6.1: General flow chart of the multi-objective proposed approach for data-driven DoE.

Although the properties of the ingredients are known, their interaction as well as how they affect the laboratory descriptors (W_{lab}) is not known a priori. The optimizer proposed in this work not only allows to efficiently navigate the search space to accelerate the design of optimal fuels but also provides useful insights into the effect of each ingredient over the descriptors of interest.

From the set of descriptors, W^1_{lab}, W^1_{calc} and W^2_{calc} are desired to be maximized in the multi-objective optimization problem.

Figure 6.1 depicts a schematic view of the multi-objective sequential learning framework for fuel design presented in this paper. Considering the high dimension of the search space and that laboratory experiments are time consuming and subjected to budget restriction, it is advantageous to assist the fuel design task with a multi-objective optimization algorithm that efficiently and iteratively explores the Pareto frontier guiding the search towards an optimal fuel that meets all the previously mentioned constraints.

By leveraging past experiments (step 1a), our methodology aims at finding a sweet spot between testing new fuels with different compositions from those already tested (exploration) and exploiting the knowledge enclosed in the existing data points, to suggest a batch of new formulations to be tested in the laboratory. Once these new fuels are tested in the laboratory, if none of the suggestions meets the constraints, the optimization procedure is run again to suggest a new batch of experiments (step 3) until convergence to a final set of optimal solutions is achieved (step 4). Additionally, if no past experiments are provided, we propose a method based on a Latin hypercube sampling for proposing an initial set of experiments (step 1b, Section 6.3.4) to be tested in the laboratory (step 2). The data-driven optimization

process is carried out starting from either a set of historical data, or, the set of proposed experiments (XOR).

The proposed approach also allows including domain knowledge to further guide or restrict the search for optimal solutions. For example, experimentalists can be interested in fixing the amount of certain ingredients to either 0 or a finite value between 0 and 100 and no solutions should suggest different values for those ingredients whose amount has been fixed. Previous research in this line has referred to these variables as environment variables [Soloviev et al., 2022b].

6.2.1 Problem formulation

The problem involves $n = p + c$ variables, $\boldsymbol{X} = (\boldsymbol{Y}, \boldsymbol{Z}) = (Y_1, \ldots, Y_p, Z_1, \ldots, Z_c) = (X_1, X_2, \ldots, X_n)$, which represent percentages of each ingredient in the total mixture of the fuel, where $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are the environment variables whose value ($\boldsymbol{y}$) is predefined by the experts and the decision variables ($\boldsymbol{Z}$) that are to be tuned by the algorithm, respectively. The optimization problem is defined as

$$
\begin{aligned}
\max_{\boldsymbol{x}} \quad & G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), g_3(\boldsymbol{x})) \\
& g_1(\boldsymbol{x}) = W^1_{lab}(\boldsymbol{x}) - i(\boldsymbol{x}) \\
& g_2(\boldsymbol{x}) = W^1_{calc}(\boldsymbol{x}) - i(\boldsymbol{x}) \\
& g_3(\boldsymbol{x}) = W^2_{calc}(\boldsymbol{x}) - i(\boldsymbol{x}) \\
& i(\boldsymbol{x}) = \sum_{i=1}^{4} j(\boldsymbol{x}, W^i_{lab}(\boldsymbol{x})) + \sum_{i=1}^{7} j(\boldsymbol{x}, W^i_{calc}(\boldsymbol{x})) \\
& j(\boldsymbol{x}, W^i_{lab}(\boldsymbol{x})) = \begin{cases} 0 & \text{if } \mathrm{LB}^i_{lab} \leq W^i_{lab}(\boldsymbol{x}) \leq \mathrm{UB}^i_{lab}, \\ \delta & \text{otherwise} \end{cases} \\
& j(\boldsymbol{x}, W^i_{calc}(\boldsymbol{x})) = \begin{cases} 0 & \text{if } \mathrm{LB}^i_{calc} \leq W^i_{calc}(\boldsymbol{x}) \leq \mathrm{UB}^i_{calc}, \\ \delta & \text{otherwise} \end{cases} \\
\text{subject to} \quad & \boldsymbol{x} \in \mathbb{R}^n_+ \\
& \mathcal{L}(\mathcal{G}_0, \boldsymbol{\Theta}_0 : \boldsymbol{x}) + \alpha \leq 0 \\
& \sum_{i=1}^{n} x_i = 100,
\end{aligned}
$$

where $\mathcal{L}(\mathcal{G}_0, \boldsymbol{\Theta}_0 : \boldsymbol{x})$ is the log-likelihood of a solution $\boldsymbol{x}$ in the probabilistic model learned from the initial data formally represented by a DAG $\mathcal{G}_0$ and some parameters $\boldsymbol{\Theta}_0$ (Equation 2.7); $\alpha \geq 0$ represents the level of exploration-exploitation, being the higher, the more exploratory; $\delta > 0$ represents the penalization term for the constraint set in the problem; and $(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), g_3(\boldsymbol{x}))$ are the three objectives related to $(W^1_{lab}(\boldsymbol{x}), W^1_{calc}(\boldsymbol{x}), W^2_{calc}(\boldsymbol{x}))$ functions to optimize, respectively.

6.3 Methods and results

In this section we deeply explain the methodology implemented to solve the task described in Section 6.2. Figure 6.1 outlines the framework used in this task. The pseudocode in Algorithm 6 shows the algorithm behind the multi-objective EDA framework described in Figure 6.1.

Algorithm 6 Multi-objective data-driven EDA

Input: Population size N, selection ratio τ, cost functions (g_1, g_2, g_3), exploration ratio α and historical data $\mathcal{X}_{train}$

Output: Best individual $\boldsymbol{x}'$ and cost found $G(\boldsymbol{x}')$

1: $G_0 \leftarrow \mathcal{X}_{train}$ historical data
2: **for** $t = 1, 2, ...$ until stopping criterion is met **do**
3: Evaluate G_{t-1} according to (g_1, g_2, g_3)
4: Rank G_{t-1} according to Pareto dominance for (g_1, g_2, g_3)
5: $G^S_{t-1} \leftarrow$ Select $\lfloor \tau N \rfloor$ individuals from G_{t-1}
6: $p_{t-1}(\cdot) \leftarrow$ Learn a Gaussian BN from G^S_{t-1}
7: $p_{t-1}(\cdot) \leftarrow$ Set environment variables $\boldsymbol{Y} = \boldsymbol{y}$
8: $G_t \leftarrow$ Sample N individuals from $p_{t-1}(\boldsymbol{Z}|\boldsymbol{Y} = \boldsymbol{y})$
9: **end for**

In each iteration of the general workflow (Figure 6.1), the EDA approach (MO_EDA) is executed and receives a new historical data $\mathcal{X}_{train}$, which includes the historical data used in the previous iteration as well as the new batch of solutions suggested and validated in the laboratory. The exploration ratio α is tuned in each iteration of the workflow as desired by the experts.

The EDA approach is an iterative algorithm that evaluates a set of individuals (line 3) in each iteration, starting starting from the historical data, according to (g_1, g_2, g_3) (Section 6.2 and Section 6.3.1). The solutions are ranked based on the Pareto dominance criteria for the cost functions (line 4 and Section 6.3.3). From this ranking, $\lfloor \tau N \rfloor$ solutions are selected (line 5) to train the posterior conditional probability described in Section 6.3.2 (line 6-7), and sample N new solutions (line 8).

6.3.1 Prediction of descriptors

As described in Section 6.2, one of the objectives to be maximized (W^1_{lab}) cannot be computed analytically, and thus, has to be predicted based on the rest of variables. Therefore, a regression model has been implemented to predict W^1_{lab}.

Considering the target variable W^1_{lab} found in the historical data provided by the experts, we train different regression models, where the input variables are the decision variables $\boldsymbol{Z}$ (set of fuel ingredients in the feature space to be tuned by the EDA approach). We prepare a cross-validation leaving-one-out experiment to find the model that most accurately predicts W^1_{lab}. The tested models [Pedregosa et al., 2011] are Bayesian ridge regression, Lasso regression, Ridge regression, XGBoost regression and kernel ridge regression and are

	BRidge	Lasso	XGBoost	Ridge	KRidge
R2 (%)	57.01	55.53	39.73	53.54	47.82
MAE	1.47	1.51	1.95	1.58	1.75
MSE	1.48	1.53	2.07	1.61	1.79
RMSE	1.22	1.24	1.44	1.27	1.34

Table 6.1: R2, MAE, MSE and RMSE results of cross-validation leaving-one-out experiments after predicting W^1_{lab} using $\boldsymbol{Z}$ as input variables

available at scikit-learn-1.3.0 [Pedregosa et al., 2011]. Hyperparameter configurations for these algorithms can be found in C.1.

Table 6.1 shows the R2, mean absolute error (MAE), mean squared error (MSE) and root mean squared error (RMSE). It can be observed that Bayesian ridge regression obtains the best results, while XGBoost performs the worst. In order to improve the performance of the regression model, we have carried out a hyperparameter tuning using Bayesian optimization [Bergstra et al., 2013], and R2 results have been increased to 60%. The final configuration is available in C.1.

For each sample generated from the probabilistic model, we predict W^1_{lab} with the regression model and use $\hat{W}^1_{lab}$ to evaluate the solution in the cost function $G(\boldsymbol{x})$ (Section 6.2). W^1_{calc} and W^2_{calc} are computed analytically, and the rest of the W^i_{calc} are sampled from the probabilistic model.

6.3.2 Probabilistic model

The algorithm will utilize the embedded probabilistic model to learn patterns identified during the optimization process and to sample new feasible solutions. Gaussian BNs have been chosen for the following reasons: (i) they allow the use of evidence in the model, causing that all the samplings will have the environment variables fixed to a specific value [Soloviev et al., 2022b]; (ii) they represent uncertain knowledge as a graph; and (iii) we are dealing with continuous variables.

In the historical data, we find fuel formulas with ingredients that traditionally belonged to the set of decision variables $\boldsymbol{Z}$. However, due to current governmental restrictions on fuels, they must be restricted to a specific value or be entirely banned. Thus, we must set these variables to a fixed value, i.e. environment variables [Soloviev et al., 2022b]. The environment variables $\boldsymbol{Y}$ are evidence in the learned Gaussian BN, so that future solutions are sampled from the posterior conditional probability $f(\boldsymbol{Z}|\boldsymbol{Y} = \boldsymbol{y})$.

The sampling process has been parameterized so that the degree of exploration and exploitation is controlled by the log-likelihood ($\mathcal{L}(\dot{)}$) of each sample $\boldsymbol{x}$ in the model learned from $\mathcal{X}_{train}$, where $\mathcal{X}_{train}$ is the historical data. A sample is only valid if $\mathcal{L}(\mathcal{G}_0, \boldsymbol{\Theta}_0 : \boldsymbol{x}) + \alpha \leq 0$. As a reference, we have computed the log-likelihood of each sample in $\mathcal{X}_{train}$ under the model represented by $(\mathcal{G}_0, \boldsymbol{\Theta}_0)$. A box plot is shown in Figure 6.2 where mean and median are approximately -46 and -42, respectively. Thus, for increasing values of exploratory parameter

($\alpha \to \infty$), the sampling process becomes more exploratory, and for α values close to zero, the sampling process tends to an exploitation behavior. Further analyses on the performance of the EDA approach depending on α parameter are analyzed in Section 6.3.5.

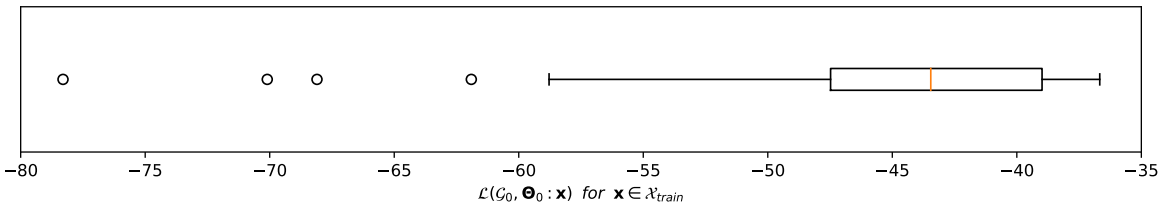


Figure 6.2: Box plot of the log-likelihood of each sample from $\mathcal{X}_{train}$ in the model represented as $(\mathcal{G}_0, \boldsymbol{\Theta}_0)$, where mean and median are approximately -46 and -42, respectively.

6.3.3 Truncation

The truncation process is performed in each iteration of the algorithm. Most of multi-objective approaches use the hypervolume metric to select the top solutions that best approximate the Pareto frontier. However, in our approach we rank to solutions using the different frontiers as shown in Figure 6.3, where solutions from different frontiers (from the first one onwards) are selected to train the probabilistic model of the respective generation. This way, we expect to avoid local convergence by including more diversity in the truncation set. Moreover, this would reduce the computation time by preventing to compute multi-objective quality indicators in each iteration.

6.3.4 Initial data generation

Although some historical data was provided for the problem we are addressing in this paper, we propose a methodology based on Latin hypercube sampling (LHS) to generate the initial

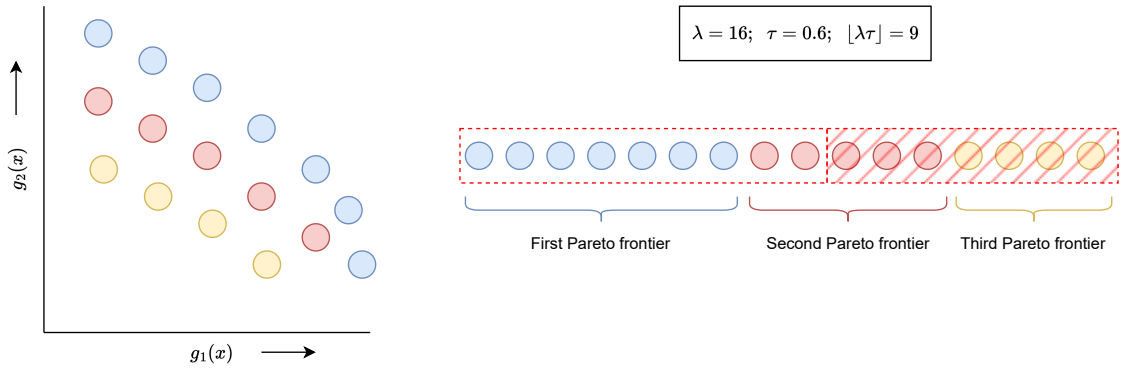


Figure 6.3: First, second and third Pareto frontiers represented as blue, red and yellow circles, respectively, are identified for maximizing $g_1(\boldsymbol{x})$ and $g_2(\boldsymbol{x})$ objectives, and ranked for the truncation process. An example for $N = 16$ and $\tau = 0.6$ is shown.

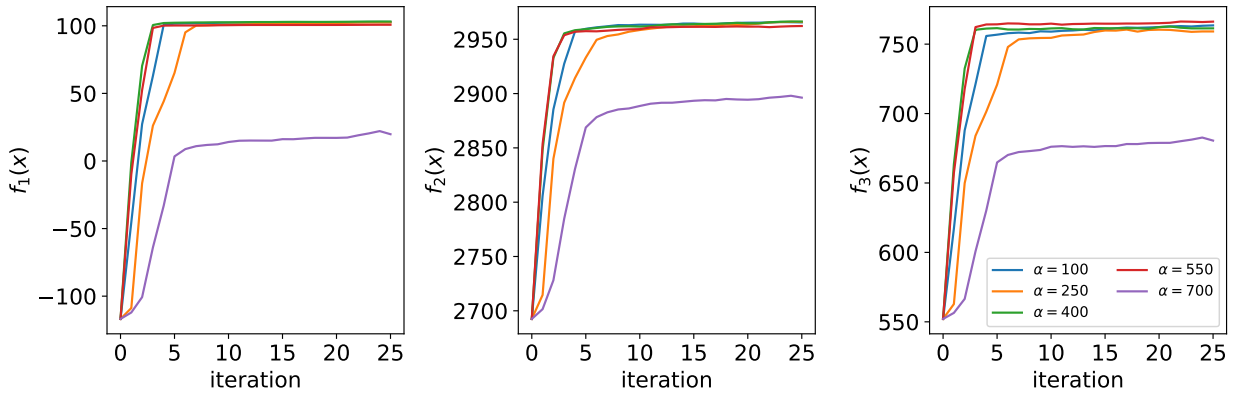


Figure 6.4: f
or different values of exploratory parameter αConvergence plot of the three objectives $(g_1(\boldsymbol{x}), g_2(\boldsymbol{x})), g_3(\boldsymbol{x})$ for different values of exploratory parameter α. Each line represents the mean objective value in each iteration after executing the EDA approach 25 times for each α configuration.

set of experiments (Figure 6.1, step 1.b). LHS has traditionally been used in a wide range of sampling designs [Fang et al., 2005].

The overall performance of the optimization process heavily depends on the quality of the samplings of the initial data. A set of samplings that efficiently explores the landscape of solutions, will facilitate, and most probably, ensure the discovery of good feasible solutions. However, a poor set of samplings may lead the optimizer to converge to local optima.

LHS is a statistical technique that efficiently explores a multi-dimensional parameter space while ensuring a representative sample of the input variables. LHS divides each variable range into equally likely intervals (number of samplings) and selects a single value from each interval, creating a stratified, non-repetitive sample that maximizes the coverage of the entire parameter space. A permutation between the samplings of each dimension is then run. We add a post-processing hill-climbing method to ensure equally spaced samples using Lloyd-Max algorithm [Lloyd, 1982].

Although this intelligent sampling process ensures that all areas of the landscape are sampled, it is desired to execute in the laboratory those that are most promising for promoting the next generations of the algorithm. Thus, in Figure 6.1, the initial data process collaborates with our approach. The initial sampling is used to initialize the EDA, and the best solutions found are taken to the laboratory. The exploratory parameter α will be tuned by the user in such a way that the solutions are as exploratory as possible.

6.3.5 Performance analysis

Figure 6.4 shows a comparison of the three objectives for different values of exploratory parameter α after running the EDA approach 25 independent times for each configuration.

Firstly, it can be observed that along the iteration process (x-axis) the objectives are maximized

until they reach a constant value, and the algorithm is considered to have converged.

Secondly, depending on the values that the parameter takes, the algorithm reaches different optima. In the case of W^1_{lab} the best result is reached for almost all the values of the parameter. Parameters of $\alpha = 400$ and $\alpha = 580$ seem to be the best configurations, reaching the best values for the three objectives. Finally, $\alpha = 700$ returns the worst results for the three objectives.

To analyze the best parameter configuration we will use the HV quality indicator (Equation 3.7). In this case, the experts fix the reference ideal point to $g^{ref} = (102 + \text{RMSE}(\hat{W}^1_{lab}), 2980, 780)$ for the HV computation, where $\text{RMSE}(\hat{W}^1_{lab})$ is the root mean squared error obtained by the regression model, computed in Table 6.1.

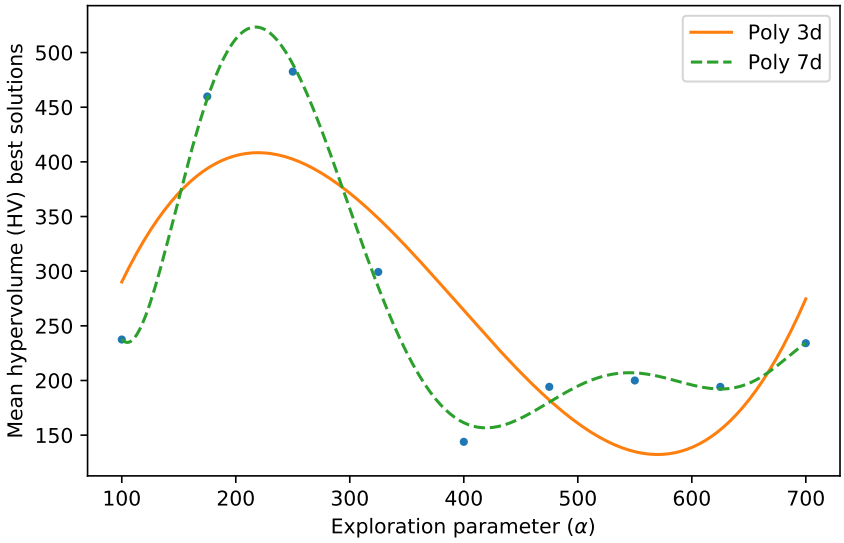


Figure 6.5: Mean HV after running the EDA approach 25 independent times for different values of exploratory parameter α. Blue dots represent the HV mean values, and orange and green dashed lines represent a function approximation using a polynomial with degree three and seven, respectively.

Figure 6.5 shows the mean HV achieved after 25 independent executions for each parameter configuration. It can be observed that there are no signs of increasing or decreasing monotony. In fact, we observe a non monotonic line for each function approximation with a valley around values of $\alpha = 400$ and $\alpha = 580$. Note that the results obtained for low values of α, that is, low exploration and high exploitation with solutions similar to the ones provided by the experts in the historical data, achieve a HV of approximately 250 while the best results are obtained for $\alpha = 400$ with a mean HV of approximately 150. For greater values of parameter α, the mean HV is worse, but still better than the one achieved by the historical data. When $\alpha \to \infty$, an increasing tendency of the HV has been observed, which agrees with the results shown in Figure 6.4.

We conjecture that although the historical data contains good performing fuel formulas, if we move to another area of the search space within the landscape of solutions, we will find a fuel

formula with better performance and objective values. Therefore, the experts were using a formulation corresponding to a local optimum. An overly exploratory approach can lead to very poor solutions, as can be seen for $\alpha = 700$ in Figure 6.4 and Figure 6.5.

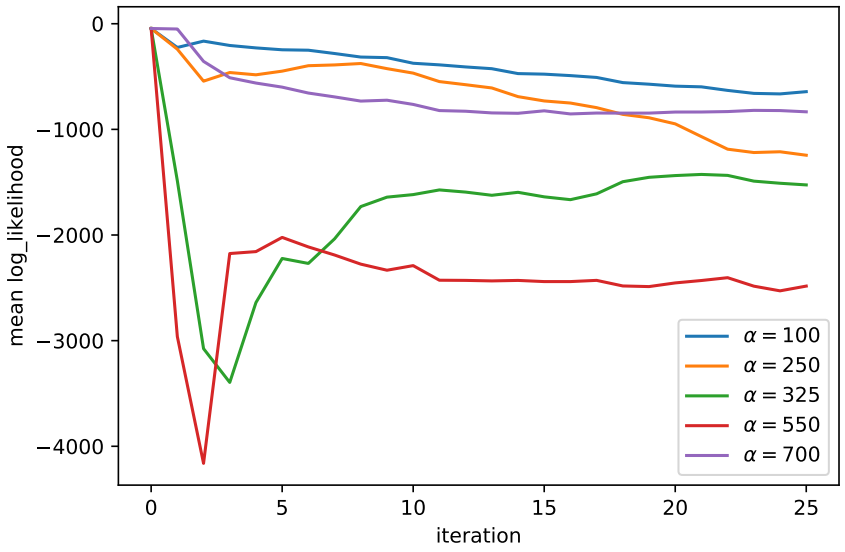


Figure 6.6: Mean log-likelihood of the set of solutions in each iteration along the optimization process for different values of α parameter.

Figure 6.6 shows the mean log-likelihood of the solutions in generation G_t along the optimization process. In general, a decreasing trend can be observed independently of the values of α, always respecting the threshold imposed by the exploration parameter.

Note that a common behavior among the different α values is that in the first iterations the log-likelihood decreases more sharply than in the following iterations. We conjecture that this behavior is justified by the fact that the EDA approach in the first iterations becomes more exploratory, but as the optimization advances it becomes more exploitative (within the range allowed by the α parameter), converging to already explored areas of the landscape.

Figure 6.7 shows an example of the new optimal proposals provided by the EDA approach for different values of exploration parameter α, analyzed in two dimensions of the landscape (Table B.1). Orange circles represent the traditional fuel formulas found in the historical data presented by the experts, while blue symbols represent the best EDA solutions for different values of the exploration parameter.

The left panel illustrates a comparison between D4 and D3 ingredients, which belong to the same category of ingredients. Traditionally, the experts use either one of the two ingredients, however, EDA proposes new formulas in which both ingredients are combined.

The right panel shows a comparison between F2 and C2, which do not belong to the same category of ingredients. Typically, experts tend to maximize the first ingredient while minimizing the second one. It is observed that some of the EDA solutions try to imitate this knowledge learned from the historical data, but other solutions adopt a different perspective,

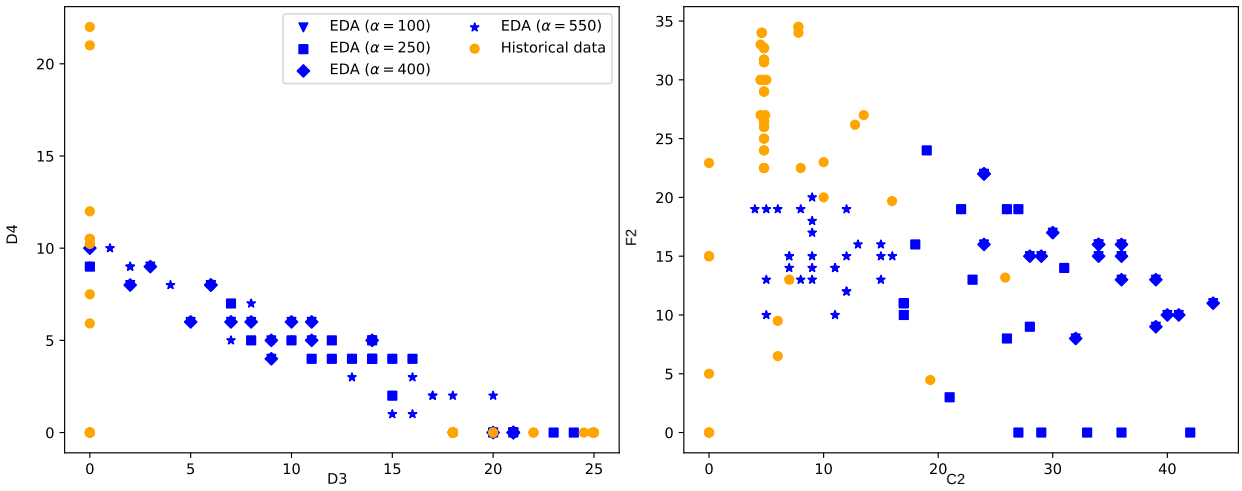


Figure 6.7: Comparison of two dimensions of the problem landscape where orange circles represent traditional formulas found in the historical data, and the rest of symbols represent the new optimal formulas found by the our approach. The left panel compares D4 and D3 ingredients and the right panel compares F2 and C2 ingredients.

maximizing the second ingredient while minimizing the other. This reflects the exploratory power of our approach.

6.3.6 Knowledge discovery

In this section, we analyze the set of variables that are strongly related to $g_1(\boldsymbol{x})$ in order to identify the dependencies between the fuel ingredients and the optimization objective, as this information is not known a priori.

It is worth mentioning that the Gaussian BN analyzed has been learned from the solutions found when the exploration parameter has been tuned to $\alpha = 400$ (where best results have been found). Thus, the strongly related variables found for different values of the parameter may vary.

Figure 6.8 represents the structure of the Gaussian BN learned with the best solutions found in the last 5 iterations of the EDA approach with $\alpha = 400$, where only the arcs contained in the Markov blanket of $g_1(\boldsymbol{x})$ are shown, i.e. $\mathbf{MB}(g_1(\boldsymbol{x}))$.

The Markov blanket of $g_1(\boldsymbol{x})$ includes ingredients from the categories $\{C, D, E, F\}$ and the objective $g_3(\boldsymbol{x})$. Remarkably, all the variables included in $\mathbf{MB}(g_1(\boldsymbol{x}))$ are the parents of $g_1(\boldsymbol{x})$ in the graph (Figure 6.8), and the conditional probability density (Equation 2.4) is expressed as:

$$\begin{aligned} f(g_1(\boldsymbol{x})|\boldsymbol{Pa}_{g_1(\boldsymbol{x})}) = \mathcal{N}(&142.01 - 0.239\, D4 - 0.158\, C2 - 0.105\, F3 + \\ &- 0.123\, D3 - 0.166\, F2 - 0.065\, g_3(\boldsymbol{x}) \\ &- 0.073\, E2;\ \sigma^2), \end{aligned} \tag{6.1}$$

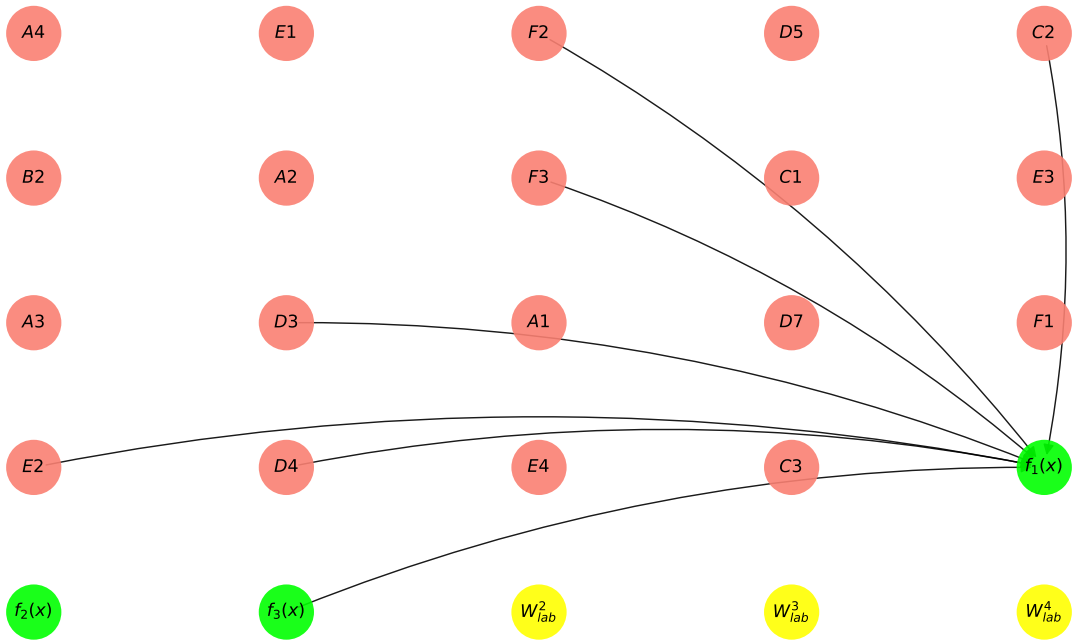


Figure 6.8: Gaussian BN structure learned from the best solutions found in the last 5 iterations of the EDA approach with $\alpha = 400$, where only the arcs contained in the Markov blanket of $g_1(\boldsymbol{x})$ are shown. Red, green and yellow nodes represent the ingredients, optimization objectives and W^2_{lab}, W^3_{lab}, W^4_{lab}, respectively.

where numbers represent the weights $(\beta_1, \ldots, \beta_7)$ of how the value of each variable influences the mean objective $g_1(\boldsymbol{x})$ in the conditional probability density; $\beta_0 = 142.01$ is the intercept coefficient; and σ^2 is the variance.

It is observed that the two variables with the highest weight associated, and thus, with more influence on the final objective are $D4$ and $C2$. Although the weight associated to $g_3(\boldsymbol{x})$ is low, the range of values of this objective is more than one magnitude higher than the values of the ingredients. Thus, we find a strong relation between both $g_1(\boldsymbol{x})$ and $g_3(\boldsymbol{x})$ objectives.

Figure 6.9 shows Pearson correlations computed over the set of best solutions found in the last iterations of the algorithm. Those variables more correlated with the objective $g_1(\boldsymbol{x})$ will denote a big influence between both variables. It can be observed that $C2$ and $F2$ are the two variables most correlated with $g_1(\boldsymbol{x})$.

Note that $g_2(\boldsymbol{x})$ and $g_3(\boldsymbol{x})$ heavily depend on the analytical descriptors, and thus the dependence between the ingredients and these objectives is already known.

6.3.7 Comparison

To perform a fair analysis of our approach (MO_EDA) it is compared to others, such as NSGA-II [Deb et al., 2002] and MOEAD [Zhang and Li, 2007]. The algorithms hyperparameters

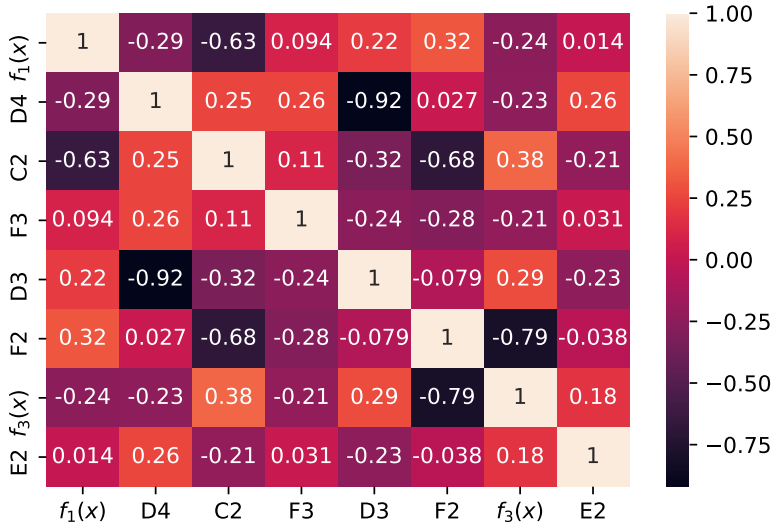


Figure 6.9: Heatmap that represents the Pearson correlation found between the variables in $\mathbf{MB}(g_1(\boldsymbol{x}))$ and $g_1(\boldsymbol{x})$ in the set of best solutions obtained in the last 5 iterations of the EDA approach.

	MO_EDA	NSGA-II	MOEAD
HV	124.63 ± 74.99	535.13 ± 265.8	260.7 ± 38.13
DM	7.27 ± 7.57	213.82 ± 304.0	4.38 ± 4.46

Table 6.2: Mean and standard deviation of hypervolume (Equation 3.7) and diversity (Equation 3.9) indicators for the best solutions obtained by MO_EDA, NSGA-II and MOEAD approaches.

tuning can be found in Appendix C.2. HV (Equation 3.7) and DM (Equation 3.9) are used for the evaluation of the algorithms.

Figure 6.10 represents the best results found by NSGA-II, MOEAD and MO_EDA approaches, plotted with blue, orange and green markers, respectively. Note that, in the case of our approach, the two first Pareto frontiers are shown as two separate groups in the landscape of objective values.

Figure 6.10 shows that, while MOEAD seems to find an area of the search space in which all the solutions have a similar result (in terms of the three analyzed objectives), NSGA-II returns solutions with very varied results. With focus on the reference point ($g^{ref} = (102 + \text{RMSE}(\hat{W}^1_{lab}), 2980, 780)$), the solutions found by our approach seem to outperform the ones found by its competitors.

Results observed in Figure 6.10 are contrasted to the results of hypervolume and diversity metrics found in Table 6.2. Note that to compute those metrics, just the first frontier of solutions from EDA approach was considered. The large variance among the results of

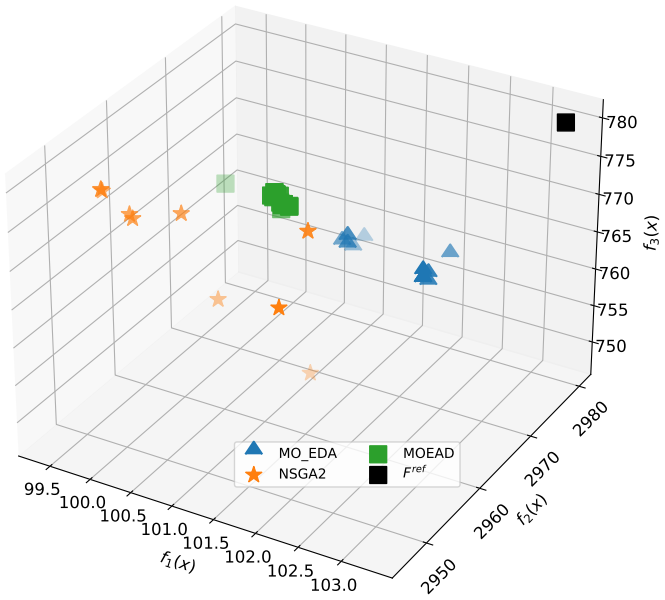


Figure 6.10: Blue, orange and green markers represent the MO_EDA, NSGA-II and MOEAD best solutions, respectively in the landscape of the three objectives to be optimized. Black square represents the reference point (g^{ref}) used to compute HV indicator. More than one frontier of solutions for MO_EDA were included. X, Y and Z axis represent $g_1(\boldsymbol{x})$, $g_2(\boldsymbol{x})$ and $g_3(\boldsymbol{x})$, respectively.

NSGA-II is reflected in DM indicator compared to the computed metric for MO_EDA and MOEAD approaches. Also, Figure 6.10 places the solutions far from the reference point (g^{ref}), and the hypervolume computed for this algorithm is the worst one with a large standard deviation. Although MO_EDA and MOEAD approaches achieve a similar diversity indicator, our approach outperforms its competitors in terms of hypervolume minimization.

ANOVA test [Kaufmann and Schering, 2007] has been computed to reject the null hypothesis of equal means for hypervolume and diversity indicators, where statistical significant differences have been found (p-value $= 3.07e-8$ and p-value $= 3.82e-28$, respectively).

6.4 Conclusions

In this chapter we have proposed a new methodology to be applied in formulation laboratories for the experimental design of fuels. We expect to change the paradigm of traditional methods, to a data-driven approach in which the number of real experiments is reduced leading to cost-cutting and time-saving.

The experimental design has been approached as an optimization problem where three properties of the fuel mixture have been maximized as a multi-objective task. Comparing our approach to some state-of-the-art algorithms, experimental results have found that the

EDA-based optimizer outperforms its competitors.

Our approach includes an α parameter which controls the level of exploration and exploitation. Experimental results have shown that accurately tuning this parameter will lead to discovering better results than those already known by the experts. Moreover, the probabilistic model embedded by the EDA approach is able to learn patterns between fuel ingredients that were not considered by the experts in their experimental designs, thus enriching the interpretability of the solution.

We are aware that this is a first step towards the adoption of a new paradigm in the design of fuels, and there exist several ways of improvement for this approach. Among future steps, the following are included.

- The proposed approach samples $f(\boldsymbol{Z}|\boldsymbol{Y} = \boldsymbol{y})$ in each iteration, where $\boldsymbol{Z}$ and $\boldsymbol{Y}$ are the decision and environment variables of the problem, respectively, and uses $\hat{W}^1_{lab}$ to compute the multi-objective cost function. An improvement of the probabilistic model could happen if W^1_{lab} was introduced in the probabilistic model as an additional node, and the sampling of W^1_{lab} from the model was considered to compute the multi-objective cost function. The uncertainty of this sampling could also be analyzed.
- Exploration parameter (α) was tuned empirically, so finding a way to automatically tune it would mean a more autonomous approach.
- Although the control of exploration/exploitation of the algorithm is tuned through the log-likelihood, the use of geometric metrics in the landscape would better explore the optimization surface.
- Although in this optimization problem the historical data fits Gaussian distributions, most of the industrial data does not fit normal distributions. Improving this approach with a semiparametric perspective [Soloviev et al., 2023a] where the algorithm can decide itself the distribution to be used would enlarge the area of application of the data-driven approach.

Chapter 7

Semiparametric EDA for Continuous Optimization

7.1 Introduction

Chapter 5 assesses the performance of an EGNA approach for a real application in the Industry 4.0. Future work included overcoming the limitation of the Gaussian assumption over data. In this chapter, we propose a novel methodology (SPEDA) in which this limitation is overcome by allowing the coexistence of Gaussian and KDE-based probability distributions in the probabilistic model. Moreover, the proposed methodology considers information retrieved from more than one past generation, implementing an archive-based approach, which also satisfies another future work proposal.

In recent years, the automation of processes in industry has increased the need to optimize certain tasks that involve continuous variables, either given some data generated by sensors or cost functions. In the literature, we can find different probabilistic models embedded in EDAs being used to approach continuous optimization problems, such as the EGNA [Larrañaga et al., 2000], due to their simplicity and speed.

However, such models have certain disadvantages due to the use of parametric probability distributions. First, assuming Gaussianity when the search space of the optimization problem is a continuous environment that does not fit a Gaussian distribution, may result in poor solutions. Second, the use of Gaussian distributions during the runtime of the algorithm tends to shrink the search space represented by the standard deviation or the covariance matrix as the iterations of the algorithm progress, which can result in premature convergence, or genetic drifts [Neumann and Cairns, 2012]. Third, some algorithms, such as EGNA, use probabilistic models that consider dependencies between the variables; assuming Gaussianity in this type of model also implies assuming linear dependencies between variables, which has consequences for the way in which the algorithm navigates the landscape.

To try to address these shortcomings, some works use non-parametric models to avoid assuming a specific probability distribution over the search space, such as u_KEDA [Luo and Qian, 2009] (Section 3.3), in which a univariate KDE is used for modelling each variable as the

probabilistic model of the EDA. Despite improving the results over some state-of-the-art EDAs, this approach does not consider KDE multivariate probability models, or models in which only some of the variables fit a parametric probability distribution.

The premature convergence of EGNA approach has been widely studied in the literature. Some approaches involve modifying the variances of the variables manually so that they are not drastically reduced, such as Yuan and Gallagher [2005] which does not allow the variance of any variable to be reduced by more than one during the runtime of the algorithm, and Pošík [2008], which multiplies the variance of each variable by a constant value lower than one. The strategy of variance reduction when no better solutions are found after a certain number of iterations and increasing it when a new better solution is found to avoid falling into a local optimum was proposed in Grahl et al. [2006]. Other more complex approaches include setting a search direction based on information gathered during the execution of the algorithm. In this research line, it is worth mentioning the algorithm covariance matrix adaptation (CMA-ES) [Hansen, 2006], similar to EGNA, that is able to detect and establish a correct search direction when updating the covariance matrix and the vector of means. Other strategies include shifting the mean vector while keeping the covariance matrix [Fang et al., 2016].

However, some works [Chow and Yuen, 2011] have determined that these approaches could be improved, if instead of using only the information of the previous generation, the information of many previous generations were also used. Having a set of solutions of the entire history visited by the EDA, similar to a tabu list, is proposed. This strategy behaviour similar to some differential evolution approaches, such as the JADE algorithm [Zhang and Sanderson, 2009], which instead of using information from good solutions, uses information from bad solutions to generate new ones. Finally, in the area of EDAs, it was proposed to update the GBN considering the best individuals of several previous generations [Liang et al., 2018][Gao and Wood, 2012]. These approaches are the archive-based algorithms, where the archive length is the number of previous generations considered in the probabilistic model update.

In this chapter, we propose a new variant of EDAs for optimizing problems in continuous multivariate environments, in which we take advantage of the benefits provided by GBNs, and the accuracy of KDE-based models to find a trade-off between accuracy and computational cost. Our approach embeds a semiparametric Bayesian network [Atienza et al., 2022b] in which KDE variables coexist with Gaussian variables in the same probabilistic graphical model, where it is the algorithm itself that decides in each generation which variables take which type of probability distribution throughout the runtime. During the runtime, if the algorithm detects that all the variables are Gaussian in some iteration (extreme case), the proposed approach will use a GBN. Thus, we believe that our algorithm is able to identify the promising areas of the search space due to the use of semiparametric Bayesian networks but is a less heavy computational approach than using only KDEs. Furthermore, in order to reduce the probability of premature convergence occurrence [Ceberio et al., 2022b], we have designed an archive-based approach in which the probabilistic model is updated not only with the best individuals of the previous generation, but also with the best individuals of several previous generations.

This chapter includes the developed methodology and results included in Soloviev et al.

[2023a]. The implemented methodology is available in a GitHub repository[1], and the code of SPEDA, m_KEDA, EMNA, EGNA and benchmark suites are publicly available in `EDAspy` [2].

Chapter outline

The outline of this chapter is organized as follows. Section 7.2 introduces the new semiparametric EDA we propose. Section 7.3 reports the experimental results after comparing our approach to some state-of-the-art optimizers and analyses their CPU time and complexity. Section 7.4 ends the chapter with some conclusions and further work.

7.2 Semiparametric estimation of distribution algorithm

This section describes the proposed approach. First, the main drawbacks of the traditional EGNA algorithm are analyzed, and then we introduce SPEDA. Note that EMNA has similar drawbacks to EGNA due to the Gaussianity assumption, but in this section we will focus on EGNA since SPEDA aims to improve the results found by EGNA by using a more complex type of BN than GBNs.

7.2.1 EGNA

EGNA is one of the most extended multivariate EDAs among the state of the art methods of solving optimization problems in continuous spaces. In this algorithm, a GBN is learned in each iteration using the best individuals of the previous iteration. This GBN is sampled to obtain the new individuals of the present generation. This loop is iterated until a stopping criterion is met following the scheme of Algorithm 4. Thus, the mean $\boldsymbol{\mu}$ at each iteration t is estimated by the mean vector,

$$\boldsymbol{\mu}_t = \frac{1}{|G^S_{t-1}|} \sum_{i=1}^{|G^S_{t-1}|} \boldsymbol{x}_i^{t-1},$$

and the covariance matrix estimated by

$$\boldsymbol{\Sigma}_t = \frac{1}{|G^S_{t-1}|} \sum_{i=1}^{|G^S_{t-1}|} (\boldsymbol{x}_i^{t-1} - \boldsymbol{\mu}_t)(\boldsymbol{x}_i^{t-1} - \boldsymbol{\mu}_t)^T,$$

where G^S_{t-1} is the set of solutions selected from the previous generation, $\boldsymbol{\mu}_t$ is a vector that contains the mean of each of the n variables, $\boldsymbol{x}_i^{t-1}$ denotes the i-th selected individual in generation $t-1$, and $|\cdot|$ refers to the cardinality of the set.

In this chapter, to perform a fair comparison with the proposed algorithm, EGNA has been designed to learn the parameters of the model by maximizing the log-likelihood (Equation 2.7), and to learn the structure of the GBN using the HC algorithm [Gámez et al., 2011] (Algorithm 1) to optimize the BIC score [Schwarz, 1978] (Section 2.10).

[1] https://github.com/VicentePerezSoloviev/SPEDA
[2] https://pypi.org/project/EDAspy/

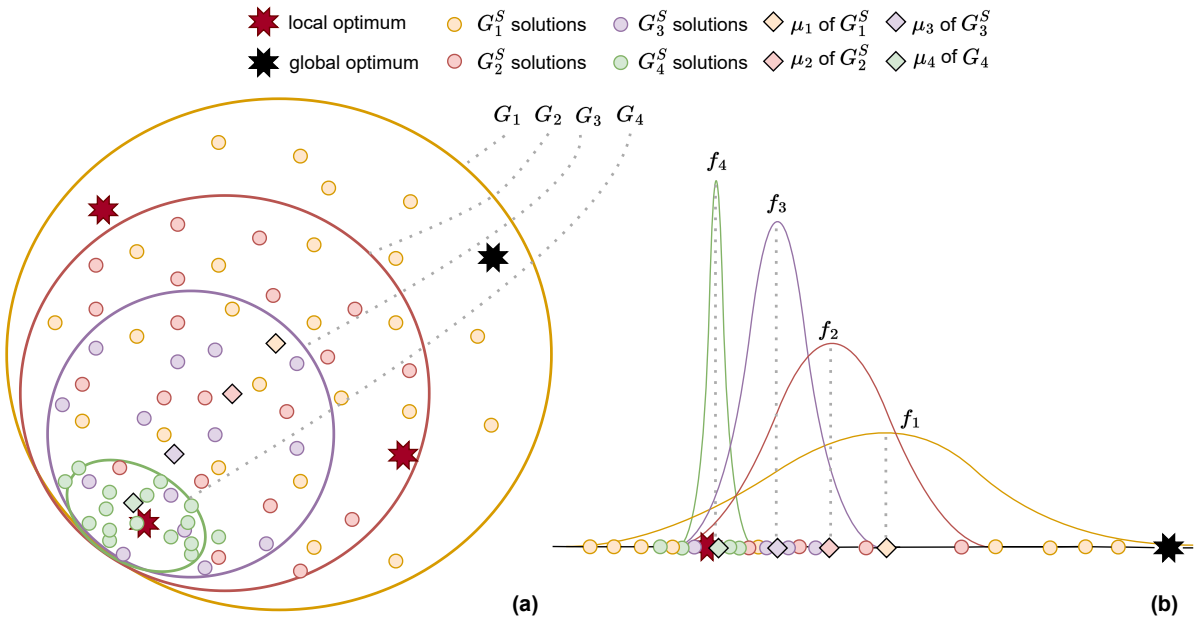


Figure 7.1: A search space that is iteratively approximated by an EGNA in two (a) and one (b) dimensions, where a single global optimum (black star) and some local optima (red stars) exist. The means and the search space area defined by the solutions of each generation G_t are represented by squares, and ellipses, respectively, in the two dimensional landscape and by squares and a Gaussian density function in the one dimensional landscape.

The EDA literature describes the search process of EGNA as a procedure in which the area of the space explored by each successive generation becomes increasingly small, until the algorithm converges to a small area in the landscape. This is represented in two dimensions in Figure 7.1(a) where the space explored by each iteration and $\boldsymbol{\mu}_t$ are represented as ellipses in the landscape and different coloured squares, respectively, and it is represented in one dimension in Figure 7.1(b). The sizes of these ellipses are defined by the covariance matrices $\boldsymbol{\Sigma}_t$, which play a key role in the explored search space of each generation, and the search direction is influenced by the positions of the means of consecutive generations $\boldsymbol{\mu}^t$ and $\boldsymbol{\mu}^{t-1}$.

Figure 7.2(a) shows how the Euclidean distance between the covariance matrix of each generation and the identity matrix becomes increasingly small as the iterations of the algorithm progress. Represented geometrically, the set of solutions selected from the previous generation G_{t-1}^S tends to lie in the semi-ellipse with a smaller distance to the global optimum being sought. By estimating a probabilistic model $f_{t-1}(\cdot)$ of this search subspace, the area defined by the new solutions G_t sampled from f_{t-1} is not only smaller than that defined by G_{t-1} but also assumes an already visited area during the algorithm runtime. This is shown in Figure 7.2(a), where each space defined by G_t is a subspace of the search area represented by the solutions of G_{t-1}. This process can cause different runs of the same EGNA configuration to converge to different areas of the search space and can thus result in premature convergence or high variance in the results found between the different runs.

The high generalization power of Gaussian probability distributions, which is generally

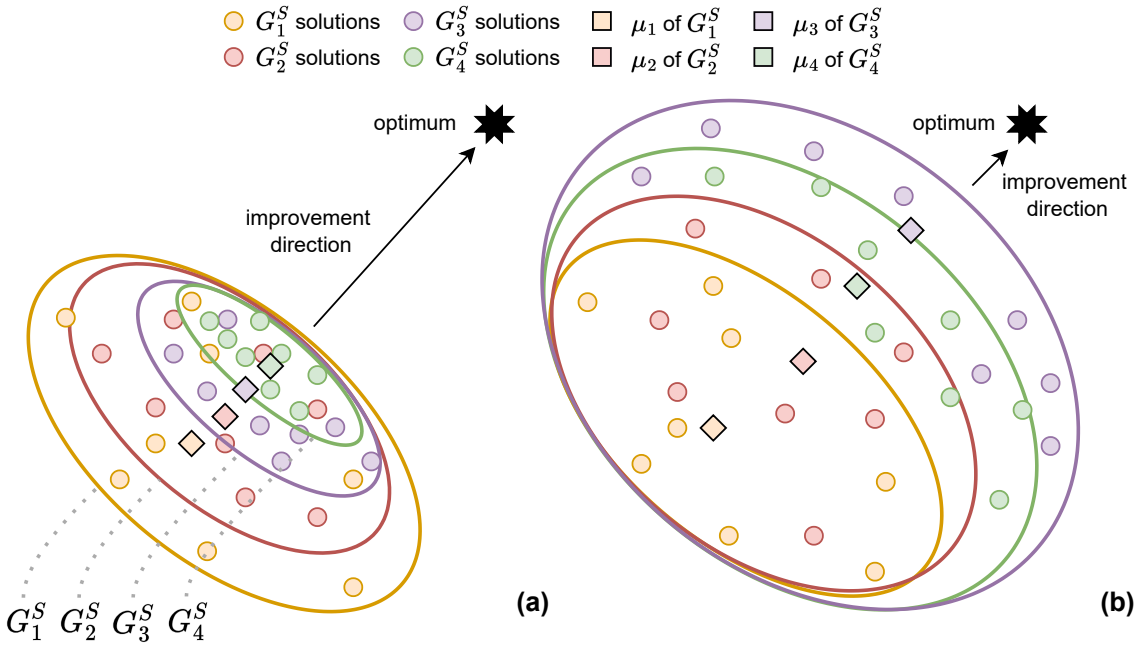


Figure 7.2: Two search processes are shown in which each generation G_t is obtained by sampling a probabilistic model estimated (a) from the best individuals of the previous generation or (b) from the best individuals of the previous l generations. The means and the search area defined by the solutions of each generation G_t are represented by squares and ellipses, respectively, on the two dimensional landscape.

advantageous, may cause EGNA to converge to local optima, which is a disadvantage. [Neumann and Cairns, 2012]. Figure 7.1 shows an example in which there are some local optima and a global optimum, and the EGNA tends to converge to one of the local optima due to the location of the initially sampled solutions in the early generations.

Furthermore, restricting the dependencies between variables linearly may imply a restriction in the way the search space is explored, since it will not be possible to consider more complex relationships either in the learning of the relationships or in the sampling of new solutions.

Therefore, the use of more complex models that do not assume a Gaussian density and do not assume linear dependencies between variables may allow the exploration of subspaces of the landscape without forcing the algorithm to choose one of the two areas of the space and thus avoid falling into local optima solutions. This idea of independently exploring different areas of the landscape was previously developed by Pelikan and Goldberg [Pelikan and Goldberg, 2000] by restricting the search space in subareas using K-means clustering and exploring them with EDAs, as well as by Peña, Lozano and Larrañaga [Peña et al., 2002], who used mixtures of Gaussians to model the population in each iteration. In this chapter, we present SPEDA as a generalization of the previous approaches where the use of mixtures is extended to the use of KDEs, but only in those variables that do not fit a Gaussian.

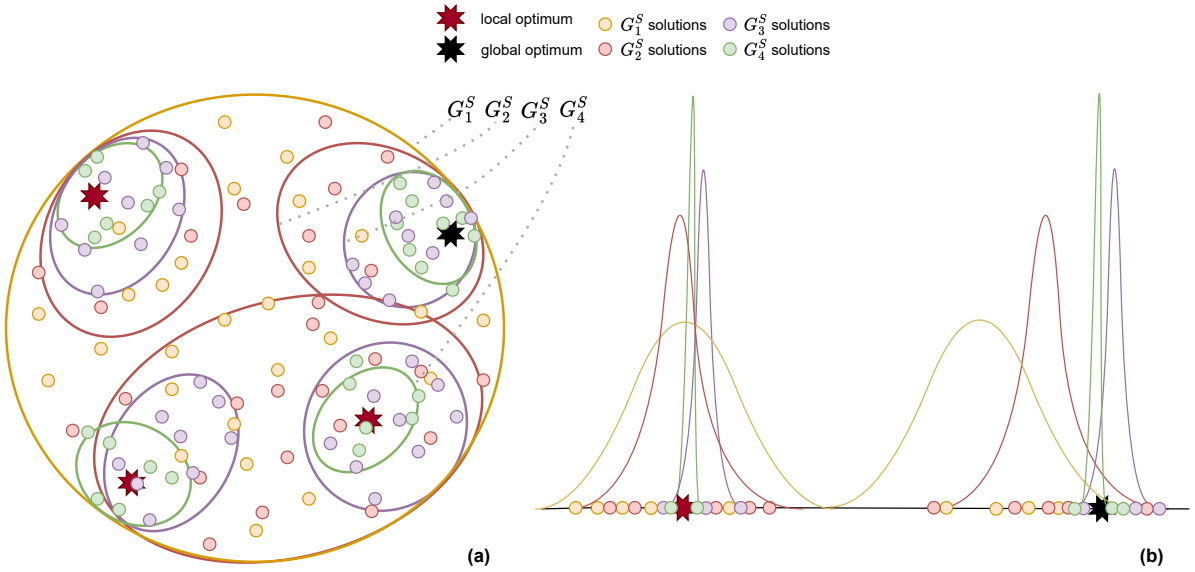


Figure 7.3: A search space iteratively approximated by SPEDA in two (a) and one (b) dimensions is shown, where a single global optimum (black star) and some local optima (red stars) exist. The search space area defined by the solutions of each generation G_t are represented by ellipses and a density function on the two and one dimensional landscapes, respectively.

7.2.2 SPEDA

In this section we present our approach to address the limitations of the traditional EGNA introduced in Section 7.2.1, thus improving the state of the art of EDAs for continuous optimization. Algorithm 7 shows the outline of SPEDA, where SPBNs are used as a more complex probabilistic model (Lines 14-16) and the new generations are sampled (Line 17) considering information learned from more than one past iteration (Lines 8-12).

The state of the art of EDAs present different ways of initializing the algorithm, such as using a dataset as the initial generation [Soloviev et al., 2022b] or, more commonly, initializing the algorithm from a set of solutions sampled from different possible probability distributions, such as a uniform distribution or a Gaussian distribution. In the case of SPEDA, it is not desired to bias the decisions to be made by the algorithm by defining a probability distribution over the solutions from which the algorithm is initialised. If the algorithm is initialized from a population that is randomly sampled from Gaussian distributions, successive generations of the algorithm will most likely also fit Gaussians, as will the relationships between the variables. For this reason, the algorithm is initialized from a set of uniformly sampled solutions in the search space defined by the problem to be optimized (Algorithm 7, Line 1). Each variable is independently sampled from a uniform distribution, $X_i \sim \mathcal{U}(a_i, b_i)$, where a_i and b_i are the minimum and maximum bounds of X_i in the optimization landscape.

Section 7.2.1 explains some of EGNA's limitations as a result of the Gaussianity assumption in GBNs during runtime. SPEDA overcomes this deficiency by using a semiparametric

Algorithm 7 SPEDA

Input: Population size N, selection ratio α, archive length l, cost function g
Output: Best solution $\boldsymbol{x}^*$ and cost $g(\boldsymbol{x}^*)$ found

1: $G_0 \leftarrow N$ individuals randomly sampled
2: $i \leftarrow 0$
3: $A^t = \emptyset$
4: **for** $t = 1, 2, ...$ until stopping criterion is met **do**
5: Evaluate G_{t-1} according to cost function g
6: Update best solution $\boldsymbol{x}^*$ obtained and compute $g(\boldsymbol{x}^*)$
7: $G^S_{t-1} \leftarrow$ Select top $\lfloor \alpha N \rfloor$ individuals from G_{t-1}
8: **if** $i < l$ **then**
9: $A^{t-1} \leftarrow A^{t-1} \cup G^S_{t-1}$
10: $i \leftarrow i + 1$
11: **else**
12: $A^{t-1} \leftarrow A^{t-1} \cup G^S_{t-1} \backslash G^S_{t-l-1}$
13: **end if**
14: $G_{t-1} \leftarrow$ Structure learning using HC (Algorithm 2)
15: $\boldsymbol{\Theta}_{t-1} \leftarrow$ Estimate parameters of model (Section 2.3)
16: $f_{t-1}(\cdot) \leftarrow (G_{t-1}, \boldsymbol{\Theta}_{t-1})$ Build probabilistic model
17: $G_t \leftarrow$ Sample N individuals from $f_{t-1}(\cdot)$
18: **end for**
19: **return** Best solution $\boldsymbol{x}^*$ obtained and $g(\boldsymbol{x}^*)$

Bayesian network. As a consequence, SPBNs allow the existence of nonlinear dependencies between variables. The key benefit is that SPEDA determines when nodes or dependencies between variables must be estimated by a KDE. If all variables and the CPDs that define the relationships found between the variables fit a Gaussian distribution, SPEDA will learn a GBN, which will make the algorithm procedure easier. Otherwise, an SPBN with some KDE variables will be learned. SPBN learning (Algorithm 7, Lines 14-15) involves the estimation of the parameters and structure learning. For parameter estimation, the relationships between variables that are linearly Gaussian are estimated using log-likelihood maximization, while those that are CKDEs are estimated using the normal reference rule, as detailed in Section 2.3. For structure learning, a modified version of the HC is executed in each iteration, which is detailed in Atienza et al. [2022b]. After learning the SPBN, the joint probability distribution represented by the probabilistic model is sampled to generate new solutions (Line 15). The sampling process (Line 17) is implemented using the probabilistic logic sampling method [Henrion, 1988], where the nodes are sampled in a forward direction following an ancestral order (from the parents to the children of the graph) using the evidence of the already sampled parents of the nodes.

The use of KDEs allows the parallel exploration of different areas of the search space that have a high probability density, which is unfeasible with the use of GBNs due to the assumption of (unimodal) Gaussianity. This implies that future generations may be explored and sampled in subspaces with high probability and may change during runtime. The expected behaviour of

both approaches are shown in Figure 7.1 and Figure 7.3, where a situation with several local optima is illustrated. While EGNA decides to exploit the search space area of a local optimum in Figure 7.1, SPEDA is able to simultaneously explore all local optima, and converge to the global optimum in Figure 7.3. Note that SPEDA in the second generation in the two dimensional landscape generalizes two of the local optima into a Gaussian, but in future generations, it decides to independently explore each of the local optima.

Several studies regarding premature convergence in EDAs for continuous optimization have been mentioned in this chapter. In the case of SPEDA, we propose that the SPBN used as the main engine of the algorithm is updated by considering the best $\lfloor \alpha N \rfloor$ solutions from each of the previous l generations (Algorithm 7, Line 12). Thus, in each iteration, SPEDA estimates the SPBN from A^{t-1} (Lines 8-12):

$$A^{t-1} = G^S_{t-1} \cup G^S_{t-2} \cup \cdots \cup G^S_{t-l}$$

where G_t has been truncated to a size of $\lfloor \alpha N \rfloor$ by selecting the best solutions according to a cost function (Line 7). Figure 7.2 shows a comparison of the performance of the traditional EGNA approach (a) and the performance when considering more than the very last generation (b). Note that a search direction is established considering the best individuals of the previous l generations. With this approach, it is expected that the new solutions sampled from the learned probabilistic model $f_{t-1}(\boldsymbol{x})$ will be located in a landscape that is not in a previously explored area, but in the desired search direction determined by the information gained in earlier iterations.

Finding a balance between exploration and exploitation in the landscape is required to design an algorithm that does not converge to local optimal solutions [Črepinšek et al., 2013]. Figure 7.2 illustrates how the behaviour of the standard EGNA favour an algorithm that exploits the search zones more than it explores the search space, creating an imbalance between these two characteristics. Nevertheless, the combination of SPBNs and the use of information gained from previous generations gives SPEDA a trade-off between both traits, since KDEs allow several zones of the space to be explored simultaneously and to be found by determining a search direction during the process, allowing each zone to be exploited independently.

7.3 Experimental results

In this section, we show the results of comparing our approach with some state-of-the-art optimizers in continuous environments on some well-known benchmarks, and in a real world portfolio optimization problem. We also report on the complexity and time analysis of our approach.

Eight different algorithms are used in the experimental comparison: EMNA, EGNA, SPEDA, m_KEDA, CMA-ES [Hansen, 2006], JADE [Zhang and Sanderson, 2009], SHADE [Tanabe and Fukunaga, 2013] and L-SHADE [Tanabe and Fukunaga, 2014]. EMNA and EGNA were chosen as continuous multivariate EDAs so that the results can be compared to those obtained with SPEDA. The pseudocode used for EGNA is fairly similar to that proposed for SPEDA where the probabilistic model, an SPBN for the case of SPEDA and a GBN for the case

of EGNA, is updated with the best solutions obtained in the previous l generations. Note that the traditional EGNA does not consider this archive-based approach, but it has been adapted to perform a fair comparison with SPEDA. The pseudocode of EMNA has been implemented traditionally, where in each iteration a multivariate Gaussian is estimated from the best individuals of the last generation. It is interesting to determine whether SPEDA significantly improves the results of EGNA and EMNA as another optimization strategy for continuous environments. The comparison also includes a multivariate version of KEDA (m_KEDA), where all the nodes and dependencies are restricted to be estimated using KDE. The algorithm shares the archive characteristic proposed for SPEDA and EGNA, so m_KEDA can be considered a particular case of SPEDA, where Gaussian variables are forbidden. The number of folds during the SPBN learning (K in Equation (2.12)) for the SPEDA and m_KEDA approaches has been set to $k = 10$, as proposed in the original work [Atienza et al., 2022b]. Because CMA-ES is a common comparison tool for this class of algorithms, its results are also included. The JADE, SHADE and L-SHADE algorithms were selected as members of the family of evolutionary algorithms in the area of differential evolution. Indeed, it has been widely used in recent years for real optimization tasks.

	EMNA	EGNA	SPEDA	m_KEDA	CMA-ES	JADE	SHADE	L-SHADE
N	300	300	300	300	4log(D)	300	300	300
α	0.4	0.6	0.4	0.4	-	-	-	-
l	-	10	15	15	-	-	-	-
c	-	-	-	-	-	0.1	-	-
p (%)	-	-	-	-	-	10	-	-

Table 7.1: Best parameter configuration found for the algorithms after parameter tuning. The parameters include the population size N, the ratio of solutions selected from each generation $\alpha \in [0, 1]$, the archive length l, and the adaptation rates of the self-adaptive parameters (c) and the greediness of the mutation strategy (p) of the JADE algorithm.

The test benchmarks are listed and characterized in the Appendix A, and were obtained from IEEE CEC2014 [Liang et al., 2013] and IEEE CEC2017 [Wu et al., 2017]. All tests are single-objective optimization problems with different difficulties grouped into unimodal-multimodal, separable-nonseparable[3] functions.

Table 7.1 shows the best parameter configuration found for the algorithms compared in this section for this set of experiments. Note that the maximum number of fitness evaluations has been set to $10000d$, where d is the dimension of the benchmark optimization problem. In this chapter, we analyze the cases of $d = 30$ and $d = 50$. Each algorithm was independently run 25 times for each benchmark and dimension. The results analyzed in this section show the mean and standard deviation of the function error value (FEV) of the 25 independent runs, where the FEV is defined as the difference between the costs of the (known) optimal solution $\boldsymbol{x}''$ and the best achieved solution $\boldsymbol{x}^*$: $FEV(\boldsymbol{x'}) = g(\boldsymbol{x'}) - g(\boldsymbol{x}'')$. Note that a difference lower than $1e-8$ is reported as zero in the experimental results, and the objective is to minimize the FEV.

[3]A function is said to be separable if it can be expressed as a mathematical operation between functions of smaller dimension.

All the experiments and algorithms were implemented in Python. The code of SPEDA, m_KEDA and benchmark implementations will be merged in the near future into the EDAspy Python package that is publicly available in a GitHub repository[4].

All the experiments and code are already available in a GitHub repository[5]. The experiments were conducted on an Ubuntu 20 machine with an Intel Core i7-6700K processor, 32 GB of RAM, and an AMD Radeon RX 460 graphic card.

Benchmark	EMNA	EGNA	SPEDA	m_KEDA	CMA-ES	JADE	SHADE	L-SHADE
cec14_1	7.6e6 ± 7.7e5	1.5e5 ± 3.3e4	0.000 ± 0.000	2.4e4 ± 2.0e4	1.4e3 ± 1.4e3	6.6e5 ± 5.8e4	4.1e4 ± 1.1e4	0.000 ± 0.000
cec14_2	3.8e7 ± 4.5e6	1.1e7 ± 3.5e6	0.000 ± 0.000	1.7e5 ± 1.0e5	0.000 ± 0.000	1.961 ± 5.732	0.000 ± 0.000	0.000 ± 0.000
cec14_3	3.1e4 ± 1.9e3	4.741 ± 14.86	0.000 ± 0.000	0.960 ± 1.310	0.000 ± 0.000	1.813 ± 3.123	0.000 ± 0.000	2.4e3 ± 0.000
cec14_4	29.25 ± 1.232	60.48 ± 15.99	27.83 ± 0.643	28.12 ± 0.090	10.82 ± 14.28	30.38 ± 19.91	28.35 ± 32.10	6.770 ± 32.10
cec14_5	20.96 ± 0.071	20.97 ± 0.043	20.18 ± 0.024	321.2 ± 0.060	32.58 ± 0.741	20.83 ± 0.113	320.4 ± 0.060	320.6 ± 0.060
cec14_6	0.034 ± 0.012	0.011 ± 0.031	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_7	11.23 ± 12.28	0.353 ± 0.032	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_8	158.7 ± 6.490	7.862 ± 6.742	4.671 ± 1.455	225.8 ± 0.920	65.61 ± 4.991	57.07 ± 7.601	146.4 ± 16.27	36.07 ± 16.27
cec14_9	157.2 ± 9.191	126.2 ± 7.841	163.2 ± 8.161	248.3 ± 10.54	187.2 ± 4.751	186.1 ± 3.091	184.8 ± 1.940	182.7 ± 1.940
cec14_10	4.1e3 ± 360.9	4.9e3 ± 276.9	1.3e3 ± 128.6	8.8e3 ± 40.85	7.5e3 ± 484.2	6.1e3 ± 1.1e3	7.1e3 ± 758.4	3.6e3 ± 758.4
cec14_11	6.2e3 ± 846.4	5.3e3 ± 263.3	5.0e3 ± 167.2	9.2e3 ± 798.2	1.1e4 ± 697.3	9.4e3 ± 1.6e3	9.6e3 ± 1.9e3	1.7e3 ± 1.9e3
cec14_12	2.541 ± 0.232	2.521 ± 0.334	2.273 ± 0.212	3.860 ± 0.230	0.040 ± 0.031	0.071 ± 0.033	0.040 ± 0.010	0.310 ± 0.010
cec14_13	0.481 ± 0.032	0.283 ± 0.031	0.242 ± 0.010	0.530 ± 0.000	0.291 ± 0.080	0.840 ± 0.161	0.380 ± 0.070	0.190 ± 0.070
cec14_14	0.322 ± 0.011	0.672 ± 0.874	0.284 ± 0.014	0.410 ± 0.010	0.481 ± 0.184	0.694 ± 0.331	0.284 ± 0.020	0.320 ± 0.020
cec14_15	6.4e5 ± 1.5e5	9.7e9 ± 1.1e6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_16	15.00 ± 0.001	12.34 ± 0.290	12.24 ± 0.151	25.50 ± 17.68	13.68 ± 0.291	13.73 ± 0.022	12.87 ± 0.780	12.80 ± 0.780
cec14_17	2.5e4 ± 7.4e3	5.4e8 ± 5.9e8	5.470 ± 1.420	66.50 ± 75.66	181.4 ± 178.5	3.000 ± 1.540	0.010 ± 0.010	0.000 ± 0.000
cec14_18	1.5e6 ± 3.5e5	7.8e9 ± 3.8e9	14.69 ± 2.110	111.0 ± 124.4	128.4 ± 78.19	81.30 ± 1.970	1.720 ± 0.200	0.270 ± 0.200
cec14_19	211.9 ± 37.9	1.0e9 ± 8.9e8	71.43 ± 1.510	100.0 ± 1.410	69.31 ± 2.080	66.75 ± 0.950	67.47 ± 0.520	66.87 ± 0.520
cec14_20	4.5e8 ± 2.2e8	9.9e9 ± 0.000	6.560 ± 1.310	50.50 ± 70.00	85.45 ± 59.54	33.40 ± 1.560	0.460 ± 0.080	0.350 ± 0.080
cec14_21	1.4e4 ± 5.7e3	1.3e9 ± 1.1e7	1.580 ± 0.510	61.50 ± 54.45	3.790 ± 2.040	0.740 ± 0.210	0.540 ± 0.110	0.520 ± 0.110
cec14_22	2.1e7 ± 1.5e7	9.9e9 ± 0.000	24.41 ± 0.400	75.50 ± 88.39	11.42 ± 3.750	1.480 ± 0.520	0.320 ± 0.040	0.200 ± 0.040
cec14_23	1.0e4 ± 594.2	9.5e3 ± 1.260	9.5e3 ± 3.390	41.50 ± 153.4	9.5e3 ± 5.630	9.5e3 ± 5.870	9.5e3 ± 4.560	9.5e3 ± 4.560
cec14_24	6.3e3 ± 25.49	6.3e3 ± 177.8	6.1e3 ± 13.81	96.00 ± 74.95	6.1e3 ± 14.94	6.2e3 ± 39.61	6.1e3 ± 32.98	6.1e3 ± 32.98
cec14_25	5.2e3 ± 0.240	5.3e3 ± 163.7	5.2e3 ± 0.060	110.0 ± 57.98	5.2e3 ± 0.030	5.2e3 ± 0.230	5.2e3 ± 0.670	5.2e3 ± 0.670
cec14_26	1.1e4 ± 0.190	1.1e4 ± 907.9	1.1e4 ± 0.570	1.1e4 ± 13.44	1.1e4 ± 0.000	1.1e4 ± 0.320	1.1e4 ± 0.020	1.1e4 ± 0.020
cec14_27	1.1e4 ± 0.990	1.4e4 ± 1.3e3	1.1e4 ± 0.830	1.1e4 ± 28.99	1.1e4 ± 0.040	1.1e4 ± 1.700	1.1e4 ± 0.010	1.1e4 ± 0.010
cec14_28	1.2e4 ± 451.4	1.5e4 ± 8.2e2	1.1e4 ± 3.150	1.1e4 ± 64.35	1.1e4 ± 0.010	1.2e4 ± 21.63	1.1e4 ± 2.200	1.2e4 ± 2.200
cec14_29	4.5e5 ± 1.1e5	8.5e8 ± 6.5e8	7.9e3 ± 1.510	7.9e3 ± 50.20	7.9e3 ± 36.64	7.9e3 ± 1.290	7.9e3 ± 0.580	7.9e3 ± 0.580
cec14_30	3.5e7 ± 2.3e6	1.0e8 ± 0.000	8.4e3 ± 1.620	8.4e3 ± 126.5	8.4e3 ± 4.290	8.4e3 ± 7.150	8.4e3 ± 5.580	8.4e3 ± 5.580
cec17_1	3.8e7 ± 4.3e6	8.2e9 ± 3.1e9	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_2	42.19 ± 6.050	8.2e4 ± 9.6e3	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.2e4 ± 0.000
cec17_3	124.9 ± 0.260	1.2e4 ± 7.8e3	24.97 ± 0.310	101.0 ± 2.830	6.100 ± 1.830	15.72 ± 1.650	18.18 ± 0.840	15.24 ± 0.840
cec17_4	244.1 ± 10.40	8.8e4 ± 3.4e4	60.06 ± 9.250	78.01 ± 15.56	52.90 ± 11.70	184.6 ± 27.93	94.03 ± 7.660	9.910 ± 7.660
cec17_5	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	51.06 ± 67.88	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_6	1.8e3 ± 141.4	3.2e3 ± 90.44	166.2 ± 4.270	60.03 ± 12.73	51.40 ± 13.41	172.5 ± 17.55	98.23 ± 9.810	112.6 ± 9.810
cec17_7	1.2e3 ± 340.5	452.3 ± 45.42	78.90 ± 0.560	79.02 ± 28.28	91.23 ± 3.455	101.2 ± 4.455	89.45 ± 4.455	79.00 ± 0.000
cec17_8	2.840 ± 0.560	118.9 ± 3.340	0.000 ± 0.000	99.50 ± 13.44	5.100 ± 1.160	6.070 ± 1.210	7.180 ± 0.380	101.0 ± 0.380
cec17_9	7.0e3 ± 463.2	9.4e3 ± 3.0e3	6.2e3 ± 483.4	200.0 ± 141.4	1.1e3 ± 3.3e3	1.1e4 ± 4.1e3	1.1e4 ± 3.5e3	2.7e3 ± 3.5e2
cec17_10	978.7 ± 172.2	203.4 ± 203.2	5.835 ± 0.310	59.50 ± 55.86	47.33 ± 25.08	10.96 ± 1.890	10.12 ± 0.290	5.930 ± 0.490
cec17_11	5.4e6 ± 8.9e5	9.6e3 ± 5.7e3	0.000 ± 0.000	120.0 ± 141.4	3.580 ± 2.740	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_12	1.7e6 ± 4.3e5	7.7e3 ± 3.1e3	120.4 ± 5.122	139.5 ± 71.42	124.4 ± 58.86	26.84 ± 6.980	11.69 ± 1.220	4.390 ± 1.220
cec17_13	1.5e3 ± 718.5	1.3e3 ± 424.8	0.000 ± 0.000	105.5 ± 92.63	279.4 ± 196.4	10.59 ± 3.070	2.800 ± 0.720	0.000 ± 0.000
cec17_14	5.1e5 ± 1.9e5	2.2e3 ± 211.3	83.23 ± 4.556	95.59 ± 7.780	132.2 ± 167.1	13.25 ± 3.590	7.070 ± 0.230	2.370 ± 0.230
cec17_15	313.9 ± 78.40	234.1 ± 21.33	65.78 ± 34.34	84.33 ± 19.80	0.730 ± 0.190	2.030 ± 0.860	1.110 ± 1.190	0.530 ± 1.190
cec17_16	2.5e7 ± 1.2e7	1.2e3 ± 334.9	0.000 ± 0.000	77.06 ± 8.490	134.7 ± 80.60	9.680 ± 4.700	2.170 ± 0.380	0.010 ± 0.000
cec17_17	8.8e3 ± 5.2e3	982.2 ± 332.5	83.45 ± 3.450	90.51 ± 10.61	103.2 ± 65.50	1.980 ± 1.520	0.480 ± 0.020	0.500 ± 0.020
cec17_18	1.7e8 ± 5.7e7	1.2e4 ± 3.1e3	18.94 ± 2.344	20.62 ± 14.14	134.3 ± 51.24	67.84 ± 2.780	62.73 ± 0.290	60.90 ± 0.290
cec17_19	43.00 ± 5.420	23.45 ± 3.344	13.45 ± 3.344	20.71 ± 14.14	16.23 ± 4.450	4.510 ± 1.190	1.250 ± 0.180	0.250 ± 0.180

Table 7.2: Mean and standard deviation of FEV after 25 executions of all benchmark functions with 30 variables ($d = 30$) obtained from EMNA, EGNA, SPEDA, m_KEDA, CMA-ES, JADE, SHADE and L-SHADE algorithms. The best result for each benchmark is highlighted in blue.

[4] https://github.com/VicentePerezSoloviev/EDAspy

[5] https://github.com/VicentePerezSoloviev/SPEDA

7.3.1 Experimental results on 30-d benchmarks

Table 7.2 shows the mean and standard deviation of FEV found by each algorithm after 25 independent runs for each of the benchmarks with $d = 30$, where the best results found are highlighted in blue.

On the one hand, it is important to stress the improvement found by SPEDA compared to the other EDAs versions. In almost all functions, SPEDA finds a better solution than EMNA and EGNA with a lower standard deviation. Note that EGNA improves the results found by EMNA in nearly all the functions, and that the exclusive use of CKDEs (m_KEDA) in contrast to our approach has only improved the results in the case of the composition functions (cec14_23 - cec14_30).

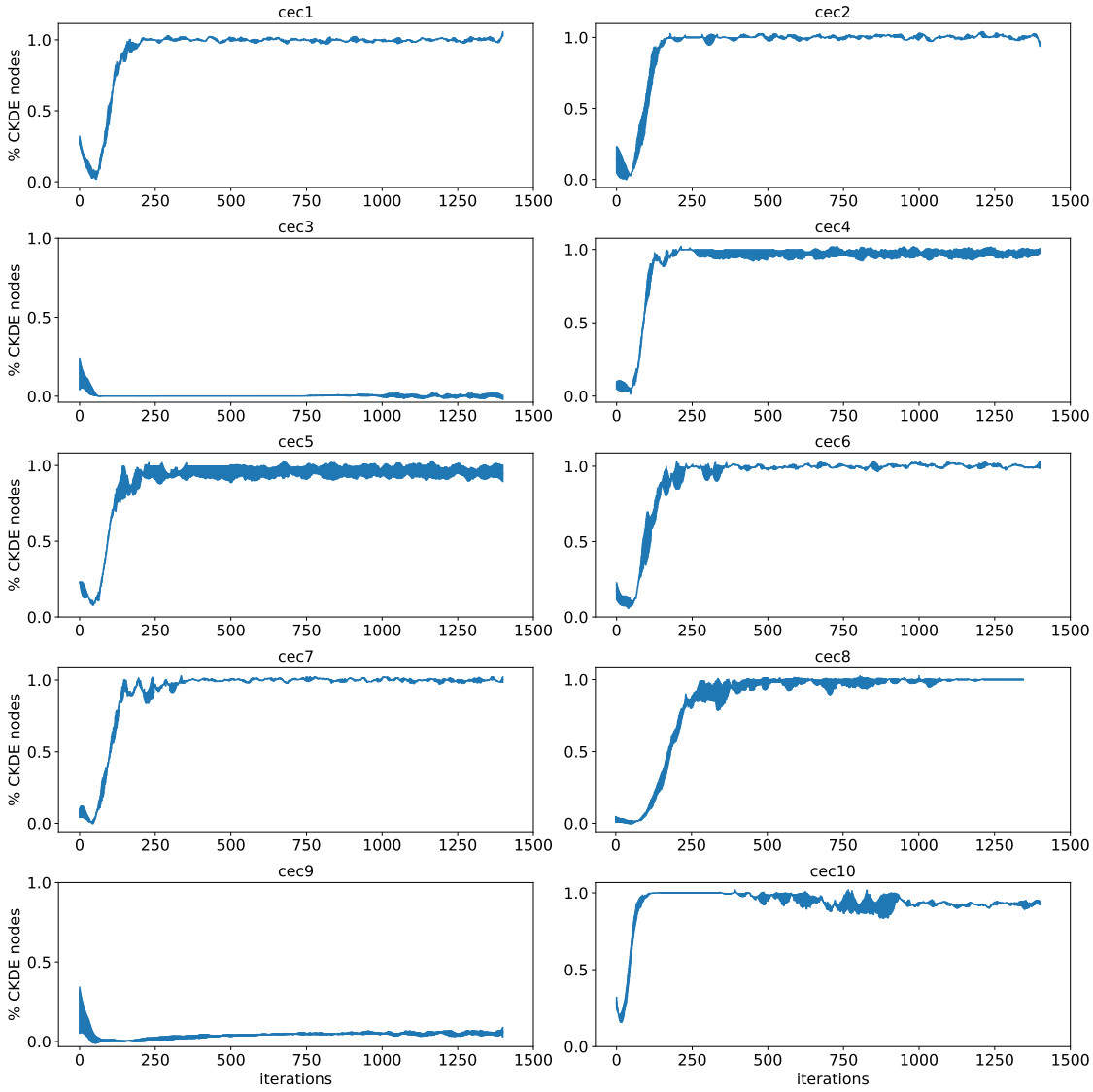


Figure 7.4: Percentage of CKDE nodes during runtime, mean and standard deviation of 25 independent executions. The experimental results for the first 10 benchmarks are shown with $d = 30$.

On the other hand, the clear competitor for SPEDA in this comparison is L-SHADE, which reaches the same solutions in some of the benchmarks and outperforms SPEDA in 19 out of 49 functions. The SHADE and JADE approaches also achieve competitive results compared to SPEDA, improving those found by SPEDA in 14 and 11 functions, respectively. CMA-ES and m_KEDA algorithms beat our approach in 5 and 9 benchmarks, respectively. Additionally, SPEDA achieves the lowest FEV standard deviation in most of the experiments.

In terms of the type of optimized functions, SPEDA is able to find the optimum in all runs for unimodal functions (cec14_1, cec14_2, cec14_3, cec17_1, cec17_2), while its competitors are not able to do so. For the cec14_4 benchmark, since there is a very narrow valley between the local and global optima, SPEDA seems not to have sufficiently exploited the search space, thus preventing it from beating CMA-ES and L-SHADE. It is worth mentioning the pair of benchmarks cec14_8 and cec14_9, as they are the same function, with the difference that one is separable (cec14_8) and the other is not (cec14_9). In this case, using a GBN instead of an SPBN improves the results for the nonseparable function. The pair cec14_10 and cec14_11 has the same feature, and for the nonseparable function (cec14_11), a similar result is obtained for both EGNA and SPEDA. The standard deviation for these two problems is high for all algorithms, but again the best result is that of SPEDA. However, in both pairs of benchmarks (cec14_8-cec14_9 and cec14_10-cec14_11), SPEDA improves the other algorithms for separable functions (cec14_8 and cec14_10). The cec14_12 benchmark is also nonseparable, and the algorithm that provides the best results is SHADE. As for hybrid functions (cec14_17 to cec14_22, and cec17_10 to cec17_19), the best results are found by SPEDA and L-SHADE approaches. In the case of composition functions (cec14_23 to cec14_30), which are multi-modal, SPEDA, CMA-ES and SHADE achieve the best results in most of the benchmarks, only beaten by m_KEDA. Finally, regarding the optimization of functions where the number of local optima is large (cec14_8 to cec14_11, cec17_3 to cec17_5 and cec17_7 to cec17_9), SPEDA is the best performing algorithm. Indeed, it is able to find the best solutions in 5 out of the 10 functions with this feature, followed by L-SHADE (3 out of 10) and CMA-ES (2 out of 10). This suggests that SPEDA is able to avoid local optima better than its competitors. Thus, SPEDA and L-SHADE seem to ensure better results in most of the benchmarks compared to the rest of the algorithms, finding the best results both in 26 out of the 49 benchmarks.

To conclude this comparison for the 30−dimensional experiments, in the case of the family of EDAs, using a more complex probabilistic model such as SPBN improves the results compared to those of the multivariate Gaussian and the GBNs. SPEDA is a competitive approximation compared to other state-of-the-art EDAs, CMA-ES and differential evolution approaches. Moreover it is able to provide results with a low variance after different independent executions. In addition, the experiments have shown that SPEDA always improves the results for at least unimodal and separable functions.

The results shown in Table 7.2 have been statistically analyzed. Figure 7.5 shows the credibility intervals (5% and 95%) and expected probabilities of each algorithm being the winner under the posterior distribution calculated in the Bayesian analysis using the Plackett-Luce model [Calvo et al., 2019]. The plot shows that L-SHADE and SPEDA are the most probable winners where the former shows a lower associated uncertainty. EMNA and EGNA are

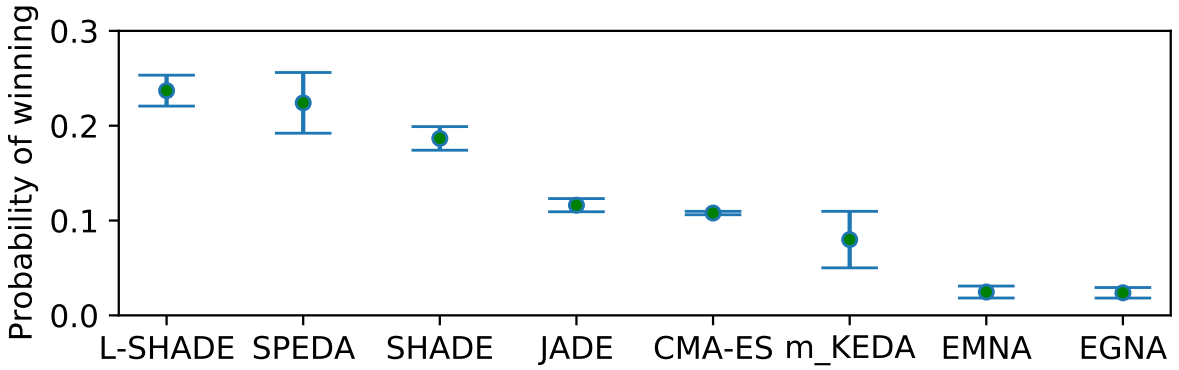


Figure 7.5: Credible intervals (5% and 95% quantiles) and expected probability of winning (green dots) for results shown in Table 7.2.

the less likely approaches to be the winners achieving a similar expected probability and associated uncertainty; thus we conjecture that both approaches perform similar. m_KEDA is the approach with higher uncertainty, but with less chances to be the winner. The plot corroborates the results shown in Table 7.2 where JADE and CMA-ES achieve a very similar performance, being the latter the one with less uncertainty. SHADE chases SPEDA in the ranking of expected probabilities to be the winner approach.

Analyzing the behaviour of SPEDA during runtime, it is observed that the initialization of the algorithm has a significant impact on the predominant probabilistic model used during runtime. When starting from Gaussian distributions, SPEDA behaves in a way that favours Gaussians over CKDEs from the outset. However, when starting from a uniform probability distribution, as detailed in Section 7.2.2, SPEDA chooses the probabilistic model to use. The number of nodes that are fitted by CKDE increases as the algorithm's evolve, as shown in Figure 7.4, where the mean and standard deviation of the percentage of CKDE-estimated nodes in each iteration are shown for the first 10 benchmarks. In most cases, SPEDA tends to adjust all the nodes by CKDE. Note that in the iterations where 0% of the nodes are estimated by CKDE, the learned probabilistic model is a GBN, where the optimization process is the same as in EGNA but conditioned to the previous iterations, where an SPBN was learned. Figure 7.4 shows that atypically, for the cec14_3 and cec14_9 benchmarks, nearly all nodes are Gaussian from the beginning of the algorithm execution, and this directly affects the results found. In cec14_3, the fact that an SPBN rather than a GBN was used in the first iterations of SPEDA suggests that SPEDA and EGNA are positioned in different areas of the search space, with the area explored by SPEDA being the area of the global optimum. A similar situation occurred for cec14_9, in this case in favour of EGNA. Similar results were obtained for the rest of the benchmarks, but were not included in Figure 7.4 for aesthetic reasons. In the experiments we have seen a tendency for all functions to converge to 100% of CKDE-estimated nodes. This can be explained because individuals sampled from Gaussians and then selected according to the objective function are more likely to be fitted by a KDE than vice versa.

For deeper insights in SPEDA performance, we aim to analyse the potential ability of the approach to avoid the local optima in the landscape. If the variance of the solutions in the

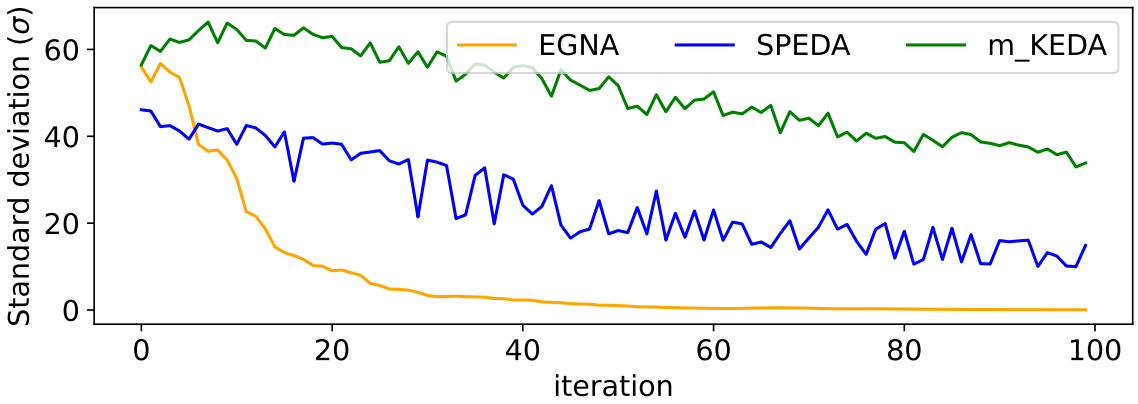


Figure 7.6: Mean standard deviation (Y-axis) between the best solutions in the same iteration during runtime (X-axis) after executing EGNA, SPEDA and m_KEDA approaches 25 independent times, for the cec14_4 benchmark ($d = 30$). Note that this plot represents one variable out of d.

same iteration of the algorithm tends to decrease over the runtime, then it may suggest that the algorithm is converging to a smaller area, which is probably contained in the area of the previous generation. However, if the variance increases and decreases over the runtime, it can be said that the algorithm is seeking a balance between exploration and exploitation of the search space, and most likely will be able to avoid local optima. This reduction of variance between individuals is related to genetic drift [Doerr and Zheng, 2020], which is present in EDAs.

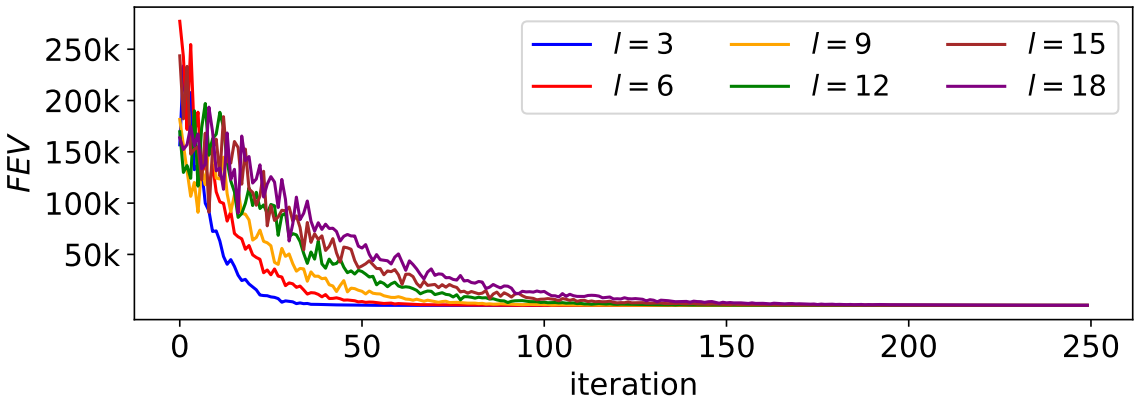


Figure 7.7: Mean best cost found (FEV) by SPEDA for the cec14_3 for different archive lengths (l) during runtime after executing each experiment 25 independent times.

Figure 7.6 shows a univariate analysis of the variance reduction of the best solutions along the runtime for the EGNA, SPEDA and m_KEDA approaches, for the cec14_4 benchmark ($d = 30$). It is observed that EGNA early reduces the variance to zero, leading to a nearly monotonic decreasing shape. This might reveal convergence to a local optima solution. Nevertheless, SPEDA and m_KEDA suggest a decreasing tendency, but much slower than EGNA. Several ups and downs during the runtime can be appreciated, which might suggest that our approach is able to avoid local optima by using CKDE nodes (Figure 7.4) and

exploring more than one area at the same time. Table 7.2 shows that both approaches converge to better solutions than EGNA approach for this benchmark. Note that, although Figure 7.6 shows the performance of a single variable, the rest of features have similar performance (not shown).

Figure 7.7 analyses the convergence speed compared to the archive length (l) for the cec14_3 benchmark. The higher the l value, the slower the speed of convergence. Although for this case SPEDA reaches the global optimum regardless of l, after the hyper-parameter tuning we have decided to use $l = 15$ as the general optimum parameter for all benchmarks in CEC2014 and CEC2017 for both $d = 30$ and $d = 50$.

7.3.2 Experimental results on 50-d benchmarks

Table 7.3 shows the mean and standard deviation of the FEV of each algorithm after 25 independent runs for each of the benchmarks with $d = 50$, where the best results for each function are highlighted in blue.

As in the 30-dimensional case, SPEDA improves the results found by EMNA, EGNA and m_KEDA, although m_KEDA provides competitive results compared to our approach. Although it is not as remarkable as in the case of $d = 30$, SPEDA provides the lowest variance among the results found in almost all benchmarks, since the use of the CKDEs reduces it. This is the case in m_KEDA, which also shows a low standard deviation, compared to the other EDAs. The results show that SPEDA is able to converge to the best solutions in 30 out of the 49 functions. The main competitor for the SPEDA approach in this comparison is L-SHADE, reaching the best solution in 22 out of the 49 functions. Moreover, L-SHADE outperforms the results found by SPEDA in 17 functions, and SPEDA outperforms L-SHADE in 18 functions.

SPEDA achieves the best results for all the unimodal functions except for the cec14_2, in which CMA-ES, SHADE and L-SHADE win it. Regarding the separable functions (cec14_8 and cec14_10), SPEDA converges to the best solution in all runs for both cases. Regarding the hybrid functions, L-SHADE approach is the best in this characteristic, followed by our proposal. SPEDA is the only approach able to find the best results in composition functions, as in the 30−dimensions case. Note that, in the case of $d = 30$, m_KEDA achieved good results for these functions, while no such results are shown in the case of $d = 50$. This may suggest that the combination of Gaussian and CKDE nodes scales better from the optimization point of view, compared to the exclusive use of CKDEs. Finally, regarding the optimization of functions where the number of local optima is large, SPEDA is the best performing algorithm (6 out of 10), as in the 30−dimensions case, followed by CMA-ES (5 out of 10).

The results shown in Table 7.3 have been statistically analyzed. Figure 7.8 shows the credibility intervals (5% and 95%) and expected probabilities of each algorithm being the winner under the posterior distribution calculated in the Bayesian analysis using the Plackett-Luce model [Calvo et al., 2019]. It is shown that SPEDA is the best approach, where its lower bound is higher than the upper one of its competitors. SPEDA is followed by L-SHADE and SHADE, where the latter is the one with highest uncertainty. JADE and CMA-ES imitate the results

analyzed for 30 dimensions. The approaches that have the least chances of being the winners are again m_KEDA, EMNA and EGNA, where in this analysis m_KEDA achieves very low uncertainty.

Benchmark	EMNA	EGNA	SPEDA	m_KEDA	CMA-ES	JADE	SHADE	L-SHADE
cec14_1	1.1e7 ± 1.0e6	2.7e7 ± 3.e06	6.2e3 ± 1.5e4	3.6e9 ± 1.5e8	4.1e5 ± 1.4e5	6.2e6 ± 2.7e6	5.1e6 ± 1.9e6	4.8e8 ± 1.9e6
cec14_2	1.1e7 ± 3.1e6	2.3e3 ± 1.1e3	7.000 ± 14.78	2.8e7 ± 2.8e6	0.000 ± 0.000	1.6e4 ± 1.2e4	0.000 ± 0.000	0.000 ± 0.000
cec14_3	3.9e4 ± 1.5e3	0.440 ± 1.670	0.000 ± 0.000	3.5e5 ± 2.3e5	1.1e3 ± 558.6	8.2e3 ± 4.5e3	477.7 ± 409.1	3.8e4 ± 409.1
cec14_4	210.7 ± 3.340	171.2 ± 14.23	151.2 ± 15.08	76.16 ± 4.000	43.40 ± 17.65	77.91 ± 38.00	158.3 ± 54.14	96.32 ± 54.14
cec14_5	21.16 ± 0.030	21.17 ± 0.030	21.13 ± 0.060	321.2 ± 0.040	32.15 ± 0.640	21.01 ± 0.120	320.9 ± 0.050	320.8 ± 0.050
cec14_6	0.030 ± 0.000	0.010 ± 0.010	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_7	0.960 ± 0.090	2.100 ± 5.330	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_8	114.5 ± 9.080	339.6 ± 11.82	180.6 ± 10.52	473.5 ± 10.33	302.1 ± 7.802	188.3 ± 13.52	260.3 ± 13.03	195.9 ± 13.03
cec14_9	335.3 ± 17.65	338.6 ± 11.03	247.5 ± 16.05	494.9 ± 29.73	425.8 ± 11.14	416.1 ± 21.13	388.3 ± 54.55	88.18 ± 54.55
cec14_10	8.6e3 ± 911.3	1.0e4 ± 410.0	6.1e3 ± 288.1	1.7e4 ± 468.9	1.8e4 ± 2.3e3	1.1e4 ± 4.1e3	1.5e4 ± 3.6e3	1.8e4 ± 3.6e3
cec14_11	1.7e4 ± 782.1	1.0e4 ± 375.1	1.2e4 ± 73.73	1.6e4 ± 469.5	5.9e3 ± 847.0	1.1e4 ± 1.8e3	8.8e3 ± 1.2e3	8.3e3 ± 1.2e3
cec14_12	3.610 ± 0.290	3.460 ± 0.260	3.430 ± 0.590	4.470 ± 0.570	0.020 ± 0.010	0.130 ± 0.080	0.020 ± 0.020	0.530 ± 0.020
cec14_13	0.780 ± 0.020	0.520 ± 0.050	0.400 ± 0.020	0.770 ± 0.070	0.600 ± 0.090	1.080 ± 0.120	0.680 ± 0.090	0.580 ± 0.090
cec14_14	0.720 ± 0.040	0.540 ± 0.110	0.440 ± 0.020	0.920 ± 0.250	0.620 ± 0.280	0.990 ± 0.330	0.450 ± 0.140	0.480 ± 0.140
cec14_15	2.3e6 ± 4.6e5	9.3e7 ± 1.4e6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec14_16	25.00 ± 0.010	22.33 ± 0.670	22.23 ± 0.260	23.59 ± 0.030	22.46 ± 0.460	22.36 ± 0.020	24.45 ± 1.231	23.23 ± 0.012
cec14_17	1.4e5 ± 3.4e4	960.5 ± 80.99	34.90 ± 3.320	5.6e3 ± 191.9	181.4 ± 178.5	10.15 ± 3.680	15.28 ± 2.270	0.000 ± 0.000
cec14_18	7.4e6 ± 8.3e5	810.4 ± 40.94	68.72 ± 6.880	5.0e3 ± 2.5e3	128.4 ± 78.19	12.83 ± 6.440	35.79 ± 3.620	1.130 ± 3.620
cec14_19	1.1e3 ± 217.5	337.7 ± 80.03	209.2 ± 0.820	263.4 ± 69.36	69.31 ± 2.080	192.2 ± 1.890	198.8 ± 0.710	194.8 ± 0.710
cec14_20	195.0 ± 20.31	99.50 ± 0.000	34.97 ± 5.800	64.32 ± 1.345	85.45 ± 59.54	6.220 ± 2.990	14.08 ± 1.560	0.480 ± 1.560
cec14_21	8.5e4 ± 1.6e4	200.3 ± 12.34	8.150 ± 0.600	12.10 ± 0.812	3.790 ± 2.040	7.280 ± 0.900	4.690 ± 0.770	1.840 ± 0.770
cec14_22	6.2e8 ± 4.3e8	1.1e3 ± 12.12	25.62 ± 0.200	17.32 ± 0.412	11.42 ± 3.750	2.750 ± 0.750	1.690 ± 0.130	0.430 ± 0.130
cec14_23	9.5e3 ± 1.141	1.0e4 ± 525.2	9.4e3 ± 0.130	1.1e4 ± 683.4	9.5e3 ± 5.630	9.6e3 ± 10.65	9.5e3 ± 6.240	9.6e3 ± 6.240
cec14_24	6.7e3 ± 31.31	6.6e3 ± 238.9	6.6e3 ± 0.520	6.6e3 ± 2.410	6.6e3 ± 14.94	6.6e3 ± 27.86	6.6e3 ± 65.65	6.6e3 ± 65.65
cec14_25	5.2e3 ± 1.040	5.2e3 ± 72.15	5.2e3 ± 0.090	5.7e3 ± 41.11	5.2e3 ± 0.030	5.2e3 ± 1.600	5.2e3 ± 1.340	5.2e3 ± 1.340
cec14_26	1.1e4 ± 0.101	1.1e4 ± 0.211	1.0e4 ± 0.090	1.1e4 ± 51.29	1.0e4 ± 0.000	1.104 ± 1.220	1.0e4 ± 0.430	1.1e4 ± 0.430
cec14_27	1.1e4 ± 1.170	1.1e4 ± 4.801	1.1e4 ± 0.190	2.1e4 ± 885.4	1.1e4 ± 0.040	1.1e4 ± 11.61	1.1e4 ± 3.600	1.1e4 ± 3.840
cec14_28	1.1e4 ± 35.14	1.2e4 ± 35.14	1.1e4 ± 10.77	1.2e3 ± 182.7	1.2e4 ± 0.010	1.1e4 ± 24.67	1.2e4 ± 9.440	1.3e4 ± 9.440
cec14_29	1.7e6 ± 2.3e5	6.9e4 ± 1.2e3	7.9e3 ± 1.670	8.1e3 ± 0.400	7.9e3 ± 36.64	7.9e3 ± 2.670	7.9e3 ± 0.470	7.9e3 ± 0.000
cec14_30	5.0e8 ± 3.0e6	5.6e4 ± 237.1	8.4e3 ± 139.9	8.4e3 ± 79.91	8.4e3 ± 4.290	8.4e3 ± 2.540	8.4e3 ± 0.100	8.4e3 ± 0.100
cec17_1	9.8e7 ± 7.2e5	1.0e6 ± 2.1e5	0.000 ± 0.000	2.9e7 ± 4.7e4	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_2	213.98 ± 3.51	9.5e4 ± 3.0e4	0.000 ± 0.000	3.1e5 ± 6.5e3	0.000 ± 0.000	302.6 ± 256.1	0.000 ± 0.000	101.1 ± 0.010
cec17_3	156.57 ± 0.74	8.3e4 ± 9.4e4	120.4 ± 7.860	85.40 ± 2.640	6.040 ± 1.830	92.10 ± 34.02	107.3 ± 39.84	2.5e3 ± 39.84
cec17_4	480.6 ± 2.620	8.5e4 ± 1.6e4	348.2 ± 1.250	496.5 ± 14.93	52.93 ± 11.69	389.1 ± 32.90	334.0 ± 6.920	89.80 ± 6.920
cec17_5	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_6	5.4e7 ± 1.2e3	3.6e6 ± 1.6e4	329.9 ± 22.70	1.5e3 ± 188.1	51.40 ± 13.41	423.3 ± 29.72	325.7 ± 21.03	103.6 ± 21.03
cec17_7	136.7 ± 20.54	80.10 ± 5.330	77.92 ± 1.233	81.21 ± 3.212	101.1 ± 21.11	93.34 ± 12.12	82.23 ± 22.22	79.99 ± 1.211
cec17_8	114.5 ± 0.950	145.3 ± 1.150	0.000 ± 0.000	8.840 ± 9.650	5.130 ± 1.160	27.30 ± 2.660	25.17 ± 2.590	0.200 ± 2.590
cec17_9	1.3e4 ± 823.3	1.3e4 ± 364.3	1.1e4 ± 259.2	1.6e4 ± 348.2	1.1e4 ± 3.4e3	1.1e4 ± 1.8e3	9.2e4 ± 236.3	7.6e4 ± 236.3
cec17_10	3.3e3 ± 606.5	9.5e4 ± 360.0	91.73 ± 9.210	7.1e6 ± 1.2e3	47.33 ± 25.08	17.62 ± 2.780	44.52 ± 3.960	15.74 ± 3.960
cec17_11	1.5e7 ± 2.5e6	1.1e8 ± 1.1e3	0.000 ± 0.000	2.6e3 ± 400.6	3.580 ± 2.740	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
cec17_12	8.5e7 ± 5.5e6	1.1e5 ± 3.1e4	92.48 ± 2.160	320.1 ± 20.23	124.4 ± 58.86	124.4 ± 8.920	95.58 ± 7.080	101.6 ± 7.080
cec17_13	1.3e4 ± 2.3e3	4.3e8 ± 5.8e7	0.000 ± 0.000	0.000 ± 0.000	279.4 ± 196.4	34.77 ± 8.660	34.00 ± 0.730	0.000 ± 0.000
cec17_14	2.6e6 ± 2.8e5	3.2e9 ± 5.5e6	4.552 ± 0.122	181.3 ± 21.11	132.2 ± 167.2	21.46 ± 2.780	30.74 ± 2.890	10.67 ± 2.890
cec17_15	1.3e3 ± 201.4	1.4e4 ± 760.1	8.348 ± 1.222	12.12 ± 1.223	0.730 ± 0.190	11.47 ± 0.920	9.070 ± 0.320	6.550 ± 0.320
cec17_16	9.2e8 ± 8.4e7	9.9e5 ± 0.000	15.67 ± 4.455	83.33 ± 3.222	134.7 ± 80.60	21.95 ± 6.650	25.44 ± 2.340	1.530 ± 2.340
cec17_17	4.1e4 ± 1.5e4	5.9e5 ± 7.4e3	12.34 ± 5.334	17.56 ± 2.112	103.2 ± 65.50	15.71 ± 5.900	7.180 ± 0.260	0.510 ± 0.260
cec17_18	9.1e8 ± 1.8e8	9.9e6 ± 0.000	20.36 ± 0.944	123.1 ± 34.33	34.30 ± 51.24	87.02 ± 3.940	101.7 ± 0.430	81.03 ± 0.430
cec17_19	76.48 ± 6.470	4.2e3 ± 943.5	13.87 ± 2.153	54.45 ± 0.122	16.23 ± 4.450	12.35 ± 3.230	11.52 ± 0.460	1.270 ± 0.460

Table 7.3: Mean and standard deviation of FEV after 25 executions on all benchmark functions with 50 variables ($d = 50$) obtained from the EMNA, EGNA, SPEDA, m_KEDA, CMA-ES, JADE, SHADE and L-SHADE algorithms. The best result for each benchmark is highlighted in blue.

Based on the results of the experiments for $d = 30$ and $d = 50$, we conclude that SPEDA can be a competitive tool for continuous optimization compared to some state-of-the-art population-based approaches. Indeed, it is able to converge to solutions with low variance in independent algorithm executions. Furthermore, the optimal landscapes for our approach

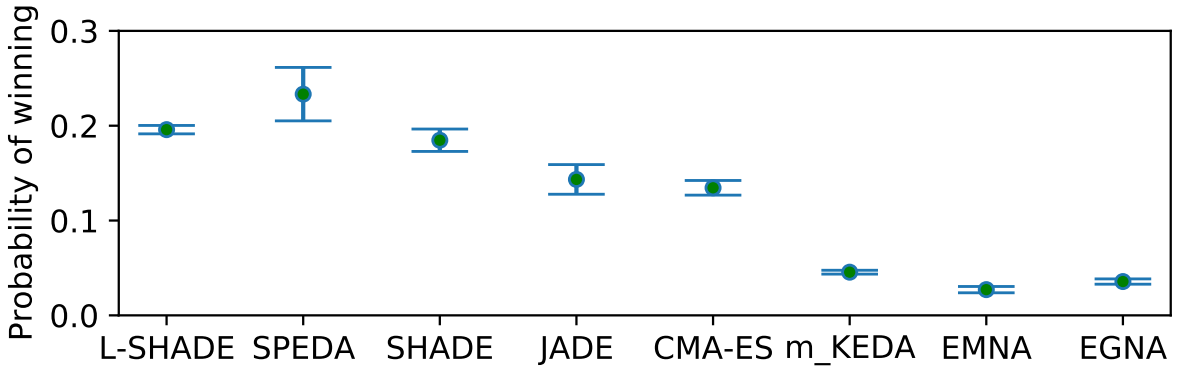


Figure 7.8: Credible intervals (5% and 95% quantiles) and expected probability of winning (green dots) for results shown in Table 7.3.

seem to be functions that are unimodal separable, although it still outperforms its competitors in most of the nonseparable and multimodal functions. A good performance has also been identified in the composition functions and landscapes with a large number of local optima, regardless of the dimension.

7.3.3 Portfolio optimization

In this section, we compare the previously tested algorithms in a real-world portfolio optimization problem.

The portfolio optimization problem is based on the diversification aspect of the investment, where investors diversify their investments into different types of assets. The objective of this optimization task is to maximize the return of the investment but also to minimize its risk. This bi-objective problem is reduced to a single-objective task by using the Sharpe-ratio metric [Sharpe, 1994], combining both aspects as follows,

$$Sharpe\_ratio = \frac{R_p - R_f}{\sigma_p}, \tag{7.1}$$

where R_p, R_f and σ_p are the return of the portfolio, the risk of the investment and the standard deviation of the portfolio's excess return for a series of time intervals, respectively.

The cost functions computations were made using PyPortfolioOpt Python package [Martin, 2021] including historical daily stock prices of 20 different assets ($d = 20$) from 12/29/1989 to 04/11/2018, also available in PyPortfolioOpt[6].

Figure 7.9 shows the Sharpe-ratio boxplot of the best solutions achieved for different algorithms. The maximum number of iterations has been limited to 300. It is observed that SPEDA, CMA-ES and JADE are the approaches that achieve the best solutions in terms of Sharpe-ratio maximization compared to its competitors. Good results are also found by m_KEDA. In this case, the results found by EMNA improve those found by EGNA, which have a large dispersion between the solutions. Note that the median of the EGNA solutions gives us a hint about the

[6] https://raw.githubusercontent.com/robertmartin8/PyPortfolioOpt/master/tests/resources/stock_prices.csv

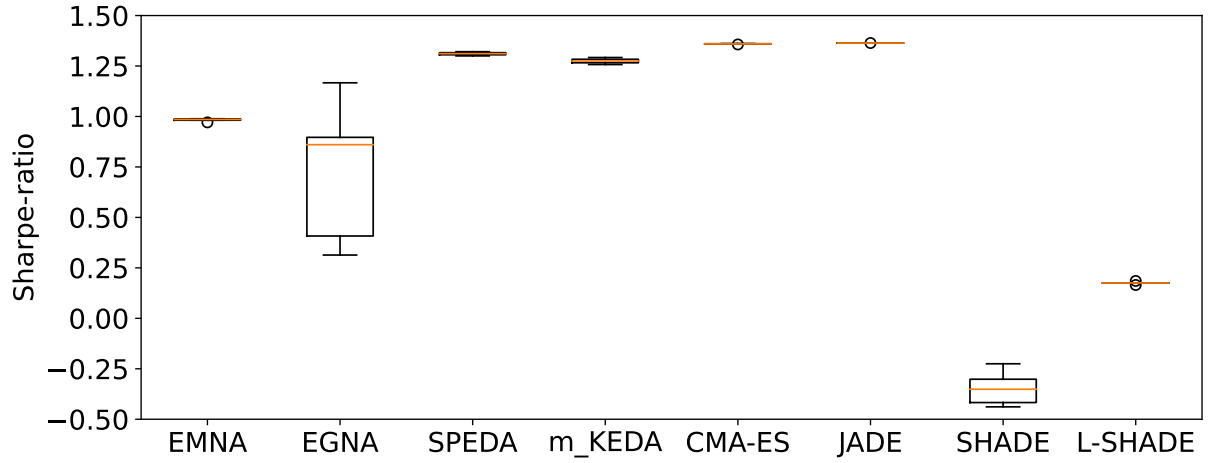


Figure 7.9: Boxplot with the best results found by each algorithm after 25 independent executions for the specific portfolio optimization problem.

presence of extreme data in the lower bound of the boxplot. CMA-ES and JADE approaches seem to converge to a unique solution in all the executions. While the SHADE and L-SHADE approaches offer good solutions for the benchmarks studied, they do perform well on this real problem. However, our approach maintains good performance on both types of optimisation problems. The results shown in Figure 7.9 have been analyzed using statistical tests to reject the null hypothesis of equal means between the different methods, and obtained a p-value of 1.75e-16. Thus, a statistically significant difference was found between the performance of the optimizers. However, analyzing the statistical tests by pairs, there is no statistically significant difference between the results obtained by EMNA and EGNA.

7.3.4 CPU time and complexity analysis

In this section, we analyze the average CPU time spent during the execution of the algorithms, and the asymptotic time complexity of SPEDA.

Figure 7.10 shows a CPU time comparison between all the approaches used for the benchmarks. It is observed that m_KEDA is the most expensive and CMA-ES the fastest. Analyzing the four EDA, the higher the complexity of the algorithm is, the longer the execution time. Thus, m_KEDA has a longer execution time than SPEDA, which in turn takes longer than EGNA, which takes longer than EMNA. The three differential evolution variants have a similar average CPU time to that found for EMNA, where JADE is the most expensive.

The increase in SPEDA complexity compared to that of EGNA is caused by the cross-validated cost function that evaluates CKDE and Gaussian nodes. Considering this, the complexity of SPEDA is simplified as $\mathcal{O}(tOi\lambda KT)$, where t is the number of iterations of SPEDA. Figure 7.10 shows that, in general, m_KEDA is more complex than SPEDA due to the cross-validated function over all the CKDE nodes, which is d in each iteration, in contrast to SPEDA, which is d at most. The cross-validated function used by SPEDA, increases the CPU time but also yields the best results, as shown in Table 7.2 and Table 7.3. Note that tuning the hyper-parameter K (number of folds) might reduce the computation time, but also may lead

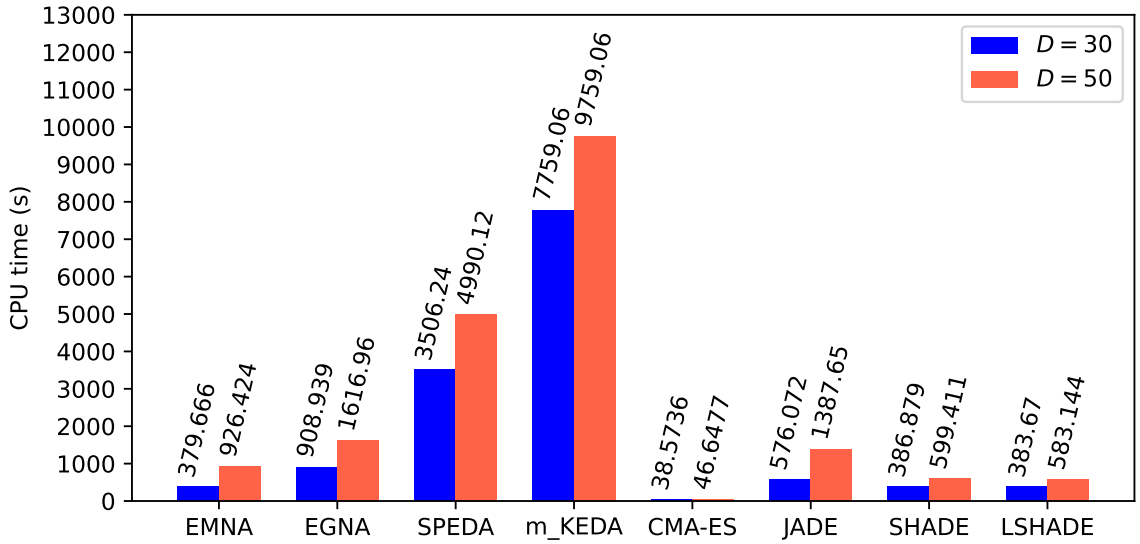


Figure 7.10: Comparison of the average CPU time (in seconds) after 25 independent executions on each of the benchmarks in CEC2014 and CEC2017, for EMNA, EGNA, m_KEDA, SPEDA, CMA-ES, JADE, SHADE and LSHADE. The results for 30 and 50 dimensions are shown in blue and red, respectively.

to poorest solutions.

7.4 Conclusions

Traditional EDAs typically use Gaussian distributions to optimize continuous functions, such as EGNA or EMNA, which use GBNs and multivariate Gaussian probability distributions, respectively. Nevertheless, using these types of probabilistic models implies assuming Gaussian distributions that only consider linear relationships between variables. In this chapter, we have propose semiparametric estimation of distribution algorithms (SPEDAs), which learn a semiparametric Bayesian network at each iteration, with the coexistence of nodes that are fitted by CKDE and nodes that assume Gaussianity. SPEDA decides iteratively whether Gaussians or CKDEs are fit at each node.

Moreover, the traditional EGNA usually learns a GBN in each iteration by considering only the best solutions of the last iteration, which may lead to a high variance between the solutions in independent runs of the algorithm, or convergence to local optimal solutions. SPEDA has been designed to overcome this limitation by learning a probabilistic model in each iteration considering the best solutions of previous iterations. This is intended to establish a search direction in the landscape based on the learned information.

The empirical results showed a comparison of SPEDA with some of the most widely used EDAs for continuous function optimization, like EMNA and EGNA. We conclude that using a more complex probabilistic model, such as an SPBN, improves the results compared to those of EGNA and EMNA. SPEDA was also compared to CMA-ES; to JADE, SHADE, and L-SHADE as members of differential evolution algorithms family; and to the multivariate

KDE EDA, the extreme case of SPEDA where the Gaussian nodes are forbidden. The experiments were run on 49 benchmarks in 30- and 50-dimensional spaces, and it was found that SPEDA provided the best results in most of the benchmarks that considered landscapes of different characteristics. The results found were analyzed using a Bayesian performance analysis with the Plackett-Luce model to estimate the probability of each algorithm to be the winner approach. L-SHADE was the most accurate algorithm for the case of 30 dimensions followed by SPEDA with similar performances. A similar behaviour was observed for the case of 50 dimensions where SPEDA is the most probable approach. The experimental results also include a real world optimization problem, where statistical significant differences were found, being SPEDA one of the best performing approaches, together with CMA-ES and JADE.

The following research lines are proposed as future work:

- SPEDA is a tool that can be of great benefit for optimization tasks in complex continuous environments where the variance between the solutions proposed by our approach in different executions is low, without sacrificing the quality of the solutions. Future work include exploring real applications in the industry and quantum machine learning tasks.
- The experiments showed that learning SPBNs in each iteration can be slower than learning GBNs, so future research would include shortening this learning time.
- The structure learning of SPBN models have been shown to be complex. Future work include reducing the number of learnings during runtime.

Chapter 8

Quantum-inspired EDA to Solve the Traveling Salesman Problem

8.1 Introduction

Chapters 5-6 presented different methodologies and applications of EDAs. In this chapter we focus on adding a QC perspective to EDAs, corresponding to *CQ* in Figure 1.1. Specifically, we present a quantum-inspired approach in which new solutions are sampled from a quantum system to solve the traveling salesman problem (TSP).

TSP is a combinatorial optimization problem widely studied in the literature in different research areas. Regarding QC, several techniques such as adiabatic QC [Kieu, 2019], and QAOA [Ruan et al., 2020] are used to solve it. Regarding classical computation EAs [Robles et al., 2002, Larrañaga et al., 1999], deep learning [Miki et al., 2018] and reinforcement learning [Zhang et al., 2020] have been applied to approach the problem.

The TSP is a well known NP-hard problem [Applegate et al., 2006]. The problem corresponds to finding the shortest Hamiltonian cycle in a graph of n cities represented as vertices in the graph and edges between the nodes to represent the interconnections. Each edge between vertices i and j is associated with a given cost denoted by d_{ij}. Thus, the TSP consists of finding a permutation π of n cities that minimizes the function of the total cost of visiting the n cities and going back to the starting point:

$$C(\pi) = \sum_{i=1}^{n-1} \left(d_{\pi(i),\pi(i+1)} \right) + d_{\pi(n),\pi(1)} \tag{8.1}$$

Previous studies on quantum machine learning [Schuld, 2018, Lau et al., 2017, Wiebe et al., 2012] have focused on exploiting the benefits of the QC to improve the performance of the optimization algorithms. With respect to EAs, quantum machine learning revolves around three main research areas [Zhang, 2011]: (i) quantum-inspired evolutionary algorithms (QIEA), which take advantage of the concepts and principles of quantum mechanics to improve the classic EAs; (ii) evolutionary-designed quantum algorithms, which develop new quantum

algorithms implemented by genetic programming; and (iii) quantum evolutionary algorithms, which develop new evolutionary algorithms to be executed in quantum devices.

In this chapter, we will focus on QIEA. Since the seminal paper [Han and Kim, 2002] where principles of QC were used in the reproduction step of the EA, other works have applied different modifications to improve the results, or accommodate the algorithm to specific optimization problems. A matrix quantum individual representation is used in Silveira et al. [2012], da Silveira et al. [2017] to solve ordering optimization problems. A novel quantum-inspired algorithm is proposed by Montiel et al. [2019] motivated by the colony behavior of the leafcutter ants. Other works propose different modifications to these algorithms in order to improve the smoothing and efficiency of the exploration [Platel et al., 2008], and compare them to the EDAs behaviour [Platel et al., 2007]. However, these works cannot be executed in a real circuit model-based quantum computer without being adapted, as mentioned in Ross [2019]. Implementing a QIEA in a quantum computer requires a hybrid implementation between quantum programming, for the reproduction step, and the classical programming, for the rest of the EA steps. Running these algorithms in a quantum computer also includes considering the quantum noise present in the quantum computers.

Here, we present a new quantum-inspired approach for solving the TSP. The Quantum-Inspired Estimation of Distribution Algorithm (QIEDA) is a population-based algorithm based on QC for the reproduction of new solutions during runtime. The solutions obtained by the QIEDA are competitive with other state-of-the-art population-based approaches in terms of convergence and quality of solutions found. Also, we measure the impact of running the QIEDA in different quantum computers with non-identical topologies, and analyze which would be the ideal quantum topology to solve the TSP. The novelty of this algorithm is that it can be executed without adaptations in a real quantum computer based on the circuit model programming. A parameterized quantum circuit is used during the reproduction step in order to sample new solutions. The experiments were executed simulating the real IBM quantum computers.

This chapter contains the developed methodology and results included in Soloviev et al. [2021]. The implemented methodology is available in a GitHub repository[1].

Chapter outline

The chapter outline is organized as follows. Section 8.2 explains the implementation of the QIEDA approach. Section 8.3 presents an empirical comparison of the QIEDA with other optimization algorithms and a benchmarking of the computational cost for the TSP. Finally, Section 8.4 concludes the chapter and proposes future research lines.

8.2 QIEDA

QIEDA is a multivariate EDA which uses probabilistic logic sampling (PLS) [Henrion, 1988] to generate new solutions from a BN. The algorithm samples new solutions taking advantage

[1] https://github.com/VicentePerezSoloviev/QIEDA

from the QC principles. The QIEDA implementation is a hybrid implementation between classic and quantum computations. Qiskit (0.16.0) and IBM platforms [Qiskit contributors, 2023] have been used to implement the quantum part.

8.2.1 Representation

In each iteration of the EA, QIEDA performs a sampling process to obtain a set of solutions.

There are a lot of ways to encode the TSP solutions [Larrañaga et al., 1999]. In this chapter, a matrix representation is used. Assume a set of cities $U = \{1, 2, \ldots, n\}$ of the TSP. The vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is a solution for the problem where $x_i \in U$, and i denotes the city ordering. Every x_i assumes one of the elements of U with a certain probability. The sum of probabilities of the elements in U must be equal to one. For example, for a TSP problem of size $n = 5$, assume for the first city to be visited (x_1) a vector of probabilities for the elements of U being observed $(0.1, 0.2, 0.1, 0.5, 0.1)$. Thus, the element x_1 takes the fourth city in U with 0.5 probability as it is the city with the highest probability, and once it is fixed, this city cannot be selected again by the rest of the elements of $\boldsymbol{x}$.

When adapting this codification to quantum individuals, each of the elements of U is translated as a pure state of an n-qubit. Then, every x_i assumes a different pure state with a certain probability. Each of the pure states that form the n-qubit have only one qubit in the $|1\rangle$ state. The position of the $|1\rangle$ state will be the city that represents that specific state. For example, in an n-qubit of size $n = 5$, the only desired pure states are $10000, 01000, 00100, 00010, 00001$ which represent the cities $1, 2, 3, 4, 5$ respectively. Thus, a universe of size n is represented by an n-qubit with n different states.

The n-qubit that represents the desired states to our problem codification is,

$$|\psi\rangle = \delta_1 |10\ldots00\rangle + \delta_2 |01\ldots00\rangle + \cdots + \delta_{n-1} |00\ldots10\rangle + \delta_n |00\ldots01\rangle , \tag{8.2}$$

where $|\delta_i|^2$ are the probabilities of each of the desired states and $\sum_{i=1}^{n} |\delta_i|^2 = 1$. Note that the rest of $2^n - n$ states, which are not desired to be observed, are set to amplitudes equal to zero.

Therefore, using the matrix quantum codification, each solution of the TSP is represented as,

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \ldots & x_{nn} \end{bmatrix}, \; x_{ij} \in \{0, 1\},$$

where rows accounts for city ordering while columns correspond to the position in U, and x_{ij} is equal to 1 whenever a city j is visited in the ordering i position. Thus,

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in 1, \ldots, n. \tag{8.3}$$

An example of a valid solution for a TSP of size $n = 5$ is,

$$\boldsymbol{x_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

which corresponds to the following ordered city path [5, 3, 4, 1, 2].

8.2.2 Algorithm

Algorithm 8 describes the QIEDA algorithm.

Algorithm 8 QIEDA pseudocode

1: $p(\boldsymbol{X}) \leftarrow$ Initialize matrix of statistics
2: $G_0 \leftarrow$ Generate N individuals from $p(\boldsymbol{X})$
3: **for** $t = 1, 2, ...$ until a stopping criterion is met **do**
4: $\quad G_{t-1} \leftarrow$ Evaluate the individuals with cost function C
5: $\quad G^S_{t-1} \leftarrow$ Select $S < N$ individuals from G_{t-1}
6: $\quad p(\boldsymbol{X}) \leftarrow$ Update the matrix of statistics from G^S_{t-1}
7: $\quad G_t \leftarrow$ Generate new generation from $p(\boldsymbol{X})$
8: **end for**

The QIEDA is initialized with a matrix of statistics (step 1) which specifies the relative frequency of appearance of each city in each ordering position, among the best solutions selected from the previous generation. In each iteration, the algorithm generates some new individuals from the matrix of statistics (step 2). Then the individuals are evaluated according to the cost function in Equation 8.1 we desire to optimize (step 4), and the best individuals of the generation are selected (step 5) in order to improve the next generation cost. The matrix of statistics is updated (step 6) in each iteration with the selected individuals.

8.2.2.1 Initialization

The algorithm must be initialized with a matrix of statistics which specifies the probability of each position to take the value 1 when sampled,

$$p(\boldsymbol{X}) = \begin{bmatrix} p(\boldsymbol{X}_1) \\ \vdots \\ p(\boldsymbol{X}_n) \end{bmatrix} = \begin{bmatrix} \delta_{11} & \dots & \delta_{1n} \\ \vdots & \ddots & \vdots \\ \delta_{n1} & \dots & \delta_{nn} \end{bmatrix} \tag{8.4}$$

The probabilities of each row verify the restriction of Equation 8.3, and correspond to the probabilities of each of the pure states described in Equation 8.2. This matrix is initialized so that no solution is favored beforehand and it is updated during the algorithm runtime until

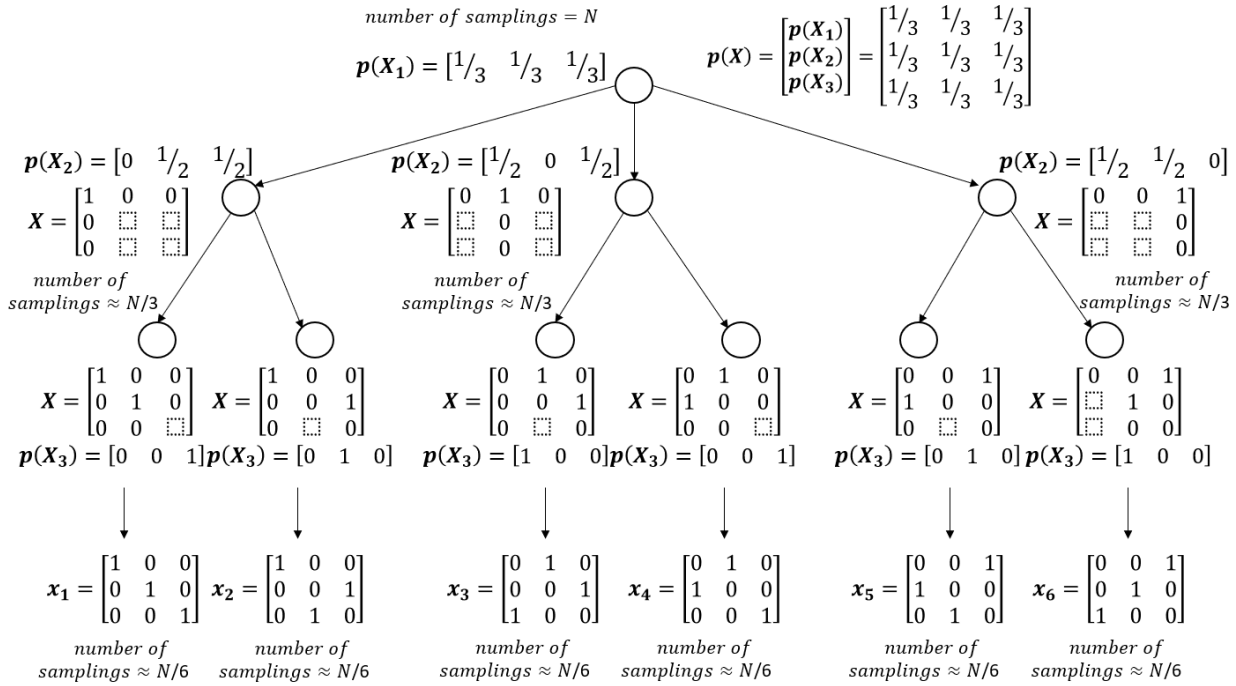


Figure 8.1: Example of the quantum individuals sampling of size $n = 3$. An initial matrix of statistics $p(\boldsymbol{X})$ is updated in each node depending on its parents samplings. In the leaf nodes, the final individuals are obtained. The algorithm samples N individuals. Each node distributes its number of samples among its child nodes.

convergence is reached,

$$p(\boldsymbol{X}) = \begin{bmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}$$

When the QIEDA converges, the matrix of statistics should be a matrix of 1s and 0s (Equation 8.3), which is the optimum solution found by the algorithm for the proposed TSP.

8.2.2.2 Individuals Generation

As described in Section 8.2.1, the individuals are coded as matrix of ones and zeros where each row i specifies the city visited in the i position of the city ordering. When a city is visited, the following rows must update the probabilities to do not allow visiting this city again. The sampling process can be seen as a sequence of dependent steps in which each step means to sample a row i, and update rows $i + 1, i + 2, \cdots, n$ according to the i sampling outcome.

Consider a row i of an individual to be sampled in the QIEDA sampling method, expressed as $[x_{i1}, \ldots, x_{in}]$. The row sampling consists of a random selection among the pure states defined by Equation 8.2 with probabilities to be selected $p(\boldsymbol{x}_i) = [\delta_{i1}, \ldots, \delta_{in}]$ defined in the row i of the matrix of statistics (Equation 8.4) of the corresponding algorithm iteration. Assume that the pure state c has been selected. Then, the following rows of the matrix of statistics are

Algorithm 9 PLS pseudocode

Input Matrix of statistics $p(\boldsymbol{X})$, population size N
Output Set of individuals

struct {
  stats Statistics to sample W state circuit
  size Number of shots from the quantum circuit
  level Level in the tree structure
  wstate W state circuit with *stats* probabilities
} *Node*;

1: $root \leftarrow Node(stats[0], size = N, level = 0)$
2: $leaf \leftarrow [root]$
3: **for** *node* in $leaf$ **do**
4: $sols \leftarrow$ Sample $node.wstate$ $node.size$ times
5: **for** *sol* in $unique(sols)$ **do**
6: $level_{sol} \leftarrow node.level + 1$
7: $size_{sol} \leftarrow$ Count samplings of *sol* in *sols*
8: $node_{sol} \leftarrow Node(stats[level_{sol}], level_{sol}, size_{sol})$
9: $node_{sol}.stats \leftarrow$ Update statistics for row $level_{sol}$ given $node_{sol}$
10: $leaf.add(node_{sol}) \leftarrow$ Add node to leaf nodes list
11: **end for**
12: $leaf.remove(node) \leftarrow$ Remove node from leaf nodes list
13: **end for**

updated with zero probability for state c ($\delta'_{hc} = 0$, for $h > i$), as the city cannot be visited more than once. Also, the remaining probabilities are updated as,

$$\delta'_{hj} = \frac{\delta_{hj}}{1 - \delta_{ic}} \tag{8.5}$$

for $h > i, j \neq c$, where δ_{ic} is the previous probability value of row i and column $j = c$, that is replaced by zero. For example, assume a row which takes the values $[0.2, 0.3, 0.4, 0.1]$, and the last probability must be replaced by zero. Then, the row would result in $(\frac{0.2}{1-0.1}, \frac{0.3}{1-0.1}, \frac{0.4}{1-0.1}, 0)$.

The generation reproduction is arranged as a tree. An example of the tree building is shown in Figure 8.1. Starting from the root node, we sample the first individual row from $p(\boldsymbol{x}_1)$. As a generation consists of N individuals, the algorithm performs N samplings that will be distributed among the possible pure states according to $p(\boldsymbol{x}_1)$. For example, in Figure 8.1, N is distributed by approximately 33% for each node. This is the first level of the tree, whose maximum width is the number of pure states defined by $p(\boldsymbol{x}_1)$. It might occur that N is distributed in such a way that a node is not sampled, and thus, the first level width would be lower than expected. After updating the following rows, probabilities of rows 2 to n, the next level of the tree is built. Each node of level 1 unfolds its possible solutions to level 2 depending on the updated $p(\boldsymbol{x}_2)$. If a node is not sampled, it is not unfolded in the next level. Each node has an independent matrix of statistics depending on the parent node. The samplings

Algorithm 10 W state modified building

Input Vector of statistics
Output W state quantum circuit

1: $stats \leftarrow [\delta_1, \ldots, \delta_n]$
2: $rot_1 = 2cos^{-1}(\sqrt{\delta_1})$
3: $wstate \leftarrow$ Quantum circuit with n qubits $(x_1, \ldots, x_n)$
4: $wstate \leftarrow$ RY in x_1 of rot_1 rad.
5: **for** $i = 2 \ldots n$ **do**
6: $\quad amp = \sqrt{1 - \sum_{j=1}^{i} \delta_i}$
7: $\quad rot_i = 2cos^{-1}(\frac{(\delta_i)^{1/2}}{amp})$
8: $\quad wstate \leftarrow CRY$ (rot_i rad.) target x_i and control x_{i-1}
9: **end for**
10: **for** $i = n, \ldots, 1$ **do**
11: $\quad wstate \leftarrow CX$ with target i and control $i - 1$
12: **end for**
13: $wstate \leftarrow X$ gate x_1

of each node for the same level are independent, and thus, can be sampled in parallel. This process continues until reaching the leaf nodes (level n), where the sampled individuals are obtained. Total tree width varies depending on how N is distributed along the tree nodes. The maximum tree width is $N!$.

If we consider the tree as a directed acyclic graph in which each level nodes depend on the previous level nodes, the tree is a BN. There are a great amount of methods for sampling BNs [Koller and Friedman, 2009]. In this paper, the process followed is similar to the probabilistic logic sampling (PLS), in which each node depends on its parents. PLS defines an ancestral ordering of nodes, which in this case is the city (row) ordering.

Algorithm 9 shows the adapted PLS for reproduction. We define a node as a structure with: (i) updated statistics depending on its parents, (ii) number of samplings (quantum circuit shots) to distribute among child nodes, (iii) level of the node, and (iv) the W state quantum circuit used for sampling.

To sample the individuals rows, the W state quantum circuits have been implemented [Cruz et al., 2019]. The W state is an entangled quantum state [Nielsen and Chuang, 2002] of n qubits, in which all possible pure states have one of the qubits in the $|1\rangle$ state, while all other ones are in the $|0\rangle$ state,

$$|W_n\rangle = \frac{1}{\sqrt{n}}(|10\ldots00\rangle + |01\ldots00\rangle + ... + |00\ldots10\rangle + |00\ldots01\rangle) \tag{8.6}$$

The general W state of Equation 8.6 has been modified in order to be able to set different probabilities to the different qubits. Thus, we can apply this approach to sample solutions according to Equation 8.2. The circuit building is described in Algorithm 10. The process is divided into two main parts: (i) probability redistribution, in which, the desired probabilities

of the pure states that constitute the W state are translated to rotations in the qubit Y axis with R_Y gates and controlled-RY gates (CRY), and (ii) state reshuffling, to ensure that the number of qubits in the $|1\rangle$ state is only one, with $CNOT$ and X gates (see Section 4.2.3 for quantum gates details).

An example of a circuit of size $n = 5$ in which all pure states have the same probabilities is shown in Figure 8.2.

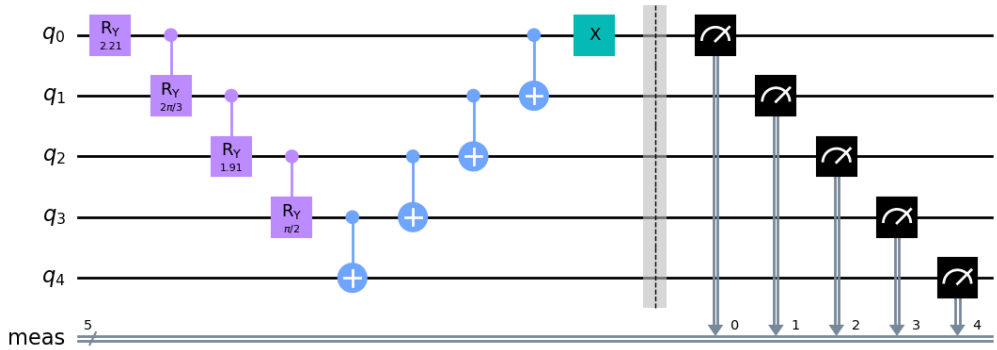


Figure 8.2: Example of a W state circuit for $n = 5$. The probabilities are set to $[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}]$. Purple gates are RY and CRY gates, blue ones are CX gates, green one is X gate, and black ones are the measurement gates.

The influence of noise in the W state circuit samplings is shown in Figure 8.3. Panel (a) shows the histogram of the results obtained executing the quantum circuit in a quantum simulator without noise, while panel (b) shows the results obtained executing the circuit in a real quantum computer. Both experiments were run 1000 times with probabilities $[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}]$. Note that in Figure 8.3(b) some not desired solutions are sampled with a small probability due to the quantum computer noise. Solutions must be filtered and the probability distribution among the valid solutions normalized. Valid solutions are the pure states defined following Equation 8.2.

8.3 Results

In Section 8.3.1 the QIEDA performance is compared with other optimization algorithms, and in Section 8.3.2 the QIEDA is executed for different quantum computing topologies to analyze the computational cost. The ideal simulator and some real devices simulators such as Johannesburg or Tokyo were used to run the experiments.

8.3.1 Algorithm Performance

The performance of the QIEDA has been compared to other well known state-of-the-art population-based algorithms: GA, a binary adaptation of the particle swarm optimization (PSO) [Hadia et al., 2012], ant colony optimization (ACO) [Yu, 2014], and a non-quantum estimation of distribution algorithm (EDA). The non-quantum EDA is a modification of the QIEDA approach in which, the solutions are not sampled from a quantum circuit. New

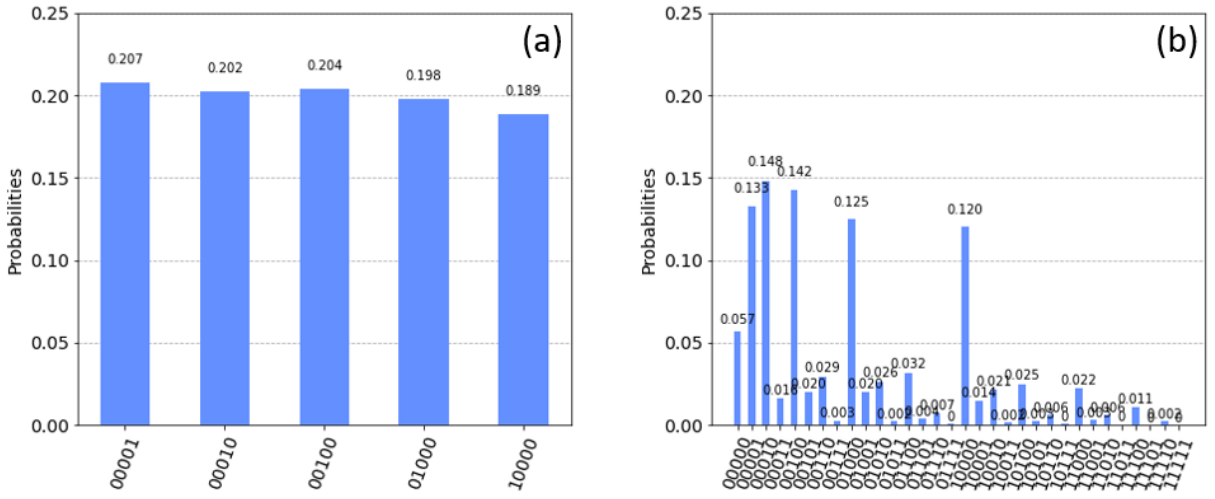


Figure 8.3: Probability distribution of the W state circuit sampling for $n = 5$ without (a) and with (b) noise. The quantum circuit was run 1000 times for each scenario.

solutions are sampled from a categorical distribution using a binary random number generator. Its pseudocode is similar to that described in the quantum-inspired evolutionary algorithms [Zhang, 2011], in which the number of qubits is not a limitation.

The hyper-parameters of the population-based algorithms are shown in Table 8.1. Despite the fact their tunning is out of the scope of this chapter, a previous analysis was done to achieve good solutions for the experiments, and compare them fairly with the QIEDA. For ACO, α is the relative importance of the pheromone and β is the relative importance of the trigger factor. As observed in [Yu, 2014], $\alpha < \beta$ achieves better results. rho is the evaporation rate of global pheromone which is equal to the evaporation rate of the local pheromone. For PSO, α is the cognitive parameter to control the exploitation component and β is the social parameter to control the exploration component of the algorithm. For GA, α is the population percentage considered as elite selection, and the mutation rate m_r is the parameter which influences how the individuals are modified in the mutation phase of the algorithm. For EDA, α is the population percentage considered as elite selection. N and Gen are the population size and the number of iterations, respectively.

	N	Gen	α	β	m_r	rho
ACO	50	40	0.4	0.6	-	0.5
PSO	50	40	0.55	0.45	-	-
GA	50	40	0.5	-	0.04	-
EDA	50	40	0.5	-	-	-

Table 8.1: Algorithms configuration.

The QIEDA is executed with the ideal simulator without quantum noise and in a quantum simulator with the Johannesburg quantum computer noise. The latter is the most realistic simulation as it is executed simulating the behaviour of a real quantum device. Johannesburg

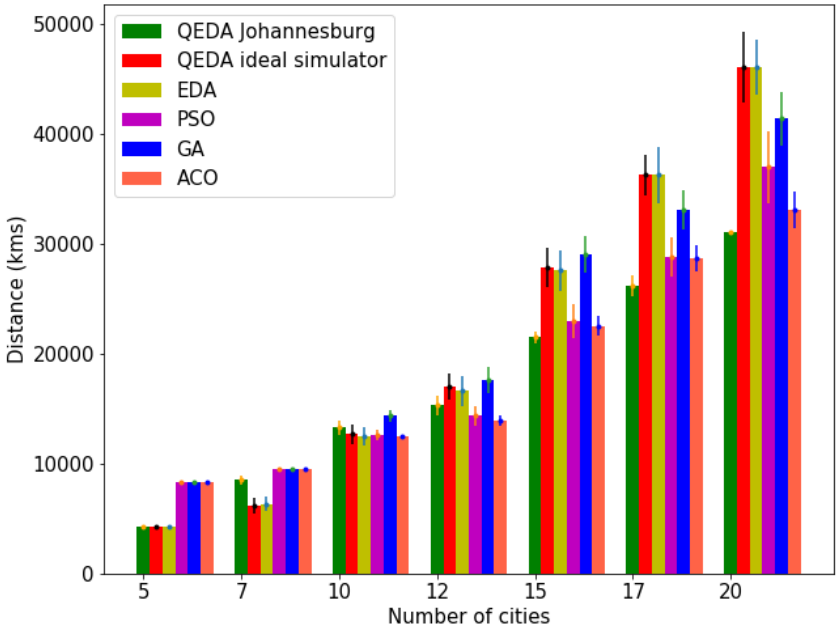


Figure 8.4: Mean best cost for the TSP for different number of cities and different algorithms: QIEDA executed in the Johannesburg quantum simulator with noise, QIEDA executed in a quantum simulator without quantum noise, the non-quantum estimation of distribution algorithm (EDA), particle swarm optimization (PSO), GA, and the ant colony optimization (ACO).

quantum device was chosen as the case study since it was observed that the quantum device selection does not influence the results obtained by the QIEDA.

Different datasets[2] have been used to simulate different TSP sizes. The experiments have been carried out for sizes $n = [5, 7, 10, 12, 15, 17, 20]$. The Qiskit simulators only allow to simulate up to 25 qubits. Thus, the aim of this analysis is to study the trend of the results in order to determine the performance of the algorithm.

Figure 8.4 shows an analysis of the optimum solutions obtained by the QIEDA compared to other population-based algorithms. The algorithms were run 100 times and the figure plots the mean and the deviations of the executions. Note that the QIEDA executed in the Johannesburg quantum computer achieves competitive results compared to other optimization algorithms. In nearly all the experiments, QIEDA achieves the best results on average compared to others, with a small deviation. In order to verify if the results are statistically significant some tests were run. After checking that the data fit a normal distribution, we calculated the p-value with an analysis of variance (ANOVA) [Kaufmann and Schering, 2007], and the Student's t-test [Kalpić et al., 2011] of the QIEDA results compared to the other algorithms for each of the experiments. A significance level of 0.05 was set. ANOVA p-values

[2] http://www.math.uwaterloo.ca/tsp/world/countries.html

are shown in Table 8.2, and all t-tests yielded p-values lower than 0.05, so we can reject the null hypothesis of equal means. The QIEDA executed in the ideal simulator does not reproduce any noise, so the performance of the algorithm is similar to the non-quantum EDA. Note that the results obtained by both algorithms are very similar.

n=5	n=7	n=10	n=12	n=15	n=17	n=20
6.1e-56	1.9e-23	0.021	8.8e-09	3.7e-18	1.4e-21	2e-23

Table 8.2: ANOVA test p-values

It is also noteworthy that when increasing the TSP size, the QIEDA performance improves compared to the other algorithms. The trend shows that, for larger (and therefore more complex) TSPs, the QIEDA will find much better solutions than the other population-based optimizers.

Figure 8.5 shows an analysis of the convergence of the QIEDA. The figure shows the mean and the deviation of 100 executions of the algorithm. The algorithm is compared to the same population-based algorithms in the same conditions as before. The EDA approaches have better convergence compared to other algorithms. Note that, regardless of the problem size, the QIEDA convergence remains approximately constant and with a small value compared to the other algorithms.

Despite the fact that the QIEDA approach was not able to be executed for larger TSP sizes, we can observe the trend of the results. As far as this behavior is concerned, the QIEDA approach achieves competitive results compared to the other algorithms. Moreover, the convergence of the algorithm remains constant independently of the size of the problem. Our results suggest that intrinsic quantum noise has the ability to enhance the convergence and the cost of the TSP as noise modifies the search space in such a way that it prevents the algorithm from falling into local optimal solutions.

8.3.2 Analysis of computing topologies

There exists a large variety of quantum computers among which the main differences are the quantum topology (Figure 8.6) which distributes the qubits and their relations. Two qubits A and B can only operate with each other only if there is a connection between both qubits. Otherwise, the system needs to swap with other qubits until A and B are connected.

To run a quantum circuit in a specific quantum computer some aspects must be analyzed:

- Available quantum gates. The circuit might have some quantum gates not available in the desired computer, so it must be adapted to execute it.
- Available number of qubits.
- Topology. The qubits of each quantum computer are distributed following different topologies. The circuit must adapt so the number of $SWAP$ operations is minimized.

Executing a circuit without optimizing the number of $SWAP$ operations increases the computational cost. Minimizing the number of quantum gates in the circuit decreases the

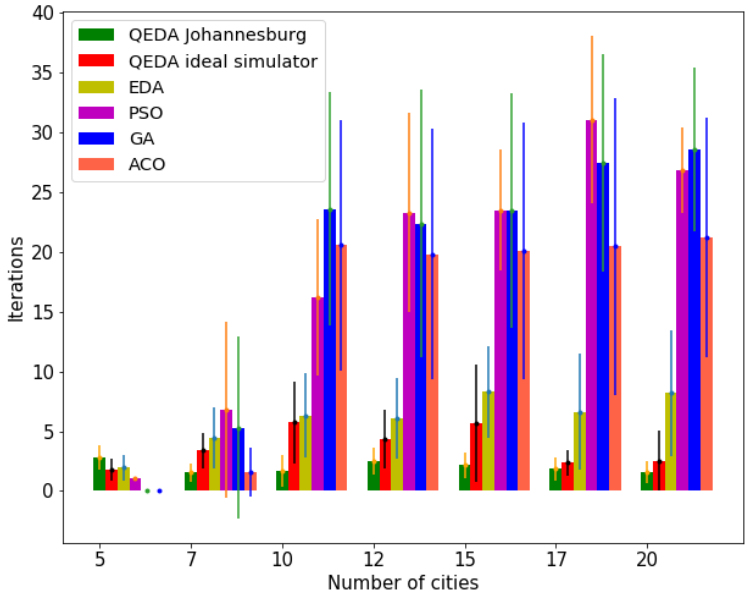


Figure 8.5: Mean convergence for the TSP for different number of cities and different algorithms: QIEDA executed in the Johannesburg quantum simulator with noise, QIEDA executed in a quantum simulator without quantum noise, the non-quantum estimation of distribution algorithm (EDA), particle swarm optimization (PSO), GA, and the ant colony optimization (ACO).

computational cost. The aim of this benchmark is to find the ideal topology to execute the designed approach. Finding the topology which minimizes the computational cost of each executed W state circuit, would mean to reduce the computational cost of the QIEDA approach. To that end, we study the circuit depth that refers to the number of time steps (time complexity) required for the quantum operations making up the circuit to run on the quantum hardware [Gyongyosi, 2020].

The analysis and adaptation of the circuit to the specific quantum computer is named transpilation. During the transpilation, the circuit can be optimized to minimize the $SWAP$ operation. Table 8.3 shows an analysis of the circuit depth with and without optimization for different topologies (Figure 8.6). All the selected quantum computers have the same available quantum gates, so the differences between the adapted circuits for each quantum computer are the number of $SWAP$ operations added to be able to execute the circuit. The larger the depth, the larger the computational cost to execute the circuit in a quantum topology. Each $SWAP$ operation is a combination of three sequential controlled-x gates between the two qubits to be swapped.

Note in Table 8.3 that for any topology, optimization improves the circuit depth as expected due to the minimization of the $SWAP$ operations.

When, due to the quantum computer topology and the characteristics of the problem, $SWAP$

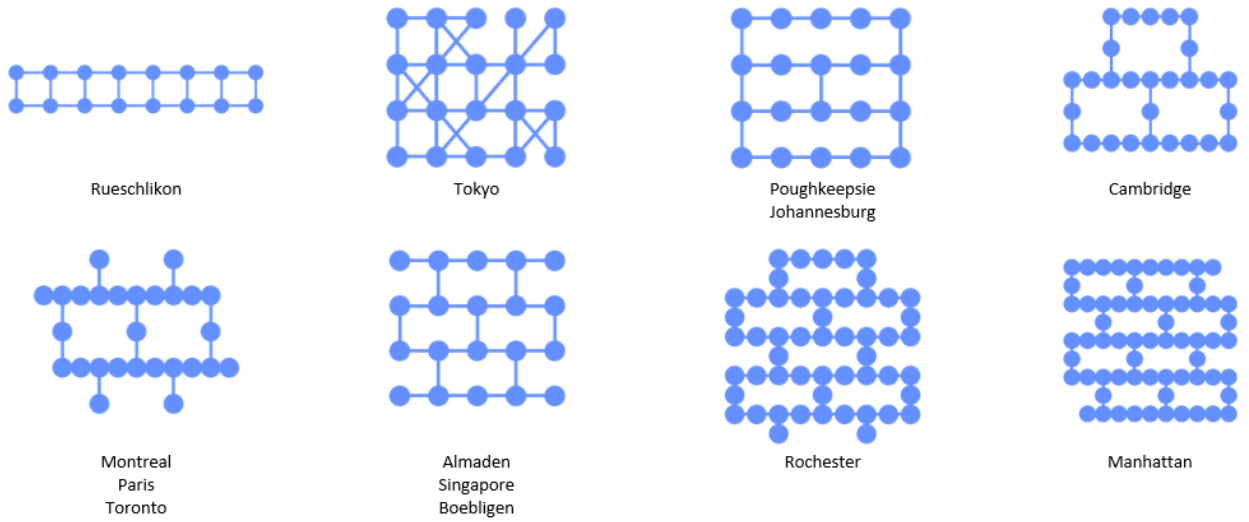


Figure 8.6: Quantum computers topologies.

Simulator	n_qubits	Without optimization				With optimization				
		10	15	20	50	10	15	20	50	60
Rueschlikon	16	62	98	-	-	56	98	-	-	-
Tokyo	20	57	90	145	-	36	56	122	-	-
Almaden	20	48	90	126	-	36	56	170	-	-
Johannesburg	20	60	102	139	-	36	56	76	-	-
Cambridge	28	72	104	184	-	36	56	76	-	-
Manhattan	65	36	116	145	449	36	56	76	196	581
Montreal	27	78	134	196	-	36	56	76	-	-
Rochester	53	69	104	181	481	36	56	76	667	-
chain_backend	-	36	56	76	196	36	56	76	196	236

Table 8.3: Benchmark. Depth analysis of the W State circuit for different topologies.

operations must be carried out to use all the qubits of the quantum computer, the depth increases considerably in comparison with other topologies, as for example for *Almaden, Singapore* and *Boebligen* for $n = 20$ (depth of 170). For the same problem size, in a topology which does not imply doing $SWAP$ operations, such as *Johannesburg* or *Poughkeepsie*, the depth decreases to less than a half. For larger problems than the executed in the experiments (for example $N = 50$), the best option would be to use the *Manhattan* topology due to the number of qubits, 65, and the multiple qubits distributions available without using $SWAP$ operations. However, using such a large quantum computer for considerably smaller problems is a brute solution.

The topology of some computers limits the number of used qubits in order to do not increase considerably the depth of the circuit. As a solution, we have designed an ideal topology for the problem we are solving. In the W state circuit we are using the qubits interaction: $x_0 - x_1, x_1 - x_2, \ldots, x_{n-1} - x_n, x_n - x_{n-1}, x_{n-1} - x_{n-2}, \ldots, x_1 - x_0$. Thus, the ideal topology

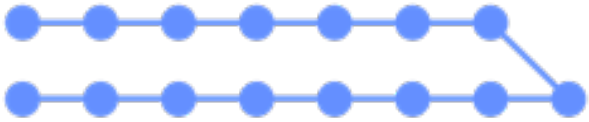


Figure 8.7: Chain topology designed as the ideal topology for the QIEDA approach for $n = 15$.

is a chain qubits distribution, named as *chain_backend* in Table 8.3. An example is shown in Figure 8.7. In the benchmark, for the problem of size 60, the chain topology improves *Manhattan* topology by a factor of 2.5.

Note that in Table 8.3, the modified version of the W state circuits depth grows linearly with the size of the problem for the *chain_backend* topology, so the implemented metaheuristic is considered to be efficient [Ross, 2019].

8.4 Conclusions

This chapter have presented a new quantum-inspired EDA approach to solve the TSP problem. The QIEDA uses a modified version of the W state quantum circuits to adapt the PLS process to sample new solutions during the algorithm runtime.

The results obtained by the QIEDA were analysed in terms of convergence and optimum solution found. The algorithm behaviour was compared to other state-of-the-art population-based algorithms. The QIEDA number of iterations until convergence remains constant with increasing number of cities of the TSP, and is smaller than in other algorithms. The solutions obtained by the QIEDA are competitive with the other algorithms, and the observed trend justifies the use of quantum computers to solve the TSP. The presence of quantum noise in the reproduction step of the algorithm improves its performance compared to others without noise. The QIEDA computational cost is also analysed by a benchmark. We have shown that the algorithm results are independent of the topology of the quantum computer chosen to be executed on. However, the topology is critical for the execution time due to the *SWAP* operations carried out by the quantum computers. We have proposed an ideal quantum computer topology to solve the TSP although other technologies as IonQ [3] uses all-to-all connected qubits configurations that may be as faster as the one proposed here. However, this technology does not allow all the quantum gates that the W state circuit involves and further studies should be carried out to adapt our approach.

Future work should include:

- The implementation of a W state quantum circuit able to sample the full individual matrix instead of executing a W state circuit per matrix row.
- Generalizing QIEDA approach for a general purpose would enforce the usability of the algorithm.

[3] https://ionq.com/

Chapter 9

QAOA for BN Structure Learning

9.1 Introduction

Chapter 8 focuses on adding a quantum perspective to EDAs by implementing a quantum-inspired approach, corresponding to *CQ* in Figure 1.1. On the other way around we find quantum approaches in which a classical contribution is needed (*QC* in Figure 1.1); that is, the VQAs (Section 4.3). In this chapter we present the use of this type of approach to face the Bayesian network structure learning (BNSL) problem, and an extensive analysis on the noise resilience for this problem. Although this chapter focuses on the application of VQAs, part of it shows relevant results applying quantum-inspired technologies (*CQ* in Figure 1.1) to the BNSL problem.

Regarding the high computational demands associated with BNs, two main problems have been studied in the literature: inference, which involves calculating a posterior probability distribution for some variables when observing the values of some other variables; and structure learning, which involves finding the optimal BN graph that best fits some given data. This chapter is focused on the latter type of problem.

Finding the structure of a BN giving some data is an NP-hard problem [Chickering, 1996], in which the number of possible structures grows more than exponentially (Equation 2.11).

More recently, the capabilities of quantum computers to reduce the required execution time when facing different optimization tasks and to solve very complex problems that may not be approachable with classic computing methods have attracted much interest. QC is based on quantum mechanics principles such as quantum entanglement and quantum superposition, which allow quantum algorithms to explore areas of the search spaces of optimization problems in a parallel and more efficient way.

Quantum annealing [Hauke et al., 2020] is a quantum heuristic that can solve certain optimization problems exponentially faster than classic approaches. The BNSL problem has been mapped to a quadratic unconstrained optimization problem (QUBO) to be solved by using quantum annealing [O'Gorman et al., 2015, Shikuri, 2020]. The company Fujitsu has recently proposed a new technology in which the performance of quantum annealing is

emulated classically. This quantum-inspired technology is digital annealing (DA) [Aramon et al., 2019].

Some studies [Streif and Leib, 2019] have proven that some types of optimization problems have landscape dispositions that make the quantum and simulated annealing methods converge to local optimal solutions, while the QAOA is able to overcome this limitation and provide better solutions. Quantum and simulated annealing have already been applied to BNSL; however, to the best of our knowledge, the use of the QAOA has not been found in the literature. In this chapter, we address the BNSL problem with the QAOA, and analyze the performance of different variants of the algorithm.

VQAs, and QAOAs in particular, are some of the most promising algorithms in the NISQ era, as their implementations optimize the number of utilized qubits, and moreover, the variational ansatzs are expected to offer resilience to quantum noise such as amplitude and phase damping errors [Xue et al., 2021, Sharma et al., 2020]. We also analyze the resilience of the algorithm to the presence of different types of quantum noise, in the particular case of the BNSL problem.

This chapter includes the developed methodology and results included in Soloviev et al. [2022a]. The implemented methodology is available in a GitHub repository[1].

Chapter outline

The outline of this chapter is organized as follows. Section 9.2 details the QUBO formulation in which this chapter is inspired. Section 9.3 describes how the QAOA ansatz is built and the characteristics integrated in our approach. Section 9.4 analyzes the performance of the QAOA approach, the resilience of the algorithm to quantum noise, and a real application of the algorithm for solving the BNSL problem. Section 9.4.4 shows further results of BNSL for large BNs using DA technologies. Section 9.5 rounds the paper off with the conclusions of our work.

9.2 QUBO formulation of BNSL

In this section, we describe the original QUBO formulation introduced in [O'Gorman et al., 2015], on which we base our approach. The formulation is based on four different Hamiltonians: H_{score}, which optimizes the likelihood of a structure given the input data; H_{max}, which ensures the maximum in-degree of each node to limit the Hamiltonian complexity; and H_{trans} and $H_{consist}$ which guarantee that the adjacency matrix that represents the BN is acyclic. The computed QUBO expression is,

$$H(\boldsymbol{A}, \boldsymbol{R}, \boldsymbol{Y}) = H_{score}(\boldsymbol{A}) + H_{max}(\boldsymbol{A}, \boldsymbol{Y}) + H_{trans}(\boldsymbol{R}) + H_{consist}(\boldsymbol{A}, \boldsymbol{R}),$$

where $\boldsymbol{A}$, $\boldsymbol{R}$ and $\boldsymbol{Y}$ are the quantum bits associated to the adjacency matrix, topological order, and the maximum in-degree restriction variables, respectively, for variables $X_1, X_2, \ldots, X_n$.

[1] https://github.com/VicentePerezSoloviev/QAOA_BNSL_IBM

The QUBO problem formulation for solving a BNSL problem with n nodes requires

$$v_{size} = n(n-1) + \frac{n(n-1)}{2} + 2n, \tag{9.1}$$

quantum bits, where $n(n-1)$, $\frac{n(n-1)}{2}$, and $2n$ are the number of bits associated with the adjacency matrix, the topological order, and the number of variables needed to restrict the maximum in-degree, respectively.

Each of the four different Hamiltonians that compose $H(\boldsymbol{A}, \boldsymbol{R}, \boldsymbol{Y})$ are deeply explained in this section.

9.2.1 H (score)

For H_{score} we need to introduce the concept of an adjacency matrix ($\boldsymbol{A}$):

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \tag{9.2}$$

where $a_{ij} = 1$ if there exists an arc from X_i to X_j and $a_{ij} = 0$ otherwise.

In this case, as BNs are represented as DAGs, the diagonal of this matrix is equal to zero, and thus, the bits of the diagonal are not required for the QUBO formulation. Then, $n(n-1)$ qubits are needed for the H_{score} Hamiltonian to learn a BN of n nodes, and

$$H_{score}(\boldsymbol{A}) = \sum_{i=1}^{n} H^i_{score}(\boldsymbol{a}_i), \tag{9.3}$$

$$H^i_{score}(\boldsymbol{a}_i) = \sum_{\substack{J \subset \{1,\dots,n\} \setminus \{i\} \\ |J| \leq m}} (w_i(J) \prod_{j \in J} a_{ji}), \tag{9.4}$$

where $\boldsymbol{a}_i = (a_{1i}, \dots, a_{ni})$, is the i-th column of $\boldsymbol{A}$, $w_i(J) = \sum_{l=0}^{|J|} (-1)^{|J|-l} \sum_{\substack{K \subset J \\ |K|=l}} s_i(K)$, in which $s_i(K)$ is the score of node i given the parent set K, and m is the maximum in-degree allowed.

Note that the constant term is $w_i(\emptyset) = s_i(\emptyset)$, which refers to the score of node X_i without its parents. If X_i has a single parent X_j, then the above equation simplifies to

$$H^i_{score}(\boldsymbol{A}) = w_i(\emptyset) + w_i(\{j\}) = s_i(\emptyset) + s_i(\{X_j\}) - s_i(\emptyset) = s_i(\{X_j\}).$$

Similarly, if X_i has two parents X_j and X_k,

$$\begin{aligned} H^i_{score}(\boldsymbol{A}) &= w_i(\emptyset) + w_i(\{j\}) + w_i(\{k\}) + w_i(\{j,k\}) \\ &= s_i(\emptyset) + (s_i(\{X_j\}) - s_i(\emptyset)) + (s_i(\{X_j\}) - s_i(\emptyset)) + w_i(\{j,k\}) \\ &= s_i(\{X_j\}) + s_i(\{X_k\}) - s_i(\emptyset) + w_i(\{j,k\}) \\ &= s_i(\{X_j\}) + s_i(\{X_k\}) - s_i(\emptyset) + s_i(\{X_j, X_k\}) - s_i(\{X_j\}) - \\ &- s_i(\{X_k\}) + s_i(\emptyset)) = s_i(\{X_j, X_k\}). \end{aligned}$$

9.2.2 H (max)

To ensure that the quantum algorithm only considers the maximum in-degree $m = 2$ to restrict the search space, the H_{max} Hamiltonian is implemented in a way such that $2n$ quantum bits are needed. These quantum bits are represented as a matrix:

$$\boldsymbol{Y} = \begin{bmatrix} y_{11} & y_{12} \\ \vdots & \vdots \\ y_{n1} & y_{n2} \end{bmatrix},$$

where $y_{ij} \in \{0, 1\}$ are random binary variables. $\boldsymbol{Y}$ represents a slack variable used to reduce the inequality constraint of the maximum in-degree to an equality constraint.

The corresponding Hamiltonian results in $H_{max}(\boldsymbol{A}, \boldsymbol{Y}) = 0$ if the restriction is met, and $H_{max}(\boldsymbol{A}, \boldsymbol{Y}) > 0$ otherwise. Thus,

$$\begin{aligned} H_{max}(\boldsymbol{A}, \boldsymbol{Y}) &= \sum_{i=1}^{n} H^i_{max}(\boldsymbol{a}_i, y_i), \\ H^i_{max}(\boldsymbol{a}_i, y_i) &= \delta_{max}(m - \sum_{j=1}^{n} a_{ij} - y_i)^2 = \\ &= \begin{cases} 0, & d_i \leq m \\ \delta_{max}(d_i - m)^2, & d_i > m \end{cases} \end{aligned}$$

where $y_i = \sum_{l=1}^{2} 2^{l-1} y_{il}$, $i = (1, \ldots, n)$, and $\delta_{max} \in \mathbb{R}^+$ is a prefixed penalization term.

9.2.3 H (trans) and H (consist)

To ensure the acyclicity of the adjacency matrix we need to implement two different Hamiltonians, $H_{trans}(\boldsymbol{R})$ and $H_{consist}(\boldsymbol{A}, \boldsymbol{R})$. The former uses the topological order to check the transitivity of the graph, and the latter checks the consistency between the topological order and the adjacency matrix.

A topological ordering of a directed graph is a linear ordering of its vertices such that for every arc $i \to j$ from vertex i to vertex j, i comes before j in the ordering ($i < j$). The topological order is represented as

$$\boldsymbol{R}_{top} = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nn} \end{bmatrix},$$

where r_{ij} can be equal to 1, only if $i \leq j$ and $r_{ij} = 0$ if $i > j$ for the given QUBO formulation. The lower triangular portion of $\boldsymbol{R}_{top}$ provides no additional information to the upper triangular portion of $\boldsymbol{R}_{top}$. Moreover, if a matrix is acyclic, then the trace of $\boldsymbol{R}_{top}$ is equal to zero. Considering this, the variables used for the QUBO formulation are represented as part of the matrix $\boldsymbol{R}_{top}$, where the diagonal of the matrix and all the elements below it have been

removed

$$\boldsymbol{R} = \begin{bmatrix} r_{12} & r_{13} & \cdots & r_{1n} \\ \ddots & r_{23} & \cdots & r_{2n} \\ \vdots & \ddots & \cdots & \vdots \\ \cdots & \cdots & \ddots & r_{n(n-1)} \end{bmatrix}. \tag{9.5}$$

Then $H_{trans}(\boldsymbol{R})$ is zero if the relation encoded in the $\boldsymbol{R}$ matrix is transitive and δ_{trans} otherwise:

$$\begin{aligned} H_{trans}(\boldsymbol{R}) &= \sum_{1 \le i < j < k \le n} H^{ijk}_{trans}(r_{ij}, r_{ik}, r_{jk}), \\ H^{ijk}_{trans}(r_{ij}, r_{ik}, r_{jk}) &= \delta_{trans}(r_{ik} + r_{ij}r_{jk} - r_{ij}r_{ik} - r_{jk}r_{ik}) = \\ &= \begin{cases} \delta_{trans}, & [(i \le j \le k \le i) \vee (i \ge j \ge k \ge i)] \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

$H_{consist}(\boldsymbol{A}, \boldsymbol{R})$ is zero if the order encoded in the $\boldsymbol{R}$ matrix is consistent with the structure encoded in $\boldsymbol{A}$, and $\delta_{consist}$ otherwise.

$$\begin{aligned} H_{consist}(\boldsymbol{A}, \boldsymbol{R}) &= \sum_{1 \le i < j \le n} H^{ij}_{consist}(a_{ij}, a_{ji}, r_{ij}), \\ H^{ij}_{consist}(a_{ij}, a_{ji}, r_{ij}) &= \delta_{consist}(a_{ji}r_{ij} + a_{ij}(1 - r_{ij})) = \\ &= \begin{cases} \delta_{consist}, & (a_{ji} = r_{ij} = 1) \vee (a_{ij} = 1 \wedge r_{ij} = 0) \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

where $\delta_{trans} \in \mathbb{R}^+$ and $\delta_{consist} \in \mathbb{R}^+$ are prefixed penalization terms.

9.3 Method

This section explains how the variables are deployed in the QAOA approach and how the Hamiltonian is transformed to quantum circuits. All the implemented software is codified by using Qiskit-0.18.1 [Qiskit contributors, 2023] and myQLM-1.5.1 [ATOS, 2021].

9.3.1 QAOA variables

To make the QAOA able to manage the QUBO variables $(\boldsymbol{A}, \boldsymbol{R}, \boldsymbol{Y})$, it is necessary to arrange them in such a way that they are represented as a vector. Thus, the previous QUBO variables are disposed as a vector of qubits with a size of v_{size} (Equation 9.1).

As explained in Section 4.4, the QAOA is a hybrid approach in which the classic part of the algorithm computes the cost function of the obtained solutions and the expectation value of all the solutions of the corresponding iteration. Our proposal also computes the maximum in-degree ($m = 2$) of the solutions and penalizes those that do not meet the restriction in a classic manner. Upon doing so, v_{size} reduces to

$$v_{size-QAOA} = n(n-1) + \frac{n(n-1)}{2} = \frac{3n(n-1)}{2}, \tag{9.6}$$

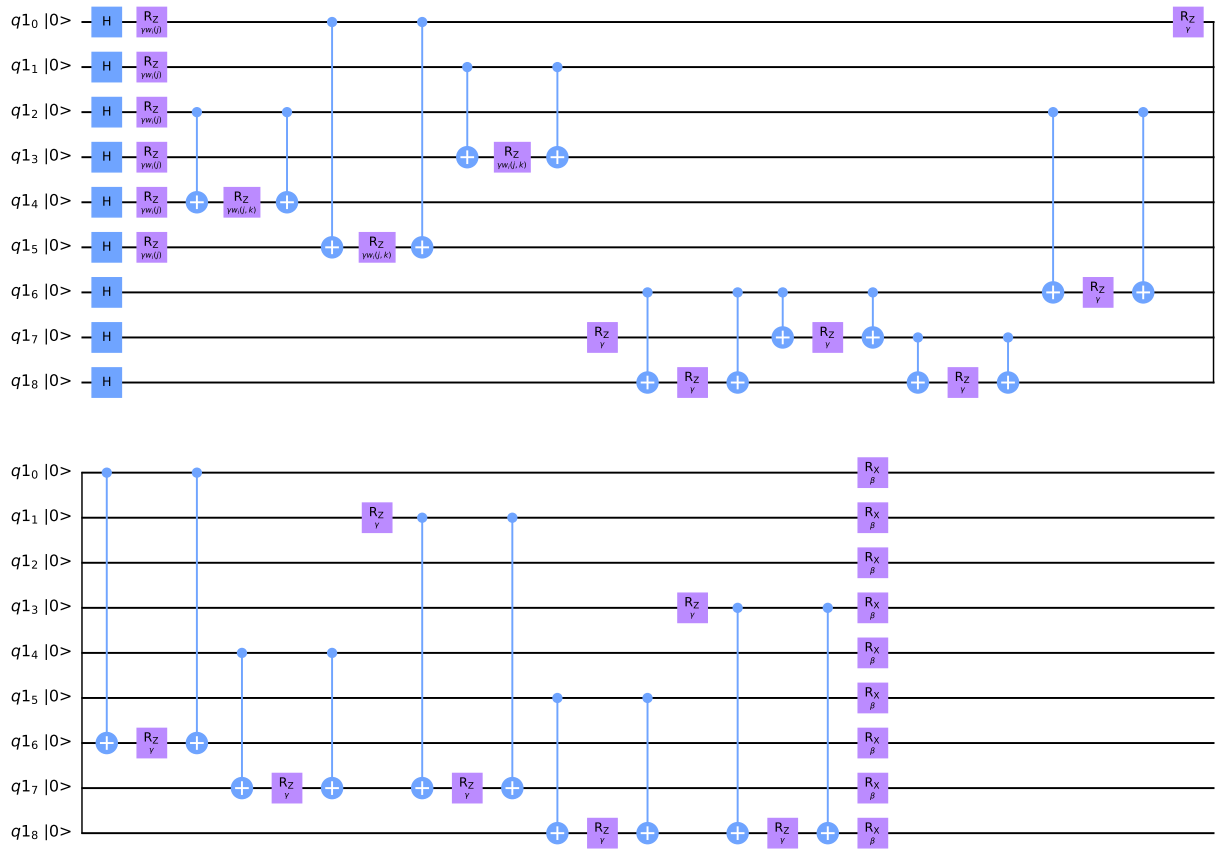


Figure 9.1: A QAOA circuit example for a BNSL problem with 3 nodes and 1 layer. This variational ansatz has $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ parameters as the parameters of the first layer of the circuit.

where $v_{size-QAOA} < v_{size}\ \forall\ n$, because the $\boldsymbol{Y}$ variables in the Hamiltonian are not considered.

Thus, the vector $\boldsymbol{q}$ needed to solve the BNSL problem for a BN of n nodes by using the QAOA is an array of size $v_{size-QAOA}$,

$$\begin{aligned}\boldsymbol{q} &= (q_1, q_2, \ldots, q_{n*(n-1)}, \ldots, q_{v_{size-QAOA}}) \\ &= (a_{12}, a_{13}, a_{1n}, \ldots, a_{n(n-1)}, r_{12}, r_{13}, r_{1n}, r_{23}, \ldots, r_{(n-1)n}),\end{aligned}$$

where a_{ij} and r_{ij} are defined in Equation 9.2 and Equation 9.5, respectively.

9.3.2 QAOA circuit

In Section 4.4, we have seen that the process of preparing the quantum state during the operation of the QAOA is composed of three elements:

1. Preparing an initial state of superposition.
2. Applying the cost operator $U(H_C, \gamma)$ (Equation 4.16).
3. Applying the mixed operator $U(H_B, \beta)$ (Equation 4.17).

9.3.2.1 Initial state

The initial state used during the QAOA is usually the superposition of all the basis states, which is defined as:

$$|\psi_0\rangle = \left(\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)\right)^{\otimes v_{size-QAOA}},$$

where $\otimes v_{size-QAOA}$ refers to the number of qubits used in the quantum state (Equation 9.6).

To reach the superposition state of all the possible basis states, we apply Hadamard gates to each qubit.

9.3.2.2 Applying the cost operator

As the maximum in-degree verification is implemented in the classic part of the VQA, the Hamiltonian to be implemented is reduced to

$$H(\boldsymbol{D}, \boldsymbol{R}) = H_{score}(\boldsymbol{D}) + H_{trans}(\boldsymbol{R}) + H_{consist}(\boldsymbol{D}, \boldsymbol{R}). \tag{9.7}$$

The Hamiltonian described in Equation 9.7 involves binary variables, and the QAOA needs the Hamiltonian to be transformed into a spin Hamiltonian where all the variables are spins in $\{-1, 1\}$. Thus, each binary variable X_i in the QUBO formulation must be transformed as $X_i \rightarrow \frac{1-Z_i}{2}$, where Z_i is the Pauli-Z operator (Table 4.1).

Thus, the QUBO formulation is transformed into a formula in which all the variables involved are $\boldsymbol{q}$. The QAOA is a circuit model-based approach, and thus, each Pauli operator Z_i is a quantum gate in the QAOA circuit. Each operator is a rotation-Z gate of qubit i, and each multiplication of two Pauli operators Z_iZ_j is a sequence of three gates in qubits i and j ($CNOT$ - R_Z - $CNOT$), as shown in Figure 9.2.

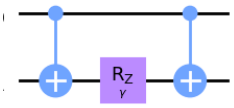


Figure 9.2: A multiplication of two Pauli operators Z_iZ_j is represented in the quantum circuit as a combination of two CNOT gates between qubits i and j and a rotation-Z gate in one of them.

Each Z_i gate has a rotation angle that is parameterized by γ and influenced by the structure evaluation scores (Equation 9.4).

9.3.2.3 Applying the mixed operator

The last step of the QAOA circuit is the mixed operator. This operator consists of applying a rotation-X gate in all the qubits of the circuit with parameter β.

An example of the resultant circuit is shown in Figure 9.1. For each extra layer, a cost and mixed operators $U(H_C, \boldsymbol{\gamma})$ and $U(H_B, \boldsymbol{\beta})$ should be added sequentially with their respective parameters to the actual circuit to increase p.

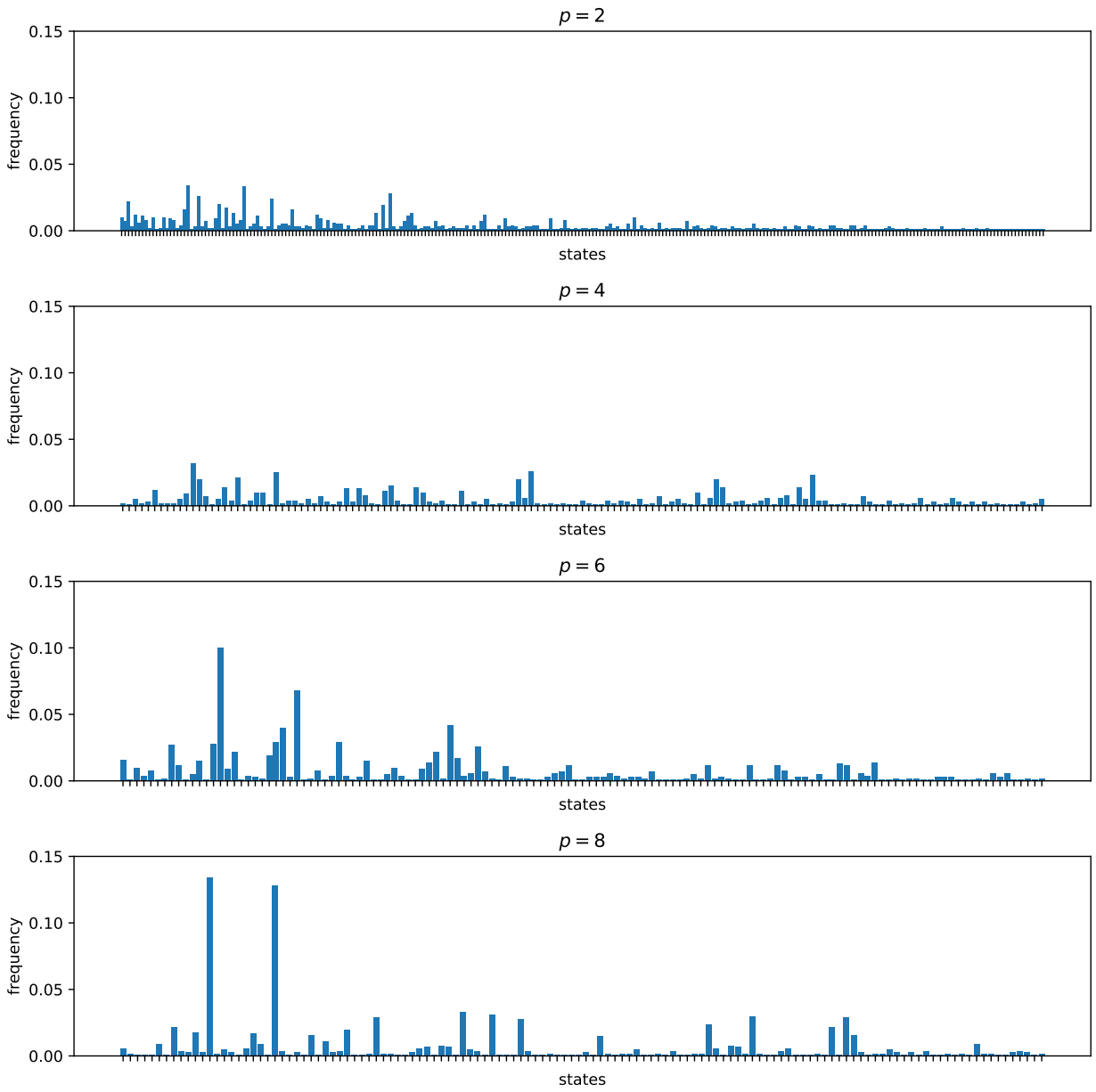


Figure 9.3: Histograms for different numbers of layers p for the same BNSL problem. The Y and X axes represent the frequency, and the solutions, respectively. The names of the solutions have been removed from the X-axis for aesthetics, but how they are sorted is the same for each subplot.

9.4 Results

In this section, some results are shown for the BNSL problem after applying the proposed QAOA using CVaR(α) (Equation 4.11). Some plots are first given to show how the QAOA performs in terms of cost function minimization (Section 9.4.1). Then, a performance evaluation of the QAOA considering different types of simulated noise is analyzed (Section 9.4.2).

Finally, a real example of BNSL is shown (Section 9.4.3).

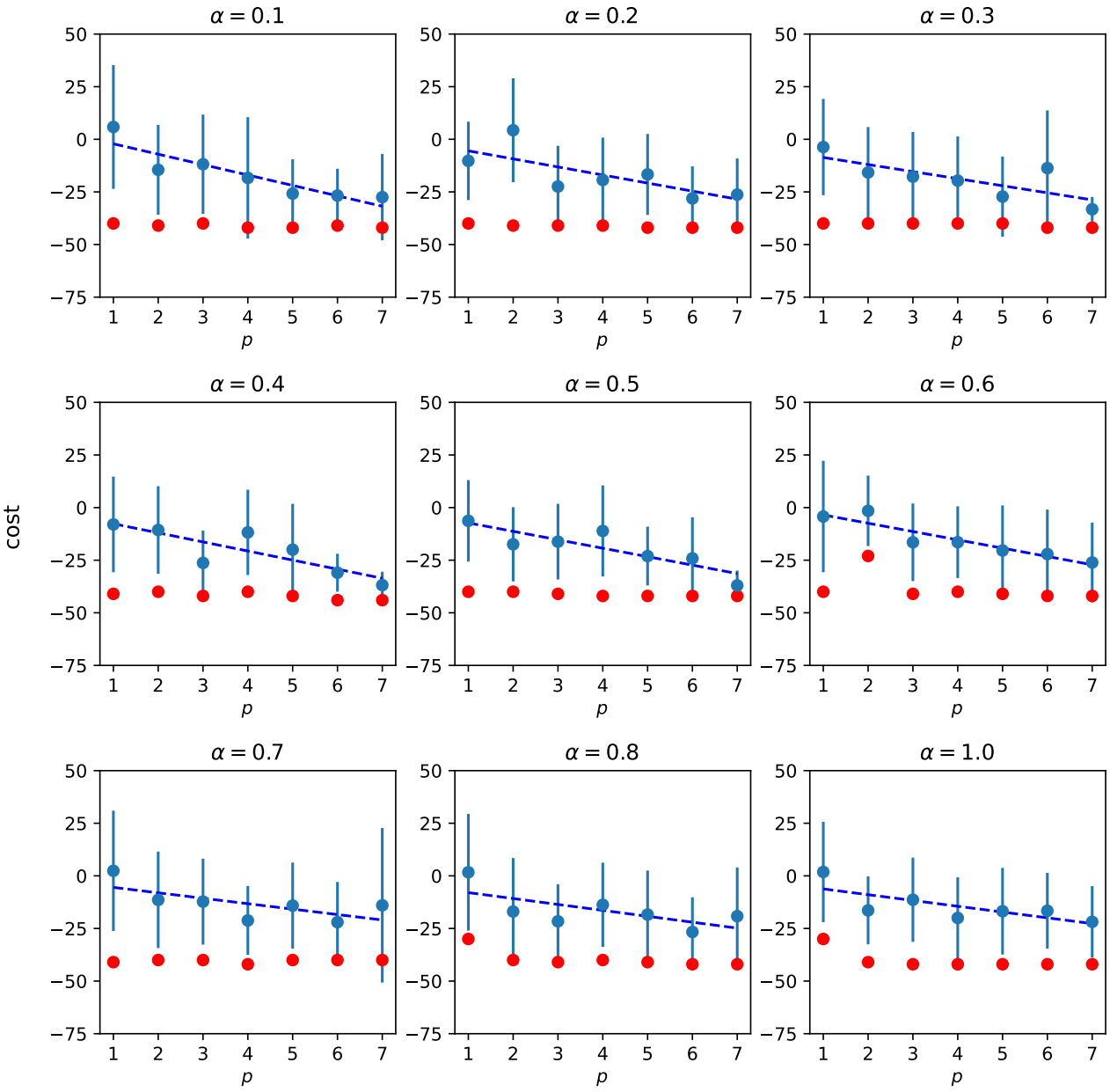


Figure 9.4: Minimization of the CVaR(α) (Equation 4.11) for different numbers of layers p (X-axis) in the QAOA circuit and different values of the parameter α. Blue dots and error bars correspond to the means and standard deviations of the best results found after executing the QAOA 50 times, respectively, and red denotes the minimum costs found in those executions. Dashed trend lines are plotted to guide the human eye.

Note that the real limitation of this algorithm has been the number of available qubits in the available architectures. In our experiments, the QAOA looks for the optimal BN structure in a search space containing 543 possible structures for $n = 4$ and 29281 structures for $n = 5$ (see Equation 2.11). This architecture restriction is imposed due to the number of qubits that

we can access at the moment in Qiskit and myQLM. Despite the fact that these problem sizes are far from those examined by the classic BNSL approaches, the number of qubits that companies such as IBM and Google are offering is increasing rapidly, and thus, the use of VQAs is increasingly justified.

The optimizer used in the implementation is the constrained optimization by linear approximation algorithm [Powell, 1994], which is widely used in the state-of-the-art VQAs [Bonet-Monroig et al., 2021].

9.4.1 QAOA performance

The QAOA approach aims at minimizing the uncertainty among the solutions obtained after completing the QAOA circuit measurement process, which is optimized by the minimization of the expectation value; see Equation 4.9. It is expected that by increasing the number of layers p of the circuit, the expectation value among the solutions must decrease. The task of the optimizer is to iteratively search the optimal parameters ($\boldsymbol{\gamma}_{opt}$, $\boldsymbol{\beta}_{opt}$) of the QAOA circuit to minimize the expectation value. When the optimizer converges to a solution, the parameters ($\boldsymbol{\gamma}_{opt}$, $\boldsymbol{\beta}_{opt}$) are set to those of the quantum circuit. Figure 9.3 shows an example histogram of the obtained solutions. This experiment is performed with different numbers of layers ($p = 2, 4, 6, 8$) to show the differences between the resultant histograms.

Figure 9.3 shows that increasing the number of layers, clearly minimizes the uncertainty. Note the existence of two clear optima with similar costs for $p = 8$; this is not as clear for $p = 2$. Moreover, a reduction in the number of solutions that are represented by the X-axis for increasing p is clearly visible. The solutions obtained with $p = 2$ have a density close to 0; for $p = 8$, they tend to have a density equal to 0 and thus become insignificant in the corresponding subplot.

As shown before, the QAOA approach is able to reduce the uncertainty among the solutions for the implemented Hamiltonian. However, the proposed approach heavily depends on the random pair of ($\boldsymbol{\gamma}$, $\boldsymbol{\beta}$) parameters from which the optimizer is initialized. Thus, depending on the initialization, different solutions might be proposed in different executions for the same Hamiltonian problem.

The minimization of the CVaR(α) (Equation 4.11) is analyzed next by using a random dataset of 4 variables. In Figure 9.4, a comparison of the performances achieved by the QAOA for different values of the parameter α and the number of layers p in the QAOA circuit is shown. Note that increasing the number of layers decreases the mean best cost although it increases the depth of the circuit and the computing time.

In Figure 9.4, we observe an improvement as p increases and α takes intermediate values. The best solutions are found for intermediate values of α in the range $[0.3, 0.5]$ and $p = 7$. Note that this improvement with increasing p is not as noticeable for large values of α ($\alpha \to 1.0$) as it is for the lowest values ($\alpha \to 0$). Despite these results, we claim that it is not necessary to increase the number of layers in the QAOA circuit to find the best results. Figure 9.4 shows cases in which the same results are obtained, with fewer layers but different values of α.

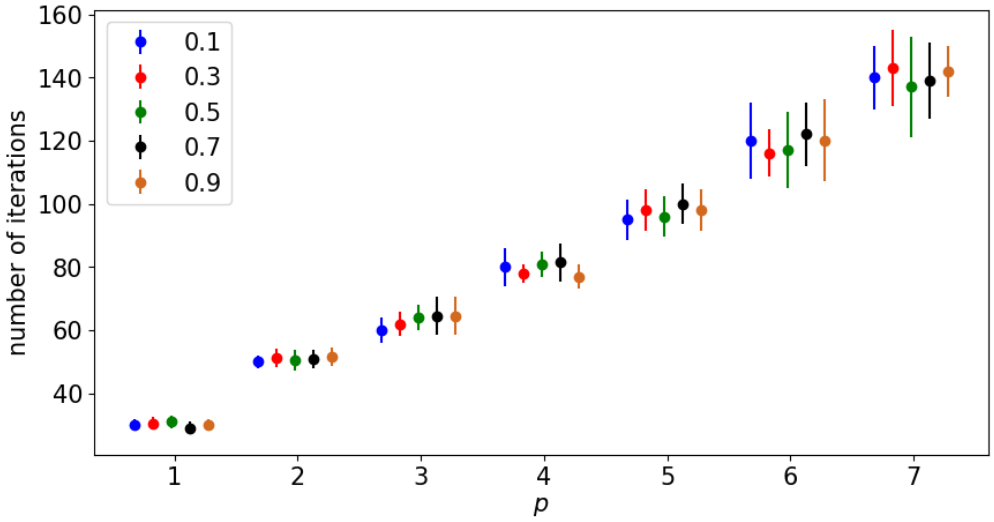


Figure 9.5: Means and standard deviations of the number of iterations until convergence after executing the QAOA 50 times for different values of p (X-axis) and α (colours).

Note that optimizing CVaR(α) considering α parameters, the implemented approach shares characteristics with EDAs. This type of algorithm tends to converge to local optima when the percentage of solutions selected to update the probability distribution is too low. However, as shown in Figure 9.4, the QAOA does converge to the same solution for any value of α. This is analyzed in Figure 9.5, where the mean numbers of iterations required until convergence are shown for different executions of the QAOA approach and different values of p. Independently of the value of α, the number of iterations remains approximately constant for the same value of p, whereas it increases with p.

9.4.2 Noise resilience

Two main disadvantages of NISQ computers are their limited numbers of qubits and the presence of quantum noise. Thus, there is a need to implement approaches that offer resilience to quantum noise and to optimize the number of qubits used to solve the given problem. It has been shown that VQAs can compensate for quantum errors such as over-/under-rotations [McClean et al., 2016]. However, a wide range of studies have analyzed the QAOA in different applications to determine the hard limit of its resilience to quantum noise [Shaydulin and Alexeev, 2019, Fontana et al., 2021, Urbanek et al., 2021, Kandala et al., 2019, Sun et al., 2021, Vovrosh et al., 2021]. In other words, we analyze how much noise the QAOA can bear without worsening its optimization behaviour.

To perform this analysis, different noise channels ε have been constructed to try to simulate the different decoherence quantum noises. The quantum channels which define the quantum noises are described in Section 4.6 using Kraus operators formalism.

Here, we parameterize the quantum channels as $\omega \in [0, 1]$, where $\omega = 0$ represents no quantum noise, and $\omega = 1$ is the maximal noise. For this analysis, amplitude damping (ε_{AD}), phase damping (ε_P) and depolarizing channels (ε_D) are considered.

Other types of quantum noises exist and are studied in the literature, such as the cross talk error, which is neglected in this study, as the main focus is to analyze the effects of decoherent noise channels. Similarly, the readout error is not considered as this type of noise is independent from the ansatz design.

For this study the amplitude and phase damping noises are only applied to the one-qubit gates, while the depolarization noise is applied to the two-qubit gates, such as *CNOT* gates.

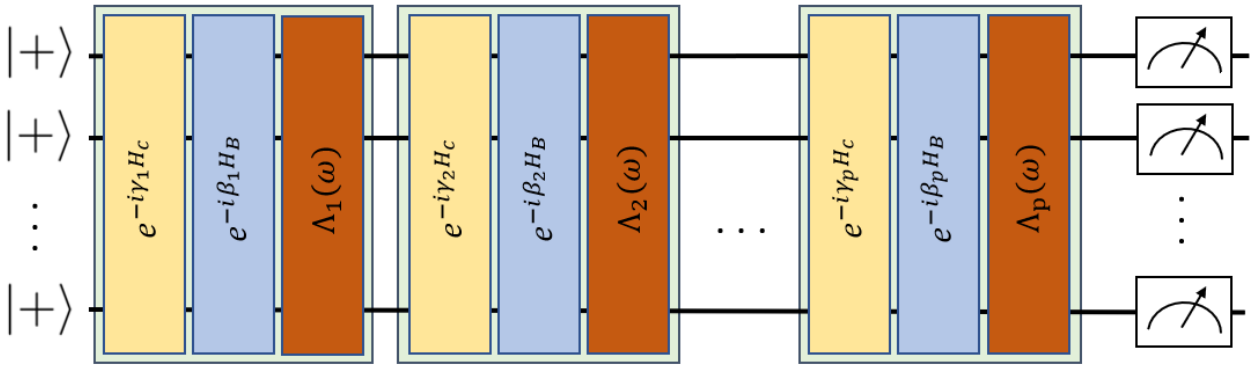


Figure 9.6: A quantum parametric circuit with p layers and $2p$ parameters. The initial state is a superposition of all the possible computational states, and after applying the p layers, a measurement along the Z axis of all the qubits is performed. Each layer is composed by the noise channel and both the cost and mixed operators.

To carry out these simulations, a quantum circuit has been designed alternating the QAOA operators (Equation 4.16 and Equation 4.17) with each noise channel operators parameterized by ω. By this way, each layer of the QAOA is accompanied by a noise operator, trying to replicate the behavior of the processes in real quantum computers. Three different quantum circuits are used in order to analyze each quantum channel separately. The quantum parametric state is defined as

$$\varepsilon(|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle) = \Lambda_p(\omega)U(H_B,\beta_p)U(H_C,\gamma_p)\dots\Lambda_1 U(H_B,\beta_1)U(H_C,\gamma_1)\,|s\rangle\,,$$

where $p \geq 1$, $\boldsymbol{\gamma} = (\gamma_1,\dots,\gamma_p)$, $\boldsymbol{\beta} = (\beta_1,\dots,\beta_p)$, $|s\rangle$ is the uniform superposition state over all possible computational states, and $(\Lambda_1, \dots, \Lambda_p)$ are the noisy operators that simulate each quantum channel. An example of a quantum circuit with p layers is shown in Figure 9.6 where the orange operators represent the quantum channels.

The operators used in the circuit construction represent the same quantum channel and affect to all the qubits in the quantum system. Their application is defined onto a quantum state as:

$$\varepsilon(\psi) = \Lambda(\omega)\psi = (\bigotimes_{i=1}^{v_{size-QAOA}} \Lambda_i^{(1)})\psi,$$

where $\varepsilon(\psi)$ is the resultant state after applying the quantum channel over the state ψ, ω is the parameter of the quantum channel and $\Lambda^{(1)}$ is the application of the one-qubit quantum channel in each qubit of the QAOA ansatz.

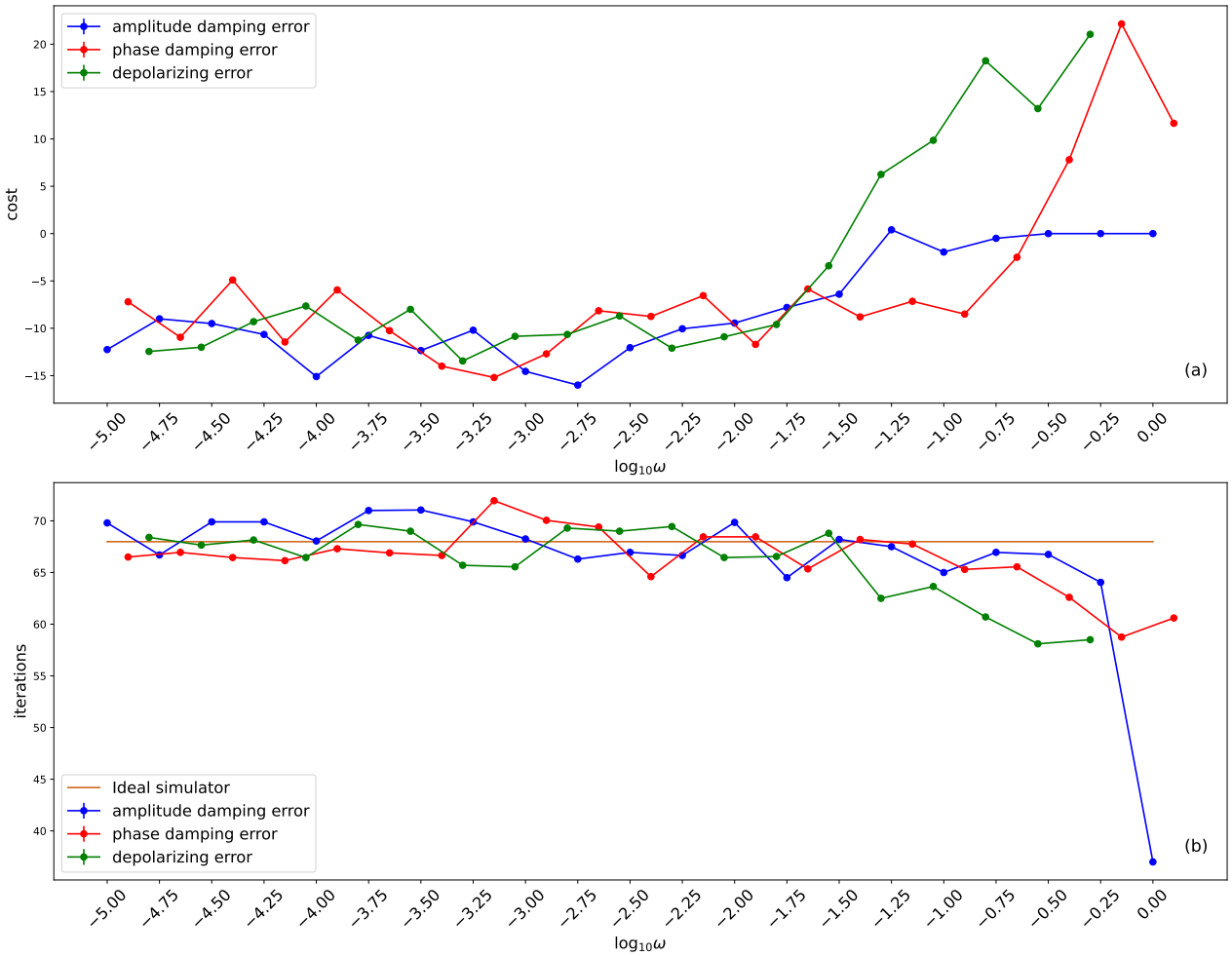


Figure 9.7: Mean best costs (a) and mean numbers of iterations until convergence (b) as a function of the noise amplitude ω. Blue, red and green lines represent the simulated amplitude, phase damping and depolarizing noise models, respectively. For the detailed mean and standard deviation values of these experiments, see Table 9.1. Fifty different executions of the QAOA algorithm were run for the 4 node BNSL problem, and $p = 3$.

Figure 9.7(a) shows the resilience of the algorithm to the three types of previously explained noise channels. The figure shows the mean best costs for different values of ω. Table 9.1 show mean and standard deviations of the numerical results.

To analyze the resilience of the simulated noises, we are interested in, fixing a range of mean cost values, knowing which is the value of ω that makes the mean best cost found to be out of this range. Since in Figure 9.7(a) for the different noises when there is nearly no noise ($\omega \to 0$), the cost values are within the range $[-15, -5]$, we are interested in the values of ω at which the different noise channels make the found costs to leave this range. Thus, we set the values of $\omega = 10^{-2}$, $\omega = 10^{-1.5}$, $\omega = 10^{-0.75}$ for the depolarized, amplitude damping and phase damping noise channels, respectively.

The number of iterations needed by the algorithm to converge are shown in Figure 9.7(b)

	cost						convergence					
	AD		PD		DE		AD		PD		DE	
$log_{10}\omega$	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
-5.00	-12.25	10.80	-7.20	15.61	-12.45	8.96	69.80	4.70	66.50	6.73	68.40	6.64
-4.75	-9.00	10.53	-10.95	10.80	-12.00	7.94	66.70	6.34	66.95	5.13	67.65	6.04
-4.50	-9.50	10.69	-4.90	11.49	-9.30	9.65	69.90	7.79	66.45	8.11	68.15	7.80
-4.25	-10.65	8.97	-11.45	10.95	-7.65	11.34	69.90	7.83	66.15	5.95	66.45	5.36
-4.00	-15.10	7.75	-5.95	9.58	-11.25	11.50	68.05	5.48	67.30	5.53	69.65	4.36
-3.75	-10.75	10.97	-10.25	11.54	-8.00	12.53	71.00	9.00	66.90	5.18	69.00	5.52
-3.50	-12.35	9.17	-14.00	8.47	-13.45	7.47	71.05	8.01	66.65	5.65	65.70	6.06
-3.25	-10.20	13.78	-15.20	8.79	-10.85	10.98	69.90	8.47	71.95	7.16	65.55	6.07
-3.00	-14.55	7.55	-12.70	8.86	-10.65	8.64	68.25	7.35	70.05	9.41	69.30	5.92
-2.75	-16.00	8.07	-8.15	13.69	-8.70	11.72	66.30	5.20	69.40	7.61	69.00	6.28
-2.50	-12.05	8.84	-8.75	13.18	-12.10	10.34	66.95	5.52	64.60	6.94	69.45	7.94
-2.25	-10.05	9.40	-6.55	12.15	-10.90	8.23	66.65	4.69	68.45	6.63	66.45	6.37
-2.00	-9.45	11.91	-11.70	9.59	-9.60	12.33	69.85	6.18	68.45	6.10	66.55	5.07
-1.75	-7.80	12.76	-5.85	12.78	-3.40	9.33	64.50	5.84	65.35	5.29	68.80	7.70
-1.50	-6.40	7.16	-8.80	11.62	6.25	15.34	68.20	5.52	68.20	6.96	62.50	4.88
-1.25	0.40	9.30	-7.15	10.16	9.85	16.68	67.50	6.06	67.75	6.22	63.65	5.58
-1.00	-1.95	5.67	-8.50	10.04	18.25	13.26	65.00	6.05	65.30	6.39	60.70	4.66
-0.75	-0.50	2.24	-2.50	7.25	13.20	12.89	66.95	6.48	65.55	5.86	58.10	4.59
-0.50	0.00	0.00	7.80	12.88	21.05	11.81	66.75	5.76	62.60	3.76	58.50	4.57
-0.25	0.00	0.00	22.15	14.53	20.01	10.30	64.05	6.73	58.75	4.02	59.10	4.59
0.00	0.00	0.00	11.65	13.35	21.50	11.20	37.00	0.00	60.60	5.86	58.60	4.80

Table 9.1: Mean best costs (μ) and standard deviations (σ) found for different values of the ω parameter over 50 executions of the QAOA approach for the BNSL problem, and the mean number of iterations (μ) and standard deviations (σ) until convergence for 50 executions. AD, PD and DE represent amplitude damping, phase damping, and depolarizing simulated errors, respectively.

(supplementary numerical results are at Table 9.1). In the figure we can generally observe a decreasing trend in the number of average iterations correlated with the growth of ω. The orange line shows the average number of iterations required to run the algorithm with the ideal simulator without quantum noise. Analyzing the values of ω identified in Figure 9.7(a), we can see how approximately at the same values, in Figure 9.7(b), there is a noticeable decrease tendency in the number of average iterations. Considering the worsening of the mean best cost found and the decrease in the number of iterations we can conclude that the algorithm is falling into local optima solutions when $\omega \to 1$.

Considering the previous analysis, we can present the following conjectures which will be extended below: (i) the approach is less resilient to the depolarization noise channel than to the other channels; (ii) the amplitude damping noise channel makes the mean best cost to converge to solutions with costs close or equal to zero for values of $\omega \geq 10^{-1.25}$; (iii) the approach is resilient to the phase damping noise channel up to values of $\omega \geq 10^{-1}$.

Firstly, despite the fact that the depolarized noise channel has only been applied on the two-qubit gates, the design of the variational ansatz from the QUBO makes the number of

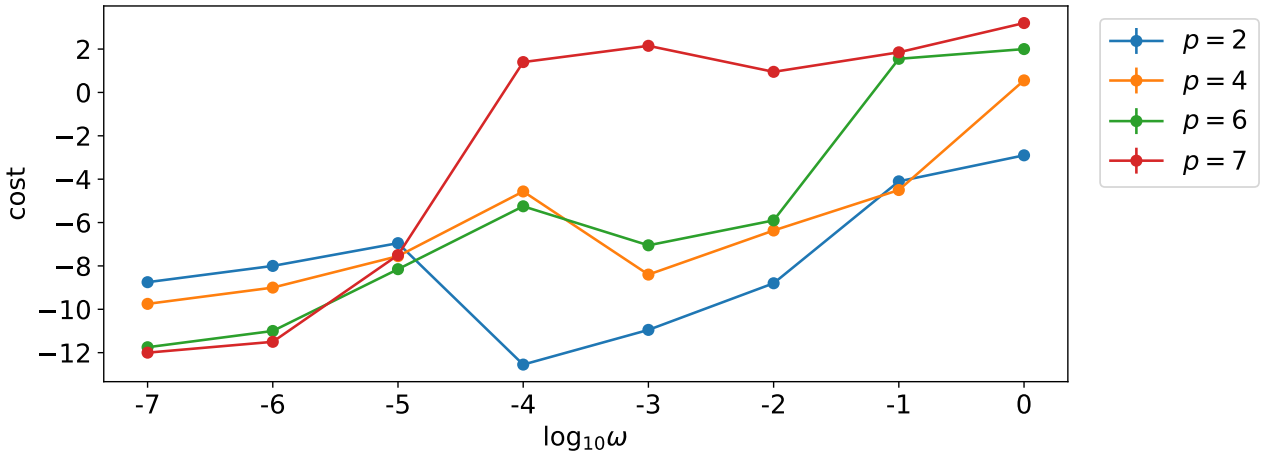


Figure 9.8: Mean best cost obtained for different values of the parameters ω and p for 50 executions of the QAOA algorithm for the BNSL problem considering the amplitude damping error.

CNOT gates high and therefore the depolarized noise channel has a high impact on the results shown in Figure 9.7(a). Consequently, we can say that the algorithm is resilient to this noise channel up to values of $\omega = 10^{-2}$. Higher values of ω rise noticeably the mean best cost found increasing also the standard deviation between solutions (Table 9.1).

Secondly, it is worth noting that amplitude damping is the only noise channel that converges to mean best costs of zero with no standard deviation, while the others reach mean best cost values above 15 with a high standard deviation among the best results found when $\omega \to 1$ (Table 9.1). Due to the effects that the amplitude damping noise channel has on the quantum states, we can affirm that the algorithm is converging to a partial solution, in which only some arcs are equal to those in the real BN, but all other values in the adjacency matrix are zero. The QUBO contemplates that those solutions such as $|00\cdots0\rangle$ should be penalized, and so it does not converge to them. Rather, in this case it is converging to a solution in which a large majority of the values in the adjacency matrix are zeros, except for the correctly pointed arcs. Therefore, for maximum values of ω, the mean best cost of the solutions found is better than what we find for maximum values of ω with other noise channels.

Thirdly, the phase damping error channel is the most resilient if only the value of ω that causes the mean best cost to fall out of the range of values fixed in the analysis of Figure 9.7(a) is taken as a reference. However, when $\omega \geq 10^{-0.75}$ the standard deviations among the solutions are increased (Table 9.1).

After analysing Figure 9.7, we conclude that our approach has better resilience to the amplitude damping error and phase damping error than to depolarizing error. However, we believe that it is of interest to analyze the convergence noted for the case of the amplitude damping noise channel.

In Figure 9.8, a deeper analysis of the amplitude damping channel is shown for different values of p. The figure shows the influence of the quantum noise on the results based on the number of layers p of the QAOA circuit. As p increases, the depth of the circuit also increases, and

thus, a greater part of the qubit lifespan will be executed outside the coherence times defined by T1 and T2. The results obtained by the QAOA for higher values of p are much worse than those obtained with low values of the parameter, and in most cases a worsening of the mean best cost is observed as ω increases. From this analysis we conjecture that as the number of layers p increases, the QAOA becomes less resilient to the amplitude damping error.

9.4.3 BNSL from real-world data

In this section, the QAOA approach is applied to a real BNSL problem by using the `Cancer`[2] benchmark. The `Cancer` BN (Figure 9.9) has 5 discrete variables, from which we sampled 3 different datasets using probabilistic logic sampling [Henrion, 1988] with 500, 1000 and 10000 instances. The structures provided by the QAOA are compared to the original BN structure through the structural Hamming distance metric, where a value of zero means that the QAOA approach fully recovers all the arcs of the original BN. We consider two different structure evaluation scores: the BIC and BDeu scores. This experiment is limited to 5 nodes due to the limit of qubits we can access with Qiskit and myQLM, for which the search space is composed of 29281 possible BN structures.

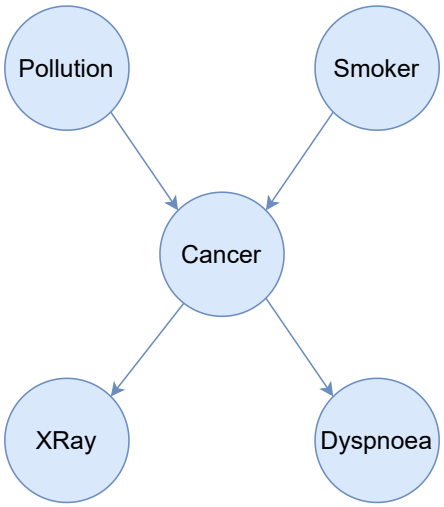


Figure 9.9: Original `Cancer` BN structure composed of 5 nodes that represent 5 discrete variables and 4 arcs.

We compare the results obtained by the QAOA approach with those of two score-based algorithms: the hill climbing (HC) [Gámez et al., 2011] and tabu search [Ji et al., 2011] algorithms; with that of a hybrid algorithm: max-min hill climbing (MMHC) [Tsamardinos et al., 2006]; and with that of the simulated quantum annealing (SQA), which is a version of quantum annealing executed in the quantum learning machine [ATOS, 2021]. The QAOA results are shown in Table 9.2 for different values of α, where the number of layers p is optimized for each case. The parameters of the algorithms are optimized, and the best results are shown in Table 9.2.

[2] https://www.bnlearn.com/bnrepository/discrete-small.html#cancer

		500		1000		10000	
	α	BIC	BDeu	BIC	BDeu	BIC	BDeu
HC	-	4	4	4	4	0	0
Tabu	-	4	4	4	5	0	0
MMHC	-	4	4	4	4	0	0
SQA	-	5	5	4	5	3	2
QAOA	0.9	0	0	1	0	0	1
QAOA	0.7	1	1	1	0	0	0
QAOA	0.5	1	1	0	1	1	1
QAOA	0.3	0	0	1	0	0	1

Table 9.2: Comparison of the QAOA approach with three different classic approaches and simulated quantum annealing. The experiment is executed 10 times for each of the non-deterministic (SQA and QAOA) approaches (the best results are shown), and three different dataset sizes are simulated with 500, 1000 and 10000 instances. BIC and BDeu scores are used for the local BN structure evaluation. The structural Hamming distance metric is shown for each experiment.

The QAOA approach obtains better results than those of classic approaches regardless of the α parameter for a low number of instances. As the number of instances increases, the α parameter seems to have a larger influence on the obtained results. The QAOA improves upon the results of the SQA in all the experiments.

9.4.4 Large BNs learning

Previous results in this chapter showed the BNSL using QAOA. However, to learn larger BNs, a bigger quantum device would be necessary. As we are restricted to the characteristics of nowadays's devices, here we shown some practical results using a quantum-inspired technology provided by Fujitsu [Aramon et al., 2019]; aka, Digital Annealing (DA). This technology allows to simulate a large number of qubits. Figure D.1 shows the relation between the number of needed (simulated) qubits and the size of the problem.

The performance of this technology is hereby compared to HC, Tabu and MMHC for BNSL of 20 and 50 nodes. The former experiment uses Child dataset[3] while the latter uses a BN random generator which firstly proposes a BN and then uses sampling methods for generating a new dataset. Dataset with 1000, 5000, 10000 and 50000 instances were sampled. SHD is used to compare the results found by the approaches compared to the original BN structure. Note that QAOA and SQA are not benchmarked due to device restrictions. Results are shown in Appendix D.

Figure D.2 shows the results of BNSL for 20 nodes. It is observed an improving performance with decreasing size of the dataset. For 1000 instances, DA outperforms the rest of approaches, and, for 5000 instances, only MMHC outperforms DA.

Figure D.3 shows the results of BNSL for 50 nodes. DA approach is still outperforming Tabu

[3] https://www.bnlearn.com/bnrepository/

and HC approaches, independently of the size of the dataset. However, MMHC outperforms DA for all the cases.

9.5 Conclusions

In this chapter the BNSL problem was approached by using the QAOA variational quantum algorithm. The problem was transformed into a Hamiltonian energy function, and then translated into a QAOA parametric circuit to be optimized into a classic loop.

A remarkable uncertainty reduction across all the possible solutions was shown for increasing number of QAOA circuit layers. We introduced the concept of the CVaR(α) to reduce the number of solutions selected for computing the expectation value of each iteration. Considering this new parameter, we could observe that it was not necessary to increase the number of circuit layers to obtain the best results since the QAOA was able to converge to similar solutions by tuning the α parameter.

The NISQ-era quantum computers are characterized by the quantum noise embedded in these systems. We analyzed the performance of our approach while simulating different types of quantum noise, and our results show that the QAOA is resilient to quantum noise over a range of noise amplitudes. More specifically, our approach offers good performance when considering the amplitude damping error, which was more deeply analyzed.

Our approach was applied to the `Cancer` benchmark and a comparison with other optimizers was shown with different structure evaluation scores and dataset sizes. The QAOA found the global optimum for every size-score combination and seemed to outperform classic and quantum approaches on any dataset.

Considering the results obtained in this chapter, we believe that the use of VQAs to solve the BNSL problem is justified.

Future research lines are listed below:

- Considering the warm starting scenario [Egger et al., 2021].
- Considering other types of classical optimizers.
- Facing the BNSL with other types of quantum approaches.
- Facing larger BNSL problems.

Chapter 10

VQAs Parameter Tuning with EDAs

10.1 Introduction

Chapter 9 presented a hybrid approach (*QC* in Figure 1.1) between classical and quantum computations for facing the BNSL problem in which a classical optimizer was used to tune the parameters of a quantum system *ansatz*. In this chapter, we propose to use EDAs to approach the *ansatz* parameter optimization. Moreover, we compare the use of different EDAs to face the VQAs *ansatz* parameter tuning. The results show UMDA, EGNA and SPEDA compared to the state-of-the-art classical optimizers used for this task, in two different VQAs, analyzing the quality of the solutions found and their computational runtime. EDAs have demonstrated to achieve very good results in continuous space optimization for a wide range of problems [Dasgupta and Michalewicz, 2014], and thus, we believe that these algorithms can provide competitive results to approach this problem and overcome limitations such as the Barren plateau problem [McClean et al., 2018].

This chapter includes the developed methodology and results included in Soloviev et al. [2022c, 2023b]. The implemented methodology is available in a GitHub repository[1], and was added as a new functionality to Qiskit [Qiskit contributors, 2023] library as a new optimizer[2].

Chapter outline

The outline of this chapter is organized as follows. Section 10.2 introduces the different variants done over the EDAs to compare their performance. Section 10.2.1 and Section 10.2.2 shows numerical results for QAOA and VQE *ansatz* parameter optimization. Section 10.2.3 simulates a molecule with different intensities of quantum noises where the EDA variants are compared. Section 10.2.4 shows the hyper-parameter dependence in EDA variants. Section 10.3 rounds the chapter off with further conclusions and future work.

[1] https://github.com/VicentePerezSoloviev/EDA_QAOA
[2] https://docs.quantum.ibm.com/api/qiskit/qiskit.algorithms.optimizers.UMDA

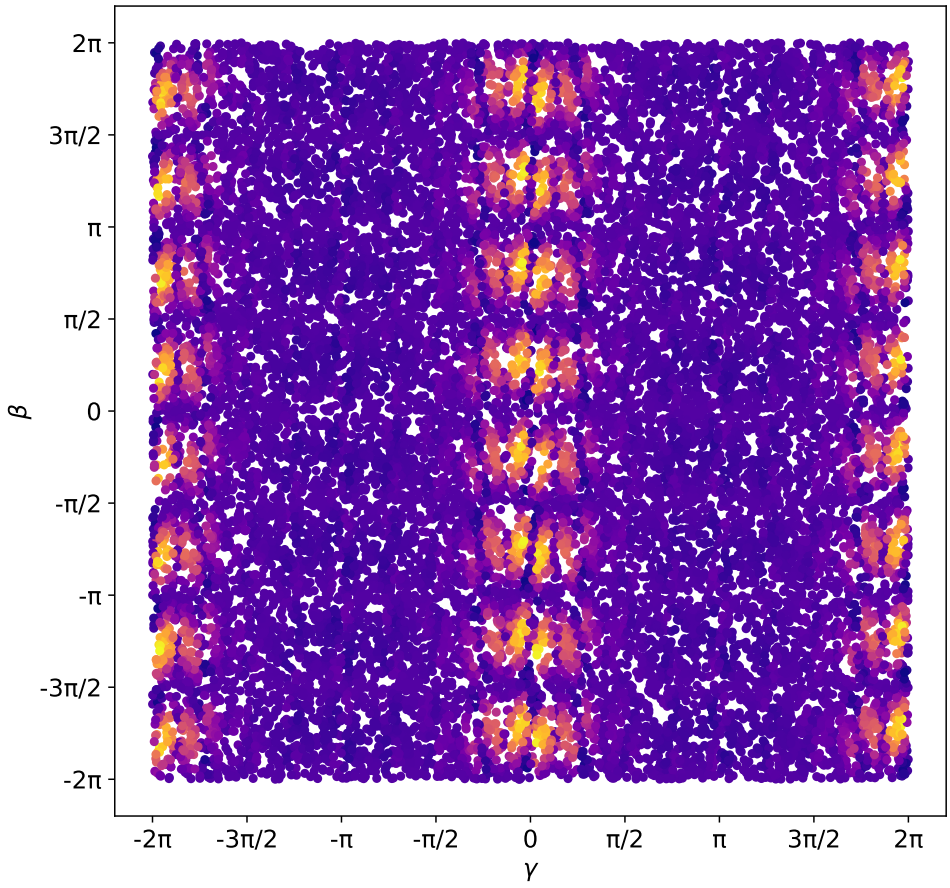


Figure 10.1: Optimization landscape where Y- and X-axis represent, respectively, β and γ QAOA *ansatz* parameters for $p = 1$ layer, and the lighted regions over the purple background represent the optimum parameters that minimize the expectation value (Equation 4.9). Although the parameter tuning has been restricted to $\boldsymbol{\theta} \in [0, 2\pi]^n$, the landscape of $\boldsymbol{\theta} \in [-2\pi, 2\pi]^n$ is displayed to show that -2π is equivalent to 2π.

10.2 Experimental results

In this section we show some numerical results when comparing different EDA complexities to some state-of-the-art gradient-free and -based algorithms. Note that all the implemented experiments for this study have been coded in Python using Qiskit-0.21.2 [Qiskit contributors, 2023] and EDAspy-1.0.2[3][4] Python packages. Different optimizers have been executed to tune the parameters of the QAOA *ansatz* and the VQE TwoLocal *ansatz*, both to optimize the same benchmark, an instance of the well-known MaxCut benchmark for 10 variables and 10 qubits ($n = 10$). Moreover, Section 10.2.3 shows some experimental results using different optimizers for tuning the parameters of an VQE *ansatz* used to simulate a molecule. Section 10.2.4 analyzes the hyper-parameter tuning of the EDA approach.

All the algorithms have been configured to a maximum of 100 iterations and the number of

[3] https://github.com/VicentePerezSoloviev/EDAspy
[4] https://pypi.org/project/EDAspy/

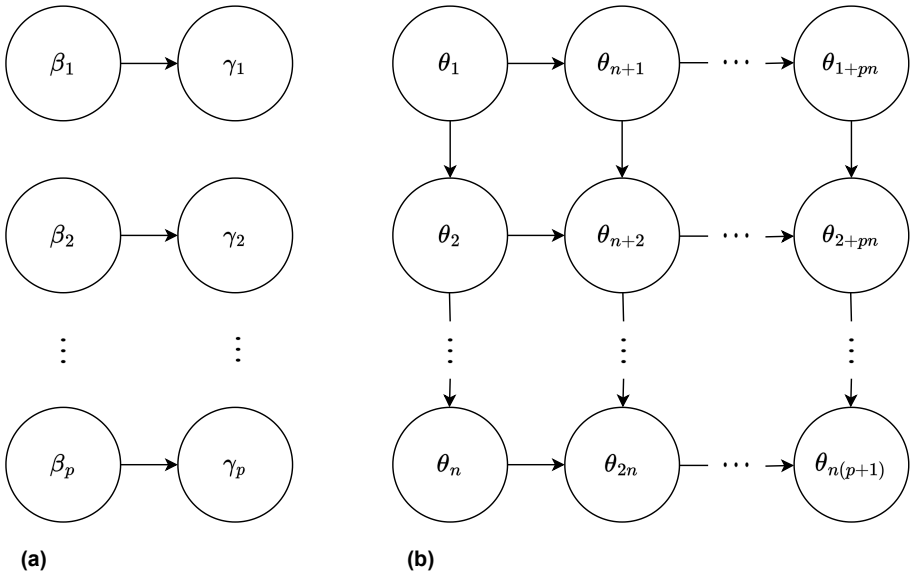


Figure 10.2: GBN structures designed for QAOA (a) and VQE TwoLocal (b) *ansatz* parameter tuning embedded by EGNA. Each node represents a parameter in the *ansatz*, and an arc between two nodes represents a linear Gaussian dependency between both variables.

shots for the *ansatz* is $t = 1000$. In the case of the EDAs, the selection ratio has been set to $\delta = 0.5$ for all the variants, the population size is $N = 60$, and the EDA is considered to have converged if after 20 generations, the algorithm has not improved the best solution found. Note that all the EDAs have been implemented as elitist approaches, where the best individuals of each iteration remain in the future, so that the algorithm never worsens the best results found in previous iterations, and also SPEDA's archive-based feature has been omitted to perform a fair comparison with the other EDAs ($l = 1$, where l is the archive length).

Different works in the literature [Brandao et al., 2018, Akshay et al., 2021] have shown that the QAOA *ansatz* parameters in the same layer are dependent, and there exist different optimum configurations for γ_i and β_i for which the expectation value (Equation 4.9) is minimum. Figure 10.1 shows an example for the MaxCut problem instance, with $p = 1$, where the optimization landscape is shown. Lighted areas over the purple background represent the optimum parameters for the QAOA *ansatz*, which our EDA approach should find.

Due to this dependency between γ_i and β_i in the same layer, in the following experiments the GBN structure embedded by the EGNA approach has been fixed. An example of the embedded GBN in the EGNA approach to tune QAOA *ansatz* parameters with p layers is shown in Figure 10.2(a), where each node represents a parameter in the QAOA *ansatz*, and γ_i and β_i are connected for each layer. Thus, the runtime of EGNA is expected to be reduced, as the structure of the GBN is not learned, and only the parameters of the model are updated according to the provided data.

Following the same strategy, a fixed structure has been proposed for the VQE *ansatz* parameter tuning using EGNA. In this case, as in each layer a quantum parametric gate is implemented

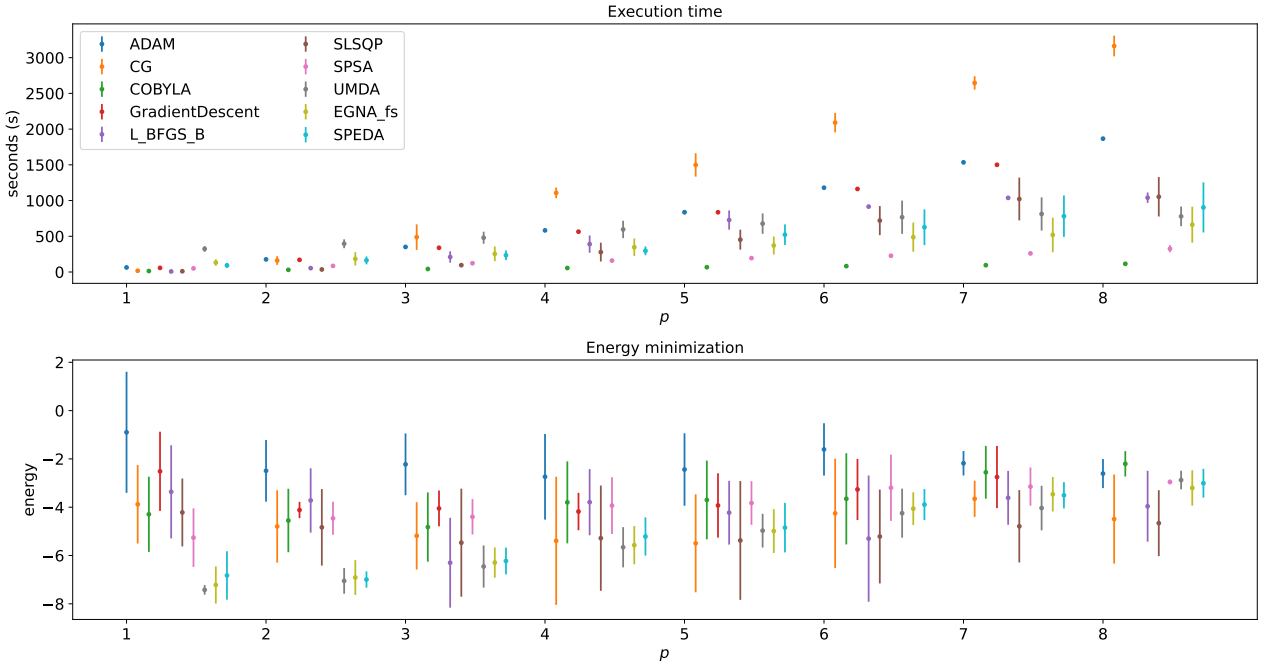


Figure 10.3: Comparison of the different EDA variants with the other state-of-the-art optimizers for the QAOA *ansatz* parameter tuning. The top panel shows the mean computation time and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \ldots, 8\}$. The bottom panel shows the mean best expectation value achieved (Equation 4.9) and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \ldots, 8\}$.

for each qubit, we have designed a structure in which each parameter $\boldsymbol{\theta}_i$ is connected to parameter $\boldsymbol{\theta}_{i+1}$ in the same layer, justified by the subsequent entanglement of the qubits following the same sequence, between each layer of the *ansatz*, as shown in Figure 4.6. In addition, the parameters of the quantum parametric gates in each qubit are sequentially connected. An example is shown in Figure 10.2(b), where each node represents a parameter in the *ansatz* with p layers, n qubits and pn parameters.

In the case of SPEDA, we have decided not to restrict the topology, since the BN structure learning algorithm embedded by SPEDA also learns the node types (Gaussian or KDE).

10.2.1 QAOA *ansatz* parameter tuning

Figure 10.3 shows a comparison of the computation time and expectation value minimization for different optimizers and the three EDAs: UMDA, EGNA with the fixed structure (EGNA_fs) and SPEDA, for the QAOA parameter tuning.

Figure 10.3 (top) shows how EGNA_fs reduces the computation time notably. Note that, when $p < 6$ the UMDA takes a notably larger computation time to converge compared to SPEDA and EGNA_fs, which is probably due to the number of iterations needed during runtime. SPEDA and EGNA_fs take a similar computation time up to $p < 5$. For larger p, EGNA_fs improves runtime.

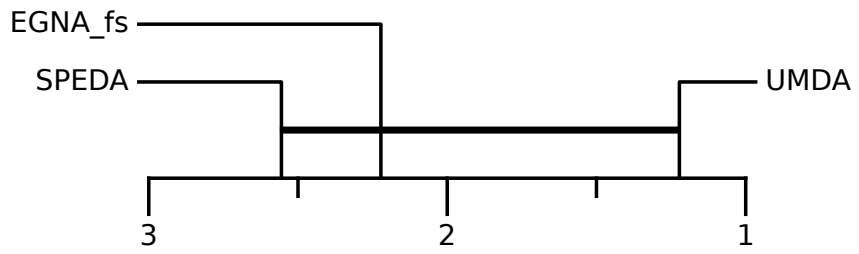


Figure 10.4: Critical difference diagram using Friedman tests to reject the null hypothesis of equal expected value, and a post-hoc analysis based on the Wilcoxon-Holm method. The horizontal black line connects the EDA variants that do not have a significant difference in the QAOA *ansatz* parameter tuning in terms of expectation value minimization using the three EDA variants.

No statistical significant differences have been found between the three EDA approaches in terms of expectation value minimization, as shown in the critical difference diagram [Demšar, 2006] in Figure 10.4. Comparing the EDA variants against the other algorithms, it is shown how the EDAs are the algorithms which achieve the smallest expectation values when $p < 5$. For more layers, the EDAs provide competitive results compared to their competitors, being only beaten by some gradient-based algorithms (CG, L_BFGS_B and SLSQP), see Fig 10.3 (bottom).

Analyzing the computation time, when $p > 5$, the EDA variants offer a computation time advantage compared to most of their competitors. Nevertheless, when $p < 6$, the computation time is slightly larger or equal than the rest of their competitors. It is worth noting that the computation time of CG is no longer competitive when $p > 3$.

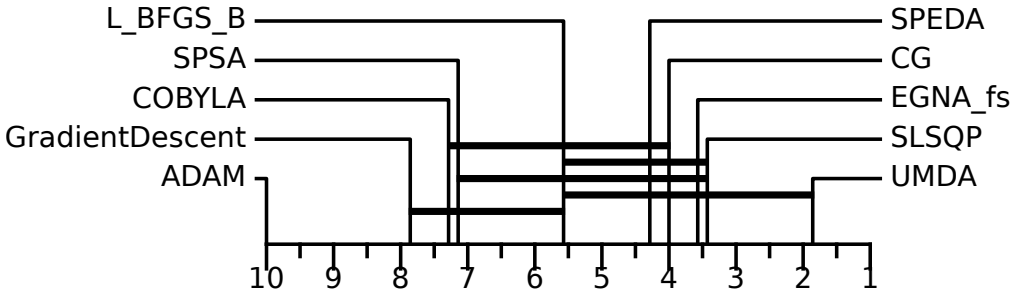


Figure 10.5: Critical difference diagram for the QAOA *ansatz* parameter tuning in terms of expectation value minimization comparing the different optimizers including the EDA variants.

Figure 10.5 shows a critical difference diagram where the statistical significant differences are identified in terms of expectation value minimization. The three EDA variants are grouped with other three gradient-based approaches (CG, L_BFGS_B and SLSQP) as the best optimizers for the QAOA *ansatz* parameter tuning for $p \in \{1, 2, \ldots, 8\}$, and UMDA improves the performance over the other gradient-free optimizers.

From this analysis we conjecture that EGNA_fs improves the expectation value minimization compared to its competitors, achieving a competitive computation time, when $p \leq 5$. When

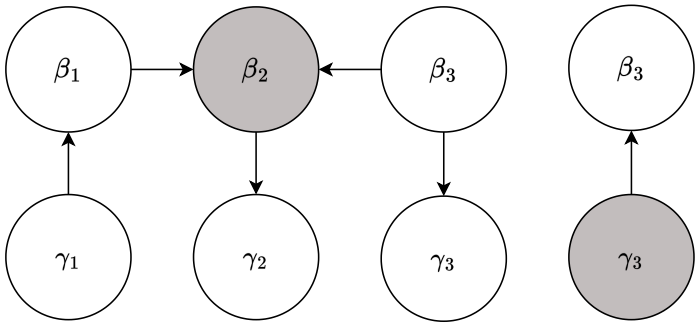


Figure 10.6: BN structure where the common arcs found in the BNs of the last iterations of 10 different SPEDA runs for the QAOA *ansatz* parameter tuning with $p = 4$ are represented. White and grey nodes represent the Gaussian and KDE nodes, respectively.

$p > 5$, EGNA_fs reaches similar results compared to its competitors but offering one of the best computation times.

One of the advantages of EDAs is the interpretability of the algorithm. Due to the use of BNs, it is possible to infer dependencies between variables, which may be of interest when analyzing the problem to be solved. It is expected that the EDA, in the last iterations of the runtime, will find the optimal structure that represents the landscape of the cost function. Figure 10.6 shows a BN structure presenting the common arcs found in the BNs of the last iterations of 10 different SPEDA runs for the QAOA *ansatz* parameter tuning with $p = 4$, where white and grey nodes represent the Gaussian and KDE nodes, respectively. Comparing the BN structure with the one fixed for the EGNA_fs approach in Figure 10.2, it can be observed the common arcs between nodes β_i and γ_i in each layer, although some arcs have been reversed, which verifies that the structure fixed for the EGNA_fs approach is consistent with the findings. Some spurious arcs have also been identified, such as $\beta_1 \to \beta_2$ and $\beta_3 \to \beta_2$.

10.2.2 VQE *ansatz* parameter tuning

Figure 10.7 shows a comparison of the computation time and expectation value minimization for different optimizers and the three EDAs: UMDA, EGNA with the fixed structure (EGNA_fs) and SPEDA for the VQE *ansatz* parameter tuning.

No significant differences were found between EGNA_fs and SPEDA; however, a significant improvement is found for the case of UMDA for $p \in \{1, \ldots, 9\}$, as shown in the critical difference diagram in Figure 10.8. Analyzing the computation time, it can be observed that UMDA, for all p, needs a larger runtime than the other EDA variants, and as p increases, a greater difference is found. For $p \geq 6$ the difference of computation time compared to EGNA_fs and SPEDA is noticeably larger, while the minimization of the expectation value no longer has a statistically significant advantage, as shown in Figure 10.9, where all the EDA variants are grouped together. From this analysis we conjecture that UMDA is a competitive optimizer for TwoLocal *ansatz* parameter tuning for low values of p. For larger values, EGNA_fs and SPEDA outperform UMDA, if a trade-off between computation time and expectation value minimization is desired. Moreover, it is worth mentioning the low

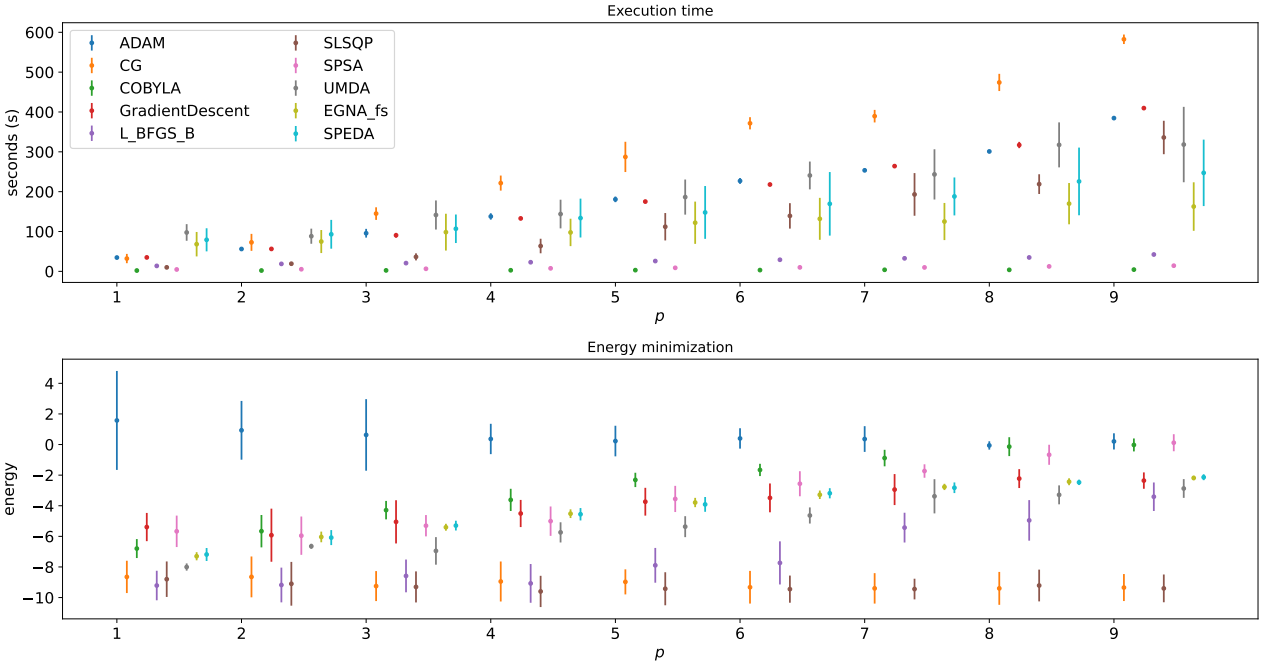


Figure 10.7: Comparison of the different EDA variants with the other state-of-the-art optimizers for the VQE TwoLocal *ansatz* parameter tuning. The top panel shows the mean computation time and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \ldots, 9\}$. The bottom panel shows the mean best expectation value achieved (Equation 4.9) and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \ldots, 9\}$.

standard deviation of the expectation value achieved with EGNA_fs and SPEDA compared to its competitors.

The computation time difference of UMDA compared to the other EDA variants has been identified in both the QAOA and VQE *ansatz* parameter tuning. Figure 10.10 shows the number of cost function evaluations of the three EDA variants, for different values of p in the VQE TwoLocal *ansatz* case. Note that each cost function evaluation involves an *ansatz* parameter configuration and measuring the quantum circuit N times. The figure shows how SPEDA is the EDA variant which needs fewer evaluations to converge to a solution that has no significant difference with EGNA_fs and UMDA, for $p \geq 6$. Note that UMDA is the algorithm which needs more evaluations for convergence, and the difference compared to EGNA_fs and SPEDA increases with p. Although the number of evaluations of SPEDA is lower than those of EGNA, the computation time has shown to be slightly higher in Figure 10.3 due to the probabilistic model complexity embedded by SPEDA and its structure learning, which was omitted in the case of EGNA_fs. SPEDA estimates some variables using KDE, exploring several areas of the search space in parallel [Soloviev et al., 2023a], so the cost function evaluations is likely to be reduced, which seems to be happening in this case resulting in a competitive computation time compared to EGNA_fs and UMDA.

Figure 10.11 shows a critical difference diagram to identify the significant differences in the results shown in Figure 10.7 for the expectation value minimization and $p \in \{1, \ldots, 9\}$. The

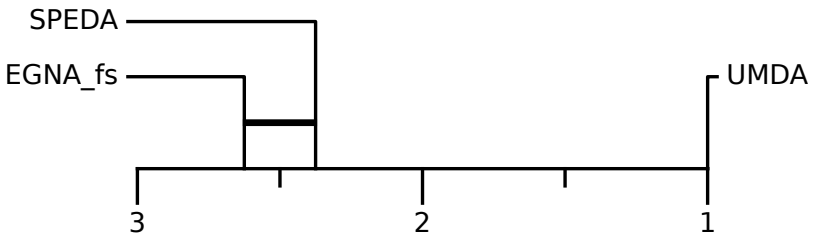


Figure 10.8: Critical difference diagram for the VQE TwoLocal *ansatz* parameter tuning in terms of expectation value minimization using three EDA variants.

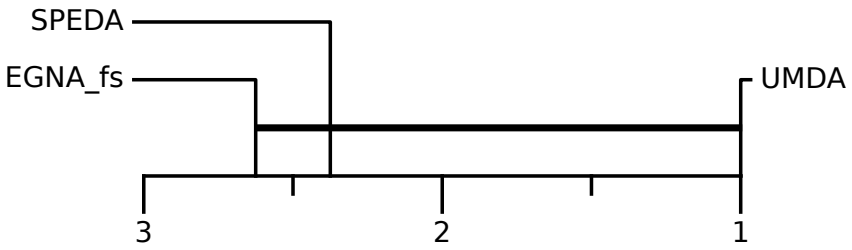


Figure 10.9: Critical difference diagram for the VQE TwoLocal *ansatz* parameter tuning ($p \geq 6$) in terms of expectation value minimization using the three EDA variants.

EDA variants are beaten by three gradient-based approaches: CG, L_BFGS_B and SLSQP, which also had a good performance in the case of QAOA. However, it is worth highlighting the computation time demand of CG for $p \geq 1$ and SLSQP for $p \geq 6$, which is improved by EGNA_fs and SPEDA for $p \geq 6$. For a large number of layers ($p \geq 8$) the expectation value minimization of L_BFGS_B is slightly better compared to the EDA variants, and its computation time is one of the best in the overall comparison. Despite the fact that the gradient-free optimizers achieve a low computation time compared to their competitors, the mean expectation value is always worse than that found by the different EDA variants.

From this analysis we conjecture that EGNA_fs and SPEDA are competitive optimizers for large number of layers ($p \geq 5$) if a trade-off between computation time and expectation value minimization is desired, where its principal competitor is L_BFGS_B. SPEDA is a better approach to minimize the resources demand, as it needs less quantum circuit measurements compared to EGNA_fs. For lower values ($p < 6$), UMDA improves the expectation values achieved by SPEDA and EGNA_fs, being one of the best optimizers in the overall comparison, although its computation time is worse in general.

10.2.3 Molecule simulation with parametric quantum noise

Simulating molecules behaviour in the area of quantum chemistry has gained a lot of attention in the last years due to its advantage compared to the classical computation [Peruzzo et al., 2014]. In this section, the VQE TwoLocal *ansatz* is used to simulate the hydrogen molecule (H_2), where the objective is to find the ground state of the Hamiltonian that defines the molecule. Furthermore, this simulation has been carried out considering different intensities of a simulated quantum noise channel. The depolarized noise [Nielsen and Chuang, 2002], parameterised by $\omega \in [0, 1]$ has been used, where $\omega = 0$ implies no quantum noise and $\omega = 1$

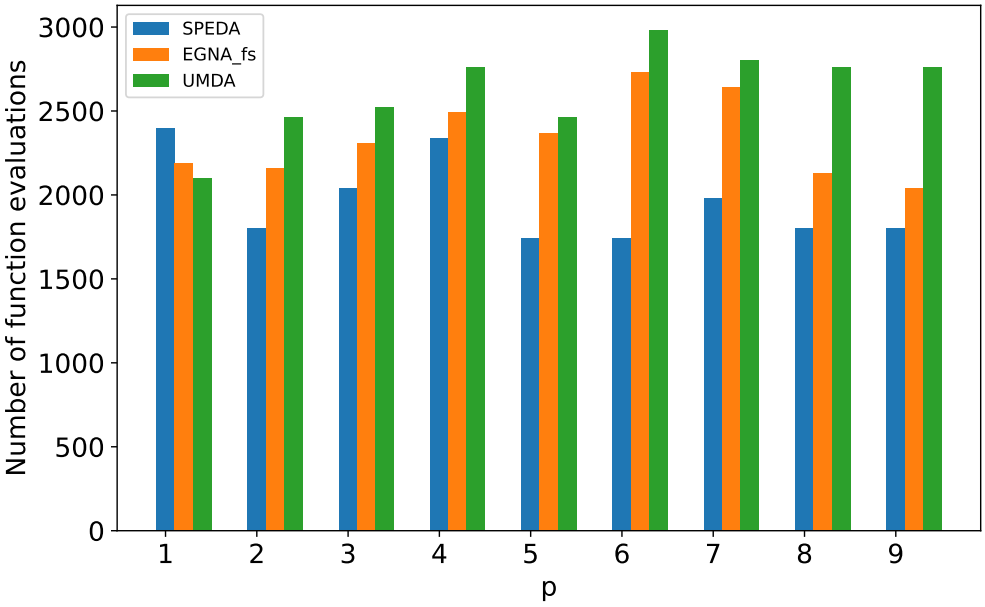


Figure 10.10: Comparison of the mean number of cost function evaluations for the VQE TwoLocal *ansatz* parameter tuning after executing each EDA variant 25 independent times. Each cost function evaluation involves a new *ansatz* parameter configuration and measuring the quantum circuit N times.

implies the maximum noise.

Table 10.1 shows the mean expectation value (Equation 4.9) achieved by the different optimizers for different values of ω. Note that the COBYLA and EDA variants are the optimizers which more resilience offer to quantum noise, being both able to outperform the results of their competitors. AQGD is the worst performing optimizer in general. Note that, for high noise intensities ($\omega \geq 0.7$), all the algorithms tend to converge to the same solutions, but for small ones ($\omega \to 0$), a greater difference is noted between the results of the EDA variants and the rest of competitors.

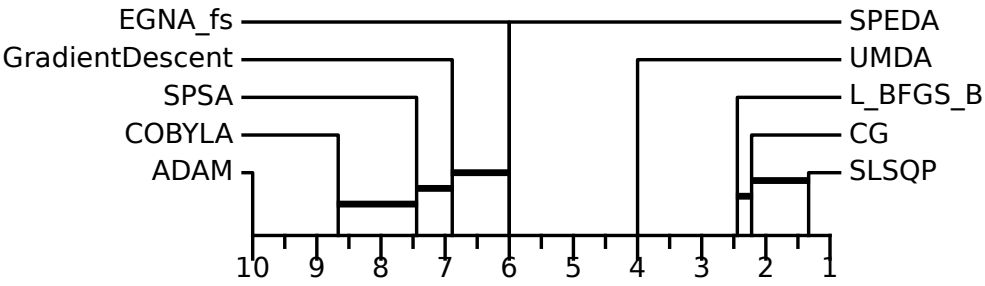


Figure 10.11: Critical difference diagram for the VQE TwoLocal *ansatz* parameter tuning in terms of expectation value minimization comparing the different optimizers including the EDA variants.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADAM	-1.12	-1.06	-1.09	-1.07	-1.09	-1.09	-1.08	-1.09	-1.09
AQGD	-1.43	-1.23	-1.16	-1.10	-1.09	-1.08	-1.08	-1.08	-1.08
CG	-1.03	-1.06	-1.10	-1.09	-1.08	-1.09	-1.09	-1.09	-1.09
COBYLA	-1.42	-1.27	-1.18	-1.13	-1.11	-1.09	-1.09	-1.09	-1.09
EGNA_fs	-1.44	-1.26	-1.17	-1.13	-1.11	-1.10	-1.09	-1.09	-1.09
GradientDescent	-1.22	-1.08	-1.08	-1.09	-1.09	-1.08	-1.09	-1.09	-1.09
L_BFGS_B	-1.11	-1.07	-1.08	-1.09	-1.09	-1.09	-1.08	-1.09	-1.09
SLSQP	-1.08	-1.06	-1.08	-1.07	-1.09	-1.09	-1.09	-1.09	-1.09
SPEDA	-1.45	-1.26	-1.18	-1.13	-1.11	-1.10	-1.09	-1.09	-1.09
SPSA	-1.42	-1.25	-1.17	-1.12	-1.10	-1.09	-1.09	-1.09	-1.09
UMDA	-1.43	-1.26	-1.17	-1.13	-1.11	-1.10	-1.09	-1.09	-1.09

Table 10.1: Mean expectation value (Equation 4.9) achieved by different optimizers for different values of $\omega \in [0, 1]$, where best values are highlighted in blue.

10.2.4 EDA hyper-parameter tuning

The results shown in Section 10.2.1 and Section 10.2.2 show that the higher the number of layers (p), the higher the expectation value $E(\cdot)$ is obtained being not able to reach the values found in the cases of lower number of layers. This factor is directly correlated to the population size (N) defined for the algorithm. In this section, we will comment on the relationship between the hyper-parameters p and N for the specific case of SPEDA for QAOA, since UMDA and EGNA have an equivalent behaviour in this aspect for both *ansatz*.

Figure 10.12 (left) shows that regardless of the number of layers (p) the expectation value is reduced by increasing the population size (N). Identifying the sweet spot between minimizing $E(\cdot)$ but also N, will lead to find the optimum value of N. The higher the N values, the more evaluations are needed, and thus, a higher computation time is required. Following a similar procedure as in the elbow method for clustering, we identify the most interesting population sizes for each p in Figure 10.12 (right). A linear dependency is observed between both parameters although this is an approximation.

10.3 Conclusions

In this chapter, a deep study on the use of different variants of EDAs for the VQA *ansatz* parameter tuning has been performed, where the algorithms have been compared to other state-of-the-art gradient-based and gradient-free alternatives widely used. The UMDA, EGNA with a fixed structure (EGNA_fs) and SPEDA algorithms were tested to tune the parameters of the QAOA and VQE *ansatz*.

In the case of QAOA, the three EDA alternatives offer similar solutions in terms of expectation value minimization, but EGNA_fs is the one which needs the shortest computation time for convergence. The results have been validated to test the null hypothesis of equal mean results versus different means among the algorithms.

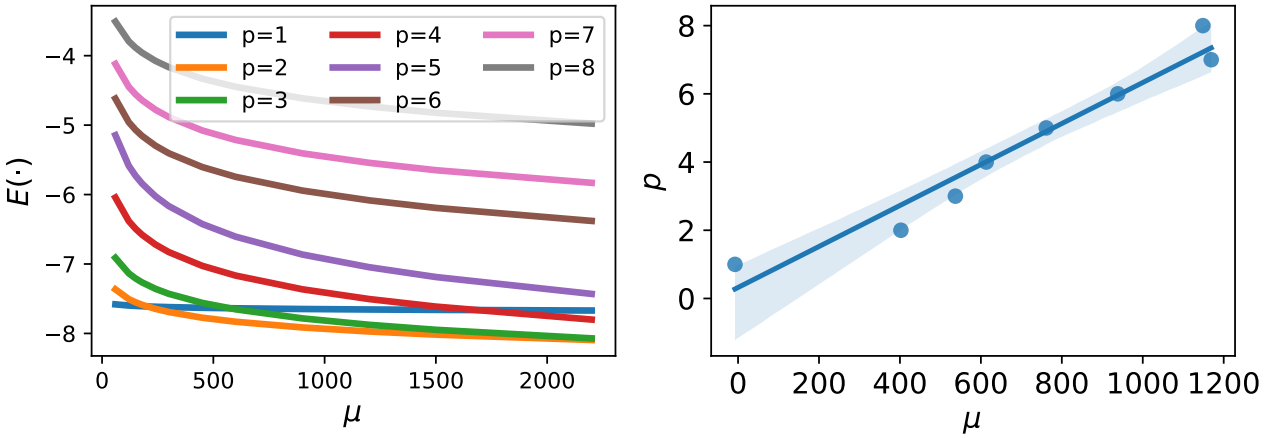


Figure 10.12: Left panel shows the mean expectation value tendency for different values of p and N. Right panel shows the linear dependency between the number of layers and the optimum values of N found. Both panels analyze the QAOA and SPEDA case.

In the case of VQE, UMDA offers a statistically significant advantage regardless of the number of layers, but requires a longer computational time. However, for large values of p, this advantage is not statistically significant, and SPEDA achieves a notable improvement in computation time compared to the other algorithms, offering competitive expectation value results and needing the fewest cost function evaluations, compared to EGNA_fs and UMDA.

While it is not the objective of this study to find the best EDA variant for the VQAs *ansatz* parameter tuning, we have found that all three EDA variants perform better than other gradient-free algorithms, and achieve competitive solutions with gradient-based ones. UMDA stands out for the quality of the solutions found. EGNA_fs is the fastest in terms of computation time, but SPEDA uses the fewest number of cost function evaluations. Moreover, the three EDA variants, together with COBYLA, have shown a better quantum noise resilience compared to the other competitors for the case of molecule simulation.

As future steps in this research line we propose the following:

- To study the performance of these EDA variants for further types of quantum noises.
- To study the performance of EDAs when Barren plateaus are present.
- To study the BN structure learned for larger number of parameters.
- To go one step forward by exploring not only the parameter space of the *ansatz* but also the circuit architecture (gates and number of parameters).

Chapter 11

Trainability Maximization for Quantum Architecture Search

11.1 Introduction

Chapter 10 proposes the use of EDAs to optimize the parameters of a given *ansatz*. In this chapter we propose optimizing one step forward, not only finding the optimal parameters of the *ansatz* but also its architecture using EDAs (QC in Figure 1.1). That is, the gates composition of the quantum parametric circuit. Similarly, in the area of classical machine learning there exists the neural networks architecture search (NAS) research line [Elsken et al., 2019], in which a similar problem is proposed for building neural networks and optimizing its weights together.

When choosing an *ansatz* for a problem and optimizing its parameters, we assume that the *ansatz* is expressive enough to converge to the ground state of our Hamiltonian. Finding the ideal *ansatz* for a given H but also the parameters $\boldsymbol{\theta}$ becomes a multi-level optimization problem [Mejía-de Dios et al., 2023] in which each proposed *ansatz* also involves a new optimization task regarding the parameters of the specific architecture. Some approaches are presented in the literature using heuristics, where most of them involve too many measurements, and therefore lead to an increase of the computational resources and time. This is crucial for the feasibility of the algorithm in NISQ devices as the number of available measurements is limited before the device is re-configured. Overcoming these limitations leads us to the quantum architecture search (QAS) research topic, where some authors have proposed different ideas.

The training/optimization of the variational parameters is known to be a non-trivial task for deep circuits, since we might face quite a few challenging trainability issues, e.g., BPs and traps [Anschuetz and Kiani, 2022]. BPs are typically described as vanishing gradients close to zero in the landscape, where the classical optimization becomes challenging, i.e., non-trainable or hard-to-train *ansatz*. Several works are found in the state of the art where this phenomenon is studied in order to analyze the trainability of the *ansatz* [Cerezo et al., 2021b, McClean et al., 2018]. However, computing these gradients involves the parameter optimization of the *ansatz*,

and thus increasing the number of quantum simulations, as we need to estimate the variance of the partial derivatives over the entire parameter space (exponential complexity). These tasks become more difficult with the number of qubits. Recently, Pérez-Salinas et al. [2023] have shown that the information content (IC) metric can reliably estimate the average (over the parameter space) norm of the gradient with a small number of evaluations of parameters of the *ansatz*.

In this chapter we propose a domain-agnostic approach based on EAs in which, given a set of *ansatzes*, for which a good performance is expected, we seek to find a new set of *ansatzes* similar to the initial one, but which are easier to train, and therefore are more likely to avoid the presence of BPs. The number of quantum simulations are drastically reduced by implementing a surrogate model which predicts the performance of the *ansatz*, and the IC is used to maximize the trainability of the proposed architectures avoiding the presence of BPs. Experimental results are shown in noisy environments for different problems. Thus, the main contributions of the chapter are:

- The use of surrogate models to rank the *ansatzes* proposed by the EDA without any measurements.
- The maximization of the trainability during the optimization process by using the IC.
- The use of multi-objective optimization to optimize the IC and the score provided by the surrogate model.

To the best of our knowledge this is the first work in which IC is optimized for quantum *ansatz* design, and we conjecture this approach can pave the way to bridging the gap towards an ideal training-free approach.

This chapter includes the developed methodology and results included in Soloviev et al. [2024a]. Implementation is based on EDAspy[1] Python package, and the experimental scripts and data are stored in a GitHub repository[2]. The dataset used for the *ansatz* comparison is published [Nakayama et al., 2023] and freely available in GitHub[3].

Chapter outline

The outline of this chapter is organized as follows. Section 11.2 reviews the QAS literature. The proposed methodology is presented in Section 11.3 and Section 11.4 shows some experimental results. Section 11.5 rounds the paper off with some further conclusions and future open research lines.

11.2 Related work

This section reviews some of the existing works regarding QAS in the literature.

Regarding reinforcement learning (RL), [Pirhooshyaran and Terlaky, 2021] uses a bi-level

[1] https://github.com/VicentePerezSoloviev/EDAspy
[2] https://github.com/VicentePerezSoloviev/QAS_EDA
[3] https://github.com/Qulacs-Osaka/VQE-generated-dataset

optimization process in which the agent proposes new architectures while a classical secondary optimizer tunes the parameters of the *ansatz*. In [Fösel et al., 2021], a RL approach is proposed with a different purpose: given an *ansatz*, return an optimized structure in terms of circuit depth and used gates. A RL approach is proposed [Ostaszewski et al., 2021b] where an agent systematically modifies the *ansatz* and achieves shallow circuits for chemical domains.

Regarding EAs, [Chivilikhin et al., 2020] proposes a bi-level genetic algorithm where a multi-objective approach is used to minimize the energy of the VQE while minimizing the number of CNOT gates, and the parameter optimization is performed by CMA-ES optimizer. In [Rattew et al., 2019] the authors use a genetic algorithm to optimize a weighted single-objective cost function combining the energy of the proposed *ansatz*, its depth, and number of two-qubit gates. Recently, GA4QCO framework [Sünkel et al., 2023] is proposed in which a single-objective optimization is performed by a genetic algorithm, and compared to random instances.

Regarding chemistry simulation, AdaptiveVQE [Grimsley et al., 2019] is a methodology that systematically grows an *ansatz* for chemical simulation; and RotoSelect and RotoSolve methods [Ostaszewski et al., 2021a] are two efficient methods for jointly optimizing *ansatz* structure and parameters.

Several works are found in the literature in which neural architecture search methodologies are applied to QAS. QuantumDARTS [Wu et al., 2023] is an adaptation of classical DARTS [Liu et al., 2018b] for neural network architecture search to QAS, in which two methods are proposed: one for whole architecture search, and another for promising sub-architectures. Another example is [Zhang et al., 2022] in which new architectures are sampled from a probabilistic model, and gradients between the best energies found are computed.

Additionally, SuperNet structure [Du et al., 2020], samples several architectures and its parameters are classically optimized. Based on the performance, the *ansatz* are ranked and a new architecture is constructed based on the knowledge gained from them. SuperNet has also been used to enhance VQAs on an 8-qubit superconducting quantum processor for classification tasks [Linghu et al., 2022].

11.3 Method

This section explains the proposed approach and describes each of the modules in the following subsections. Figure 11.1 summarizes the flowchart of the approach where the main steps of the proposed algorithm are stated.

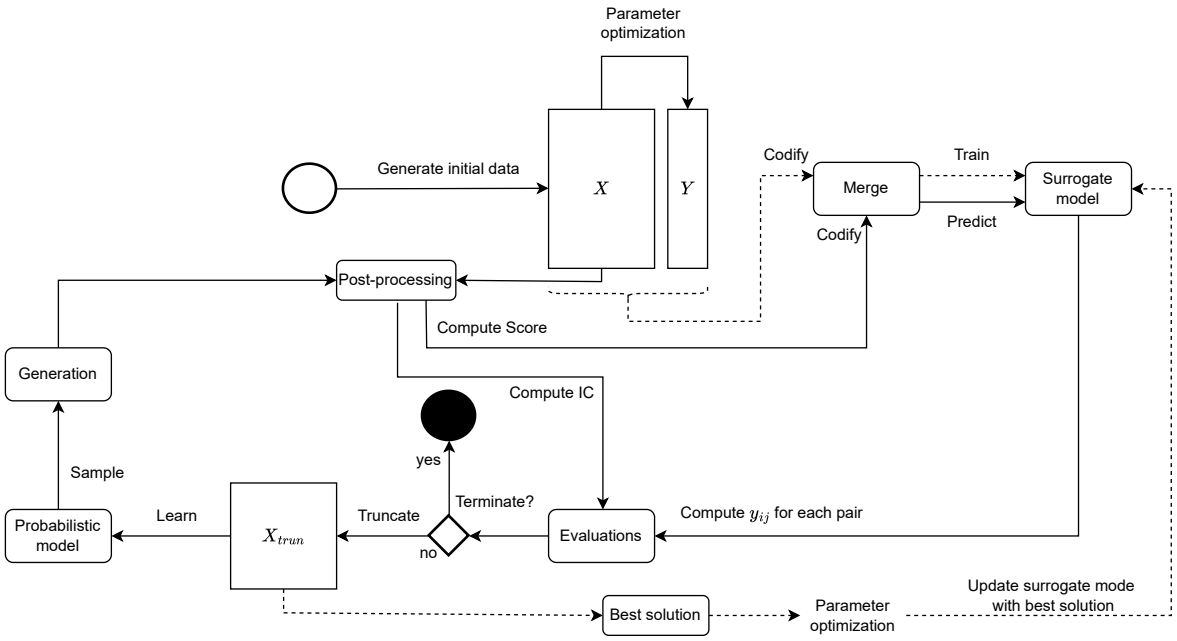


Figure 11.1: Flowchart of the proposed approach, starting from the white spot and finishing in the black spot one the convergence criteria is met. Dashed lines regard the train and update of the surrogate model.

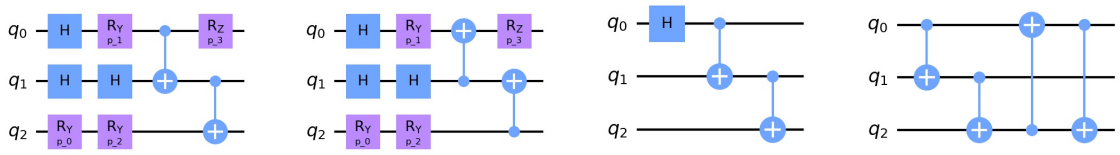
11.3.1 Codification

For an *ansatz* of n qubits and maximally depth m, we propose the following integer-valued matrix representation:

$$\boldsymbol{X} = \begin{bmatrix} X_{11} & \cdots & X_{1m} \\ \vdots & \ddots & \vdots \\ X_{n1} & \cdots & X_{nm} \end{bmatrix} \rightarrow [X_{11}, \cdots, X_{1m}, \cdots, X_{n1}, \cdots, X_{nm}], \tag{11.1}$$

where each entry $X_{ij} \in \{0, 1, \ldots, n_{gates}\}$ represents the choice of the quantum logic gate at position (i, j) of the matrix. Given a predetermined number of qubits n and maximal depth m, the architecture representation has a fixed dimension $d = nm$. This way, each column represents all the operators executed in parallel along the total depth, and each row represents a qubit.

Note that regarding two-qubit gates such as CNOT, applying a CNOT with the same control qubit, but different target qubits, are considered as different gates. This allows to restrict the evolutionary search according to hardware constraints by restricting the search space, although in this work an all-to-all connectivity is considered. In our case, $n_{gates} = (n-1) + 5$, as we consider the following universal operators: $\{Rx(\cdot), Ry(\cdot), Rz(\cdot), H, I\}$ and the CNOT gate with different target qubits. Note that CNOT(i, j) denotes that i and j are the control and target qubits, respectively. Therefore, we establish the following rules for the codification of X_{ij}:



(a) $\boldsymbol{A}_1$ (b) $\boldsymbol{A}_2$ (c) $\boldsymbol{A}_3$ (d) $\boldsymbol{A}_4$

Figure 11.2: Four examples of the ansatz codifications ($\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{A}_3$, $\boldsymbol{A}_4$) defined in Equation 11.2

- $0 \leq X_{ij} < n$ and $X_{ij} \neq i$ corresponds to the CNOT gate configurations from qubit i to the possible target qubits.
- $X_{ij} = i$ corresponds to $Ry(\cdot)$.
- $X_{ij} = n$ corresponds to $Rz(\cdot)$.
- $X_{ij} = n + 1$ corresponds to H.
- $X_{ij} = n + 2$ corresponds to I gate.
- $X_{ij} = n + 3$ corresponds to $Rx(\cdot)$.

The initial state of all the proposed architectures is set to the $|0\rangle$ state, i.e., $|00\cdots0\rangle^{\otimes n}$ state.

Figure 11.2 shows four examples where the following codifications are represented as *ansatzes*,

$$\boldsymbol{A}_1 = \begin{bmatrix} 4 & 0 & 1 & 3 \\ 4 & 4 & 5 & 2 \\ 2 & 2 & 5 & 5 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} 4 & 0 & 5 & 3 \\ 4 & 4 & 0 & 5 \\ 2 & 2 & 5 & 1 \end{bmatrix}, \boldsymbol{A}_3 = \begin{bmatrix} 4 & 1 & 5 & 5 \\ 5 & 5 & 2 & 5 \\ 5 & 5 & 5 & 5 \end{bmatrix}, \boldsymbol{A}_4 = \begin{bmatrix} 1 & 5 & 5 & 2 \\ 5 & 2 & 5 & 5 \\ 5 & 5 & 0 & 5 \end{bmatrix}, \quad (11.2)$$

where $n = 3$ and $m = 4$.

11.3.2 Probabilistic model

The joint probability distribution factorizes in a univariate EDA approach according to Equation 3.4, where $p(X_{ij})$ is the marginal probability distribution of variable X_{ij}. In this approach, $d = nm$, and $p(X_{ij})$ follows a multinomial distribution,

$$X_{ij} \sim \text{Mult}(n_m = \lfloor \alpha N \rfloor, k_m = (n_{gates} + 1)), \quad (11.3)$$

where n_m and k_m are the number of trials and mutually exclusive events that define the multinomial probability distribution, respectively.

Note that the marginal probabilities over the set of solutions are computed after the truncation process (Algorithm 4 Line 4), where the top $\lfloor \alpha N \rfloor$ solutions are selected according to the cost function to be optimized. The sampling process generates N new solutions as detailed in Algorithm 4, and duplicate *ansatz* are rejected in order to reduce redundancy. Each solution represents an *ansatz*, and the algorithm is expected to learn itself the best gates configuration during runtime.

(a) Original *ansatz*

(b) Post-processed *ansatz*

Figure 11.3: Post-processing of an *ansatz* where hard rules (Section 11.3.3) have been applied to the architecture represented in Equation 11.2 with $n = 3$ and $m = 4$.

11.3.3 Post-processing

In order to restrict the search space of the QAS problem, we establish a series of hard rules to remove redundancy and simplify the *ansatz* architectures proposed in the sampling process of the EDA.

- Two consecutive H gates are removed, as they are equivalent to an I gate.
- Consecutive application of $Rx(\cdot)$ gates, are simplified as one single $Rx(\cdot)$ gate, to remove redundancy.
- Consecutive application of $Ry(\cdot)$ gates, are simplified as one single $Ry(\cdot)$ gate, to remove redundancy.
- Consecutive application of $Rz(\cdot)$ gates, are simplified as one single $Rz(\cdot)$ gate, to remove redundancy.

Once the algorithm samples a new set of architectures (Algorithm 4 Line 6), the post-processing step is applied to each of them. Figure 11.3 shows an example of the application of these hard rules, where (i) in the second qubit, both consecutive H gates were suppressed, and (ii) in the third qubit the two $Ry(\cdot)$ gates are simplified as a single gate.

11.3.4 Surrogate model

A characteristic of traditional EDAs is that once the solutions of the same population are ranked according to $g(\cdot)$, no matter how much better a solution is compared to others, as all solutions included in the top $\lfloor \alpha N \rfloor$ will contribute equally to the probabilistic model learning [Larrañaga and Bielza, 2024] (see Algorithm 4, Line 4). The surrogate model used in this approach surrogates the minimal thing needed for the EDA, that is, the ranking of solutions (line 4 Algorithm 4). This is introduced by a metric Score(A) (inspired in [Shi et al., 2021]) which measures the quality of a solution A within the rest of solutions of the population,

$$\text{Score}(A) = \sum_{B \in \boldsymbol{X}} (h(A,B) + 1 - h(B,A)), \tag{11.4}$$

where the higher Score(A), the better the quality of A, and $h(A, B)$ compares *ansatz* A to *ansatz* B as,

$$h(A,B) = \begin{cases} 0, & \text{if} \quad P_B \geq P_A + \epsilon \\ 1, & \text{if} \quad P_A \geq P_B + \epsilon \\ 2, & \text{otherwise} \end{cases} \tag{11.5}$$

where P_A, and P_B are the minimum expectation values (Equation 4.9) found by a classical optimizer for architectures A and B, respectively and ϵ is a tolerance error configured by the user. Note that $h(A, B) = h(B, A) = 2$ means that two *ansatz* A and B are non comparable or very similar performance is expected.

Computing Score(A) involves $\lfloor \alpha N \rfloor - 1$ comparisons, and thus, this is clearly the main bottleneck of the task. In order to overcome this, we propose the use of support vector machines (SVMs) to approximate $h(A, B)$. We take the following input feature to the surrogate model:

$$\text{Flatten}(A + B, A - B) \tag{11.6}$$

where A and B are the two *ansatz* architectures to be compared, and the resultant vector size is $d = 2nm$. Thus, $h(A, B) \in \{0, 1, 2\}$ is approximated by $h'(\text{Flatten}(A, B)) \in \{0, 1, 2\}$ using SVM.

Several classification methods have been tested over some initial data randomly generated for different values of n, where SVM achieved better accuracy metrics. Results using cross-validation can be found in Appendix E.2.

The implementation has been obtained from LibSVM library [Chang and Lin, 2011].

The surrogate model is re-fitted after each iteration with the top 5 solutions in the ranking of the best solutions computed by the EDA (Section 11.3.5). Thus, in each iteration 5 classical parameter optimizations are carried out, and the number of parameter tuning processes executed during runtime is $N + 5t$, where t is the total number of iterations. Without the usage of the surrogate model approach, this number would have been $N(1 + t)$.

11.3.5 Evaluation

This approach aims to find the optimal *ansatz* for a given problem H in terms of trainability and expected energy. Here we define the following metrics to be computed for each proposed architecture.

First, IC (Equation 4.15) maximization has been proved to be able to avoid BP in the *ansatz* parameter tuning [Pérez-Salinas et al., 2023]. Those architectures with low associated IC are less trainable/optimizable, compared to those with high IC. Our approach maximizes this metric through the optimization process. Here, the IC of an ansatz A is denoted as,

$$\text{IC}(A) = \epsilon_M \sqrt{M}, \tag{11.7}$$

where ϵ_M is the ϵ associated to the norm of the gradient computed after a random walk over the parameters (Section 4.15), and M is the number of parameters of *ansatz* A.

Second, Score$(\cdot)$ (Equation 11.4) evaluates the quality of a solution compared to a subset of solutions. Our approach implements an elite approach, in which the best solution of generation G_i also appears in generation G_{i+1}. Then finding a different best solution in G_{i+1} will lead to a best global solution in the whole optimization process. Thus, Score$(\cdot)$ is also desired to be maximized.

Maximizing both metrics becomes a multi-objective optimization problem, in which the Pareto frontier between both objectives is explored. During the optimization process defined in Algorithm 4 and Figure 11.1, the truncation process ranks the solutions according to $g(\cdot)$, which is here defined as,

$$g(A) = \text{HV}((\text{Score}(A), \text{IC}(A)),\ \boldsymbol{r}), \tag{11.8}$$

where HV$(\cdot)$ is the hypervolume contribution [Beume et al., 2009] between the surrogate model output (Score(A)) and the information content computed (IC(A)), and $\boldsymbol{r}$ is the reference point. The $\lfloor \alpha N \rfloor$ best solutions in terms of HV$(\cdot)$ minimization are the ones that better approximate the Pareto frontier, and are the ones that promote to the next EDA iteration.

The reference point can be estimated based on the bounds of Score(A) and IC(A). In the former, the lower bound is set to zero (the worst solution within the population) and the upper bound to $2N$ (the best solution within the population). In the latter, the lower bound is set to zero (the least trainable scenario) and the upper bound to 2, based on previous experience. Then, Score$(A) \in \{0, 1, \ldots, 2N\}$ and IC$(A) \in [0, 2] \in \mathbb{R}$, so the reference point is set to $\boldsymbol{r} = (2N, 2)$.

Finally, the optimization problem is formalized as,

$$\begin{aligned} &\min_{\boldsymbol{X}} \quad g(\boldsymbol{X}) \\ &\text{subject to } \boldsymbol{X} \in \{0, 1, \ldots, n_{gates}\}, \end{aligned} \tag{11.9}$$

where $\boldsymbol{X}$ denotes a codified *ansatz* (Equation 11.1), and $\dot{g()}$ is defined at Equation 11.8.

11.4 Results

TThis section shows some numerical results on solving different Hamiltonians $H \in \{H_1, H_2, H_3, H_4\}$ (Appendix E.1), already studied in [Nakayama et al., 2023] for $n \in \{4, 8, 12\}$. The following sections compare the results found by the EDA approach with those presented in the dataset from [Nakayama et al., 2023]. In the original paper, the authors present several architectures which find similar state vectors in the search space of VQE *ansatz*, for each H_i. Henceforth, D_i^n denotes the set of architectures proposed in the dataset to solve the Hamiltonian H_i with n qubits.

Two experiments have been carried out in which, (i) the initial population of the EDA approach is initialized randomly to test if the algorithm is able to converge to similar solutions to those proposed in the dataset (Section 11.4.1), and (ii) the initial population is initialized from the *ansatzes* proposed in the dataset [Nakayama et al., 2023] to test if the algorithm is able to improve the given architectures (Section 11.4.2).

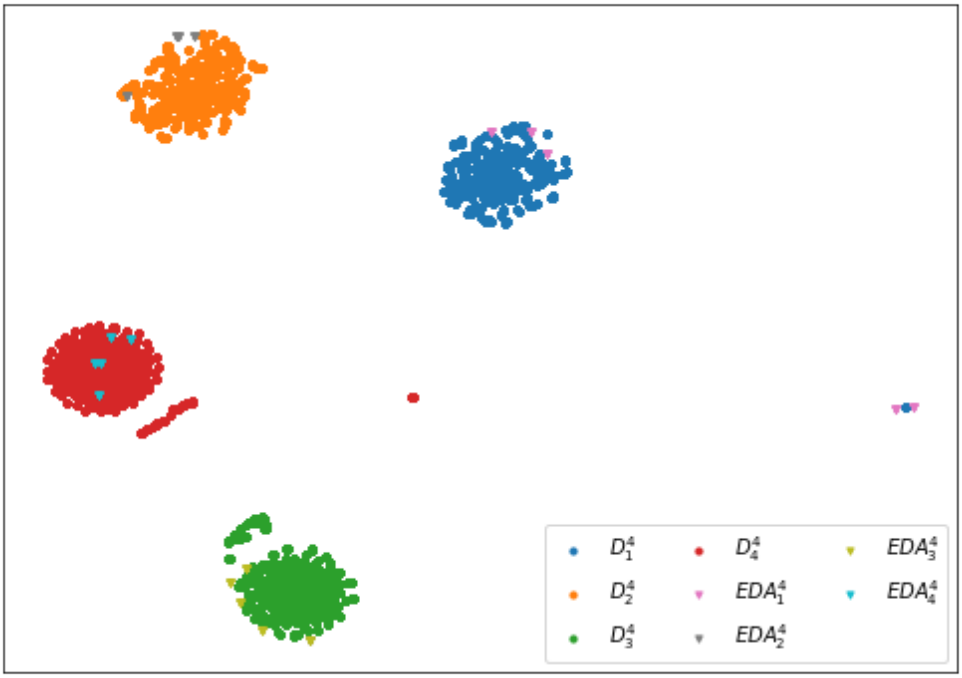


Figure 11.4: Visualization of the *ansatzes* found in the dataset (D_i^n) using t-SNE [Van der Maaten and Hinton, 2008] , which are colored depending on the Hamiltonian to be solved (H_i, where $i \in \{1, 2, 3, 4\}$). Additionally, the best architectures found by the EDA approach (EDA_i^n) are represented using different colored and shaped points. Note that EDA_i^n regards the solutions found for Hamiltonian H_i. All the results shown correspond to $n = 4$.

The size of the population, and maximum number of iterations of the EDA have been set to $N = 150$ and $t = 50$, respectively, for all the experiments. Regarding the quantum circuit simulation, we simulate the measurement noise.

11.4.1 Random initialization

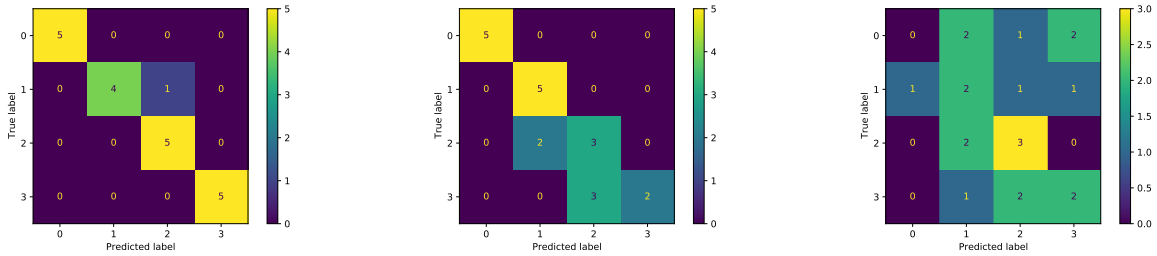
To randomly generate the initial population (G_0), a predefined probabilistic model is set to the algorithm, from which the set of solutions are sampled. Thus, some of the outcomes for each variable can be restricted, or boosted, decreasing or increasing the associated probabilities, respectively, as demanded by the user.

In this experiment, initially, all the possible outcomes have been set to equal probability for all the variables:

$$p(X_i = j) = \frac{1}{n_{gates} + 1}, \tag{11.10}$$

for all $i = 1, \ldots, d$ and $j = 0, 1, \ldots, n_{gates}$.

The initial population samples a set of N solutions, according to Equation 11.10. Each sample corresponds to a different architecture following the codification in Equation 11.1 and is post-processed (Section 11.3.3). The expectation value (Equation 4.9) of each architecture is



(a) $n = 4$ (b) $n = 8$ (c) $n = 12$

Figure 11.5: Confusion matrices for $n \in \{4, 8, 12\}$.

computed, where its parameters are classically optimized using an external optimizer. In this experiment we use COBYLA optimizer, as it has been shown to achieve good results in terms of CPU time and energy minimization [Powell, 1998]. Considering the set of solutions and associated expectation values, a surrogate model is trained (Section 11.3.4) and each solution is evaluated (Section 11.3.5).

The original dataset [Nakayama et al., 2023] proposes using dimensionality reduction to demonstrate that the minimal energy states achieved within D_i^n are very similar. Figure 11.4 shows the dimensional reduction using t-SNE [Van der Maaten and Hinton, 2008] for the Hamiltonians approached, represented as clusters in two dimensions. The solutions found by the EDA approach (EDA_i^n, where i denotes the index of the faced Hamiltonian and n the number of qubits) are also represented by stars and different colors. Note that our approach is able to reach very similar solutions to the ones presented in the dataset.

In the following analysis the fidelity of the lowest energy state found by the EDA approach is compared to those obtained by the *ansatzes* provided in the dataset for different problems $\{H_1, H_2, H_3, H_4\}$ and number of qubits (n), that is, by D_i^n.

The distance from each proposed *ansatz* (A) in EDA_i^n to each cluster of architectures D_i^n is computed by the arithmetic mean distance to each of the *ansatzes* belonging to D_i^n as,

$$\text{dist}(A, D_i^n) = \frac{1}{|D_i^n|}\left(\sum_{B \in D_i^n} 1 - F(|\Psi_A\rangle, |\Psi_B\rangle)\right), \tag{11.11}$$

where D_i^n is the subset of *ansatzes* (with size $|D_i|$) in the dataset proposed to solve H_i with n qubits and meet $m \pm \sqrt{m}$ restriction, $F(\cdot)$ is the fidelity between two quantum states, and $|\Psi_A\rangle$ and $|\Psi_B\rangle$ are the lowest energy states achieved by *ansatzes* A and B, respectively, after classical parameter optimization.

Table 11.1 shows the p-values computed using the ANOVA test[4] to reject the null hypothesis of equal means between each *ansatz* in EDA_i^n and the different clusters D_i^n, where highlighted results are rejected. Appendix E.3 details the distance computations statistically analyzed in this table. An increasing number of non-rejected hypotheses is observed for increasing number of qubits (n), which suggests that the EDA is proposing architectures much different to the ones available at the dataset for $n = 12$. Increasing the number of qubits (n) also

[4]All the data used for the ANOVA tests fit Gaussian distributions.

H_i	$n = 4$	$n = 8$	$n = 12$
H_1	**3.0e-34**	**3.0e-2**	6.0e-1
H_2	**1.3e-4**	**1.1e-2**	1.5e-1
H_3	**1.0e-15**	3.0e-1	1.1e-1
H_4	**2.0e-8**	**5.1e-2**	2.1e-1

Table 11.1: ANOVA one-way test to reject the null hypothesis of equal means between the mean distances (Equation 11.11), from the proposed by [Nakayama et al., 2023] *ansatzes* found by EDAs and $\{D_1^n, D_2^n, D_3^n, D_4^n\}$ proposed for $\{H_1, H_2, H_3, H_4\}$, respectively. A threshold of 5e-2 has been set to reject the null hypothesis, highlighting in bold those results below this value.

involves increasing the number of variables of the EDA optimizer. According to the results found, the population size set is not enough to generate a large number of samples which covers the increasing cardinality of the problem. Also, larger number of qubits should also involve a larger *ansatz* depth, so m should also be increased to allow more expressive quantum circuits. This suggests that the chosen configuration is valid to problems up to $n < 8$. For bigger instances, a different configuration of the hyper-parameters m and N should be chosen, although this would involve a drastic increase of the CPU time.

Assuming that a truly classified *ansatz* (A) is the case in which the closest cluster D_i^n represents H_i, and $A \in \text{EDA}_i^n$ was optimized for Hamiltonian H_i as well, Figure 11.5 shows the confusion matrices. The percentage of correctly classified *ansatzes* is 95%, 75% and 35% for $n = 4, 8, 12$, respectively, where a decreasing tendency is observed for increasing n; however, for $n = 12$ the EDA was not able to found any statistical significant result.

Figure 11.6 shows the IC convergence plot during the optimization process of the EDA approach. The associated shade shows a mean aggregation of the optimization processes regarding different $\{H_1, H_2, H_3, H_4\}$, where a maximizing monotonic tendency is observed. Regardless of the results encountered, the three scenarios show that the algorithm has converged. Note that, the mean IC found by the optimizer denotes an exponential decay with the number of qubits (n), as expected according to [Cerezo et al., 2021b, Pérez-Salinas et al., 2023].

Because Score(A) returns a metric comparing *ansatz* A with the rest of the architectures within the population to which A belongs, the trend throughout the optimization process is not an interesting fact to analyze.

Appendix E.4 shows the Pareto frontier approximation (non-dominated solutions highlighted as orange spots) for each H_i we are facing (in columns) and different values of n (in rows). It is observed how both objectives are conflicting, and maximizing one of the objectives worsens the second, and vice-versa. Thus, a trade-off between both objectives through the Pareto frontier approximation is desired. Note that the scale of the Y-axis (IC) is different for different number of qubits, as explained before.

Considering the best solutions found by the EDA, i.e., those that better approximate the Pareto frontier, we now compare the characteristics of the *ansatzes* proposals with those

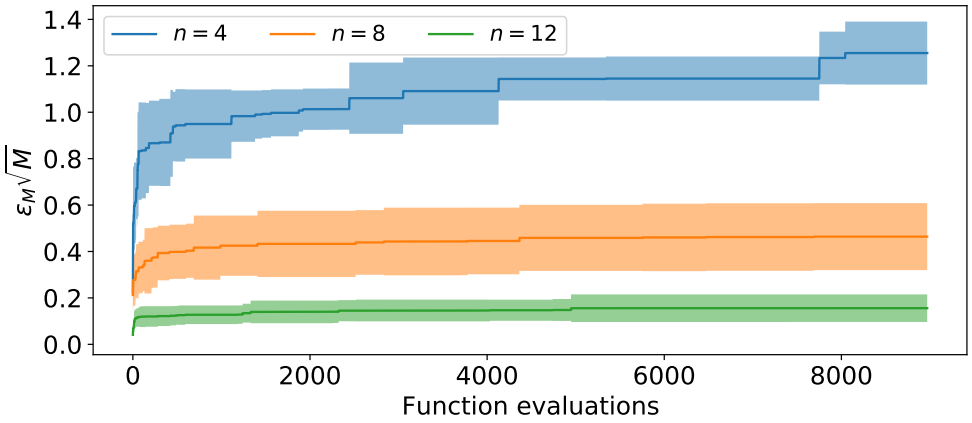


Figure 11.6: Mean and standard deviation of IC maximization aggregating the optimization process of different H_i for different numbers of qubits (n).

available in the dataset [Nakayama et al., 2023] with depth in the range $m \pm \sqrt{m}$ (for a fair comparison and ensure a minimum number of instances from the original dataset). A drastic increase in the number of certain quantum gates might improve the performance of the *ansatz*, however, this may lead to a poor trainability. Thus, the ratio among the gates set used, and the number of gates is further analyzed.

Figure 11.7 shows the ratio of the different available universal gates in the set of initial randomly generated data (G_0), the solutions found by EDA approach (EDA_i^n) and the best solutions from the original dataset (D_i^n), for different values of n. A strong correlation is observed between the initial data and the proposed solutions, independently of n, where the EDA_i^n has a slightly higher ratio of CNOT gates compared to G_0. However, comparing to D_i^n, our proposals achieve a much lower ratio of parametric gates, compensating it with superposition and two-qubit gates. Although the ratios for D_i^n seem to remain constant along n, our approach increases the number of CNOT gates with n.

Figure 11.8 plots the number of parameters as a function of n, in the set of initial randomly generated data (G_0), the solutions found by the EDA approach (EDA_i^n) and the original dataset (D_i^n). Although the number of gates increases linearly in the three cases, comparing the slopes found in the linear approximations of the three cases, the green function (D_i^n) denotes a coefficient approximately 6 times bigger than the other two functions. We show that our EDA is able to learn that a bigger number of parameters is needed, however, it does not increase this number drastically, as it is able to converge to simpler *ansatz*. Shallower *ansatzes* (low values in the Y-axis) are more convenient to be executed in real quantum devices due to quantum coherence and other issues of the NISQ devices.

In this experiment we tested whether our approach initialized from a random set of *ansatzes* is able to converge and find similar solutions to the ones proposed in the dataset, assumed to be optimal. Figure 11.4 and Table 11.1 show that our algorithm finds solutions with similar state fidelity as the ones in the dataset.

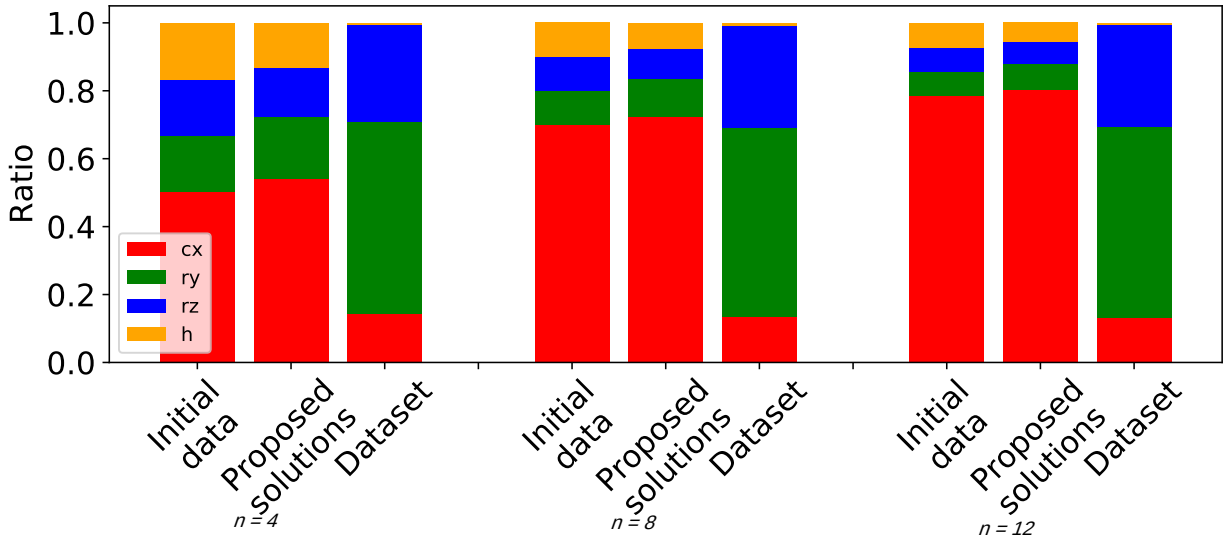


Figure 11.7: Ratio of $\{CNOT, RY, RZ, H\}$ gates in the *ansatz* design of the randomly generated initial data (G_0), best EDA solutions found (EDA_i^n), and dataset (D_i^n) [Nakayama et al., 2023], for $n \in \{4, 8, 12\}$, respectively.

11.4.2 Initialization with the dataset

The previous results have shown that the EDA approach is able to provide trainable and well performing architectures. In this section we initialize the EDA optimizer from the *ansatzes* provided in the dataset (D_i^n) to test whether it is able to converge to better solutions. Thus, the EDA execution used to face the Hamiltonian H_i will be initialized using $G_0 = D_i^n$. In this case, D_i^n will consist of all those architectures that meet the depth constraint imposed by the EDA. Note that, in case an architecture has a depth smaller than that imposed, the coding in binary (Equation 11.1) would be equivalent to fill with identity gates (I) until the desired depth is reached.

The purpose of this experiment is that, given a set of *ansatzes*, which are known to have good performance, we try to improve their trainability while maintaining a similar behavior. In order to compare the results found by the EDA, the energy (Equation 4.9) using a second level classical optimizer and the IC (Equation 4.15) are computed for all the *ansatzes* in all D_i^n. Results are shown in Table E.5.

Figure E.2 (Appendix) shows the Pareto frontier approximations for each H_i we are facing and different numbers of n. Note that, with increasing number of qubits, the conflict between both objectives becomes more drastic. However, the EDA approach is able to identify the promising solutions in the Pareto frontier. Note that the initial generation $G_0 = D_i^n$ has been also represented to establish a reference in terms of IC. However, Score(A) for the first generation should not be taken into account, as D_i^n represents similar minimal energy state vectors (Figure 11.4), and thus, are not comparable.

Table E.6 (Appendix) shows the best E and IC found by the EDA approach where COBYLA optimizer is used, for the *ansatz* parameter optimization. Note that the solutions shown in

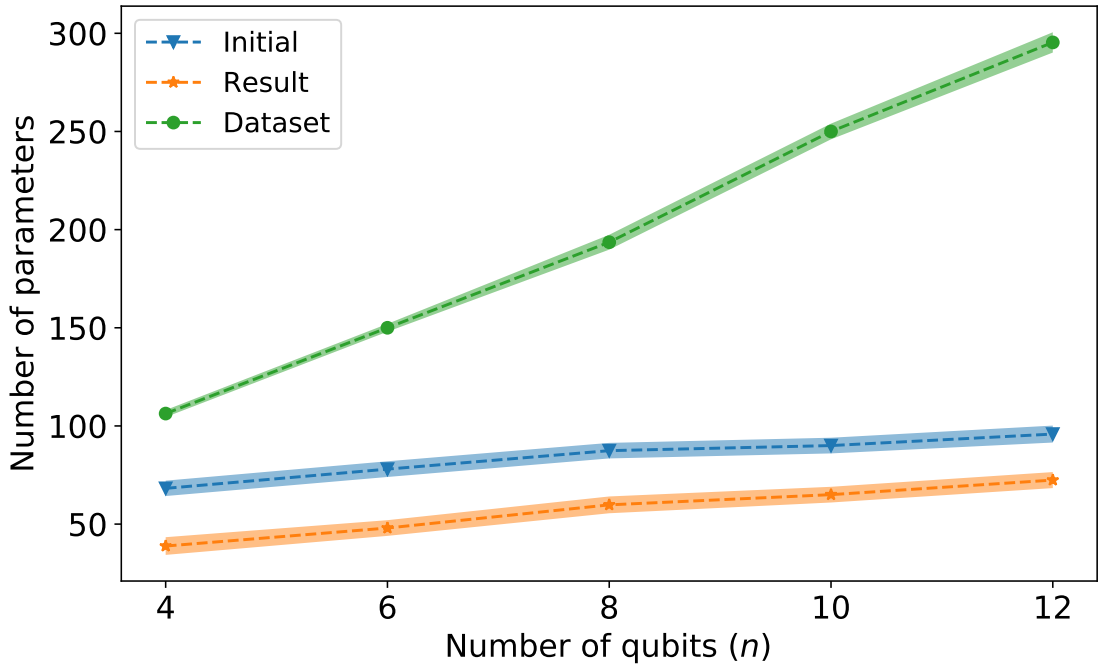


Figure 11.8: Mean and standard deviation of the number of parameters (Y-axis) as a function of the number of qubits (X-axis), in the *ansatzes* found in the randomly generated initial data (G_0), best EDA solutions found (EDA_i^n), and dataset (D_i^n) [Nakayama et al., 2023]. Note that the values for $n = 6, 10$ have been approximated through a linear regression.

the tables are the ones that maximize HV in the Pareto frontier approximation, that is, a trade-off between both objectives in the non-dominated solutions set is found. Although in this case it is important to show the solution that optimizes the HV, it is possible to analyze each of the non-dominated solutions from the Pareto front in order to maximize any of the two metrics.

Regarding the results shown in Table E.6, it is observed a good performance in terms of expectation value minimization for $n = 4$. Moreover, the IC achieved is noticeable better, which also happens in the case of $n = 8$. However, the expectation value obtained for H_3 and H_4 for $n = 8$ is worse than that described in the original dataset, which suggests that the EDA approach is not able to improve the metrics in Table E.5.

In this experiment we tested whether our approach is able to improve the quality of the *ansatz* provided in the dataset, from which the EDA is initialized. Our results show that the EDA approach is able to improve them in some of the cases, and suggest that a hyper-parameter tuning should be carried out for increasing number of qubits.

11.5 Conclusions

In this chapter we present a novel method for architecture search, in which the complexity of the multi-level optimization problem has been drastically reduced by using surrogate

modelling. The EDA approach optimizes the energy estimated by the surrogate modelling by performing comparisons by pairs, and reduces the possibility of barren plateaus issues.

The experimental results showcase two different situations for optimizing different Hamiltonians: (i) the EDA is initialized from a random subset of solutions, and (ii) the EDA is initialized from the best solutions presented in the dataset. In the former case, the results show that the optimizer is able to converge to the same solutions presented in the dataset when the number of qubits is lower than $n = 8$, and the hyper-parameters should be tuned for higher values of n. In the latter case, the EDA is able to improve the state of the art in some of the cases. Our approach is able to find solutions that keep a good performance regarding energy minimization, but also improve the trainability of the *ansatzes* encountered.

The following research lines are proposed as future work:

- The numerical results analyzed suggest that the performance of our approach worsens with the number of qubits, unless the population size (N) and the number of iterations (t) are increased. However, in order to implement a useful approach for NISQ and fault tolerant devices, the algorithm runtime for the optimization process is limited, in contrast to neural network architecture search, where the coherence of the devices does not change during time. Future work in this field would include the scalability of the algorithm to higher number of qubits (n).
- The EDA internally uses HV for ranking the architectures to be selected. Although the IC upper bound has been set based on previous experience, future work would include a dynamic definition of the reference point for the HV computation, during runtime.
- Given that this research is at an early stage, our primary focus is on showing underpinnings and initial feasibility rather than conducting exhaustive empirical comparisons with state-of-the-art methods. Comprehensive benchmarking and detailed empirical evaluations are planned for future studies.

Chapter 12

`EDAspy`: An Extensible `Python` Package for EDAs

12.1 Introduction

During the development of this thesis, a neediness has been detected to establish a reference library for the implementation of different versions of EDAs, as very few implementations have been found publicly available. This was also commented and agreed by some attendees at the EDAs Dagsthul meeting [Ceberio et al., 2022a]. All the methodologies used during this thesis have been included into `EDAspy` open-access library, and new functionalities have also been added.

In this chapter we present a `python` package in which several EDA implementations are efficiently designed. The different optimizers are easily called and can be tuned in a user friendly mode. Each EDA variant is built using different available modules, which can be customly selected to build a new implementation. These variants can be easily extended and interoperate with new components.

This chapter includes the materials presented in Soloviev et al. [2024b]. The implemented methodology is available in a GitHub repository[1] and available to be downloaded from Python repositories[2]. The documentation is available at ReadTheDocs platform[3].

Chapter outline

The chapter outline is organized as follows. Section 12.2 introduces the general organization of the library. Section 12.3 reviews other libraries in which similar approaches are implemented. Section 12.4 shows a CPU time comparison for continuous domain optimization. Section 12.5 explains several examples available at the library documentation. Section 12.6 rounds the chapter off with some further conclusions and future work.

[1] https://github.com/VicentePerezSoloviev/EDAspy
[2] https://pypi.org/project/EDAspy/
[3] https://edaspy.readthedocs.io/en/latest/

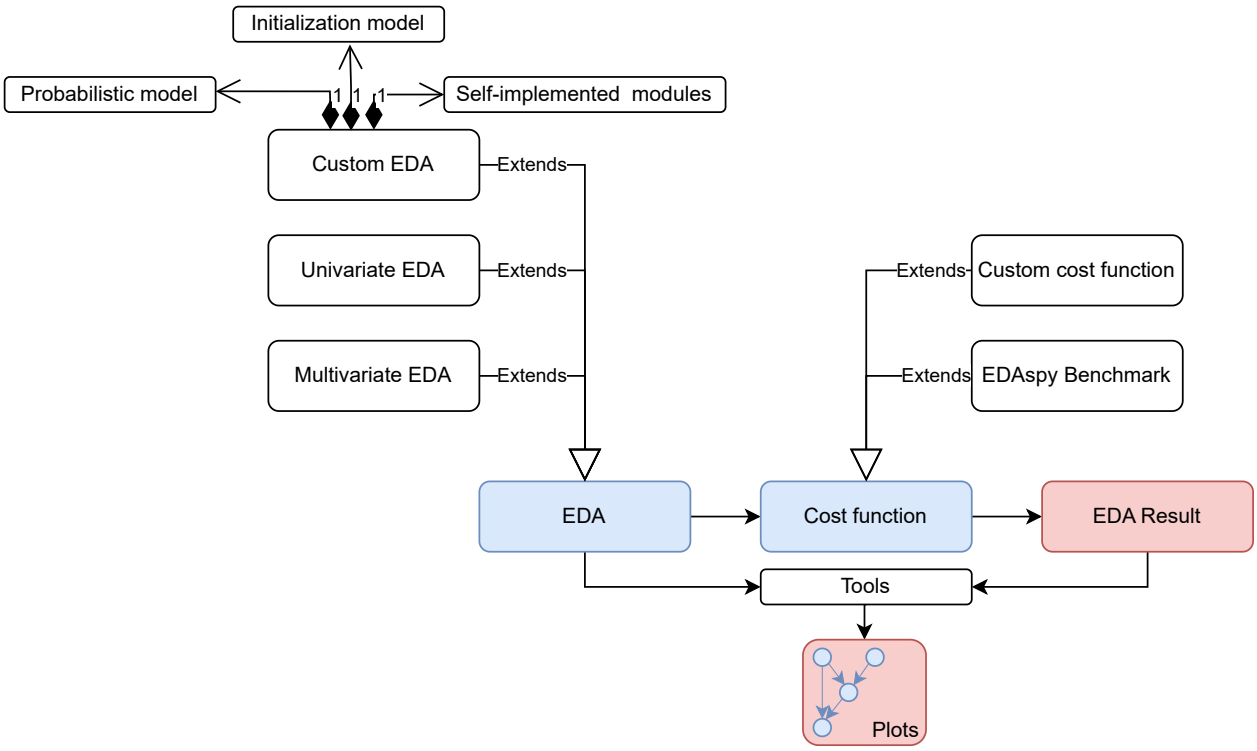


Figure 12.1: High order organization of the EDAspy library.

12.2 Software framework

Figure 12.1 represents the high order representation of the previously mentioned modules in `EDAspy`. In general, an EDA implementation is applied to a cost function to be minimized, and some results are found. There are several EDA implementations available in the library organized in univariate and multivariate modules, but it is also possible to build a customizable implementation by integrating the already available components with other modules (optionally) in the EDA object. Regarding the cost function, there are several benchmarks implemented. In addition, a custom cost function can be used. Once the optimizer has converged, several information and plots can be extracted from the execution.

Moreover, although the library has been built modular in order to allow the integration with new custom implementations, the EDA optimizer can be easily extended and built from scratch by the user without using *Custom EDA* module facilities.

`EDAspy` is organized in different modules:

- **Benchmarks**. Different test functions for benchmarking and comparing the different optimizers are included. Toy discrete functions such as OneMax [Krejca and Witt, 2017] and benchmark suites such as IEEE CEC 2014 [Liang et al., 2013] are included.
- **Univariate**. The following univariate approaches in which no dependencies between variables are considered: univariate marginal distribution algorithm (UMDA) for (i) binary [Mühlenbein and Paass, 1996] (UMDA$_B$), (ii) categorical (UMDA$_D$), and (iii)

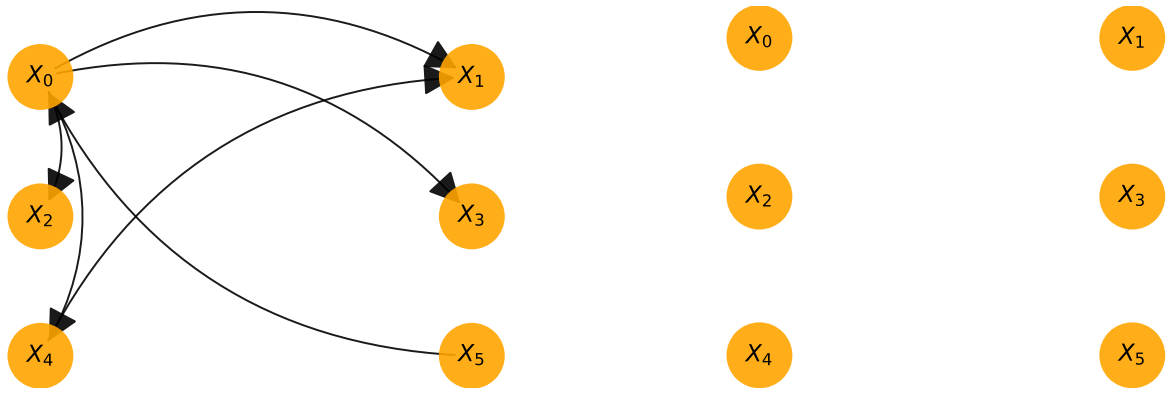
continuous optimization [Mühlenbein et al., 1996] (UMDA$_C$); (iv) kernel EDA [Luo and Qian, 2009] (u_KEDA); and (v) population-based incremental learning algorithm [Baluja, 1994] (PBIL).

- **Multivariate**. The following multivariate approaches in which dependencies between variables are considered: (i) estimation of Bayesian network algorithm [Larrañaga and Lozano, 2001] (EBNA), (ii) estimation of multivariate normal algorithm [Larrañaga and Lozano, 2001] (EMNA), (iii) estimation of Gaussian network algorithm [Larrañaga et al., 2000] (EGNA), (iv) semiparametric EDA [Soloviev et al., 2023a] (SPEDA), and (v) multivariate kernel density EDA [Soloviev et al., 2023a] (m_KEDA), (vi) Bayesian optimization algorithm (BOA) [Pelikan et al., 1999] in which a discrete BN, a multivariate Gaussian distribution, a Gaussian BN, a semiparametric BN, a kernel density estimated BN, and a discrete BN are iteratively learned, respectively.
- **Custom**: this module includes the different components to build a custom EDA variant and is divided into probabilistic and initialization models.
  - **Probabilistic model**. The following components are implemented for learning and sampling. Regarding univariate probabilistic models, (i) binary, (ii) discrete, (iii) Gaussian, and (iv) KDE models are considered. Regarding Bayesian networks, (v) Gaussian, (vi) semiparametric, (vii) KDE, and (viii) discrete models are available. Other models include (ix) multivariate Gaussian.
  - **Initialization model**. Uniform sampling meeting landscape user defined bounds, Latin hypercube sampling [McKay et al., 2000] and initialization from a given dataset are available to build the first population of the EDA.
  - **Self-implemented modules**. This includes modules implemented by users that can be integrated into the library.
- **Plotting tools**. The tools for graphically representing the probabilistic model embedded by the EDA are included in this module. Figure 12.2 shows an example of two different probabilistic models. Panel (a) represents a Gaussian BN, in which dependencies between variables are considered, while panel (b) represent a univariate model, in which no dependencies are considered.

Regarding the multivariate EDA implementations, some of the probabilistic models are interfaced to `PyBNesian` library [Atienza et al., 2022a], which uses `C++` to speed up the back-end computations. All the algebraic computations in `EDAspy` are computed using `numpy` library [Harris et al., 2020], employing C to speed up the back-end computations. Moreover, the parallelization of the optimizer is available by using `multiprocessing` library [McKerns et al., 2012, McKerns and Aivazis, 2010], and can be optionally activated in all the EDA implementations.

12.3 Related work

Although there are several libraries in which different evolutionary algorithms are available, to the best of our knowledge we have not found comparable published libraries with different



(a) Gaussian BN structure

(b) Univariate Gaussian structure

Figure 12.2: Probabilistic models graphical representations.

EDA implementations in `python`. However, here we list some libraries in which some EDA implementations are available.

- `mateda` [Santana et al., 2010] is a `matlab` library which allows building multivariate EDAs based on undirected probabilistic models and Bayesian networks. The purpose of the library is different from `EDAspy`. It offers a framework to build a multivariate EDA algorithm by modules, in which different components can be integrated. `mateda` implements categorical and Gaussian Bayesian networks, multivariate Gaussian distributions, Markov networks and mixtures of Gaussian distributions as probabilistic models. However, semiparametric and KDE Bayesian networks are missed, and the implementations for univariate approaches are omitted. Moreover, the last released version of `mateda` was in 2020.
- `inspyred` [Tonda, 2020] is a `python` library which implements general evolutionary algorithms such as genetic algorithms, evolutionary strategies, differential evolution and multi-objective genetic algorithms, among others. UMDA$_C$ is the only EDA implemented.
- `LEAP` [Coletti et al., 2020] is a `python` library built for evolutionary computation and incorporates useful visualization modules. Regarding EDAs, the population-based incremental learning algorithm (PBIL) [Baluja, 1994] and Bayesian optimization algorithm (BOA) [Pelikan et al., 1999] are expected to be available in future releases.

Table 12.1 summarizes the main differences between the listed libraries. Regarding univariate approaches, `inspyred` implements UMDA$_C$ and `LEAP` plans to integrate PBIL approache in the near future, compared to the five implemented variants in `EDAspy`. Regarding multivariate approaches, `LEAP` will incorporate BOA approach, which is also implemented in `EDAspy`. The most competitive library is `mateda`, which overlaps with some of the implemented multivariate approaches. It also allows for building a custom EDA version with some additional probabilistic models. However, `mateda` is implemented in `matlab` and seems to be no longer updated.

	EDAspy	mateda	inspyred	LEAP
Language	python	matlab	python	python
$UMDA_C$	X		X	
$UMDA_D$	X			
$UMDA_B$	X			
u_KEDA	X			
PBIL	X			X^x
BOA	X			X^x
EMNA	X	X		
EGNA	X	X		
SPEDA	X			
m_KEDA	X			
EBNA	X	X		
Custom	X	X		

Table 12.1: Summary of functionalities implemented in each library. Note that X^x denotes that the implementation is expected to be released in the near future.

12.4 Performance analysis

In this section we compare the performance of different continuous domain optimizers implemented in EDAspy. For the evaluation three different cost functions (to be minimized) have been selected from the benchmark suite in EDAspy: CEC14_3, CEC14_4 and CEC14_8, where the former is unimodal and the rest are multimodal functions.

Section 12.3 reviewed some existing software for EDAs in different programming languages. In this section we also compare the result found by the $UMDA_C$ approach implemented in inspyred. Although mateda and LEAP were also reviewed, the former is implemented in a different programming language, and thus it is not fair to be compared in terms of CPU time, and the latter does not currently include any of the implemented approaches.

All the optimizers have been configured equally in order to perform a fair comparison. Hyper-parameters and a more extended tutorial can be found in the original documentation[4].

Since a statistical study is out of the scope of the paper (see Soloviev et al. [2023a] for a more complete analysis), we show a runtime and final solutions analysis of the different variants for continuous optimization in EDAspy.

Figure 12.3 shows the mean best cost found after 5 independent executions. It is generally observed how in the three functions the best approaches are SPEDA, m_KEDA and EGNA, which find the minimal costs in the benchmarks. Previous analyses have shown that m_KEDA, SPEDA and EGNA approaches are able to achieve statistically significant improvements in terms of quality of solutions [Soloviev et al., 2023a]. In the case of the $UMDA_C$ implementation from inspyred library, a slightly worse result is found in all the three benchmarks compared

[4] https://github.com/VicentePerezSoloviev/EDAspy/blob/master/notebooks/CPU%20time%20analysis.ipynb

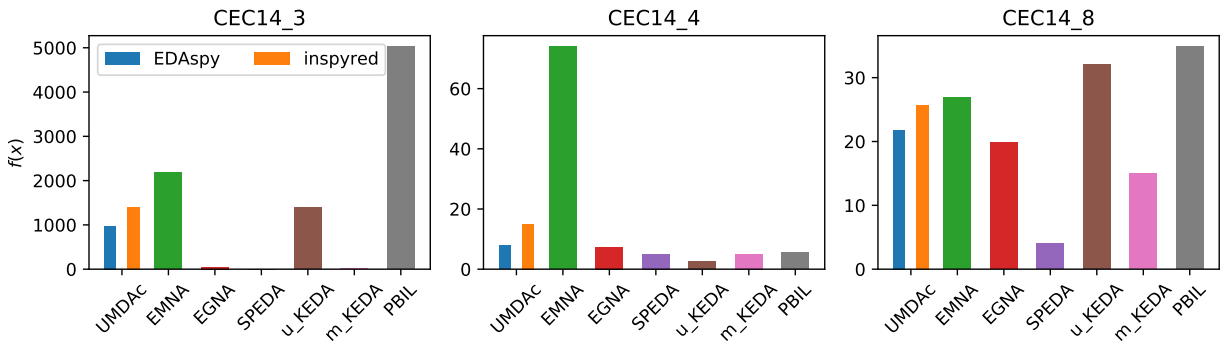


Figure 12.3: Best cost found analysis of some EDA variants for continuous optimization. UMDA$_C$, EMNA, EGNA, SPEDA, univariate KEDA (u_KEDA), multivariate KEDA (m_KEDA) and PBIL are shown.

to the implementation provided in `EDAspy`.

Figure 12.4 shows the mean CPU times of all the tested algorithms after 5 independent executions. Note that all the tested approaches have been configured in the same environment, that is, the number of function evaluations and hardware. It is observed that generally the higher the complexity of the probabilistic model embedded, the longer the CPU time required. However, PBIL is one of the slowest approaches in the comparison for CEC14_3. In this case, the multivariate version of KEDA is the most expensive algorithm in terms of CPU time, followed by SPEDA and EGNA. In the case of the UMDA$_C$ implementation from `inspyred` library, our implementation seems to be more efficient implemented in terms of CPU time consumption, keeping a good performance in terms of results found (Figure 12.3).

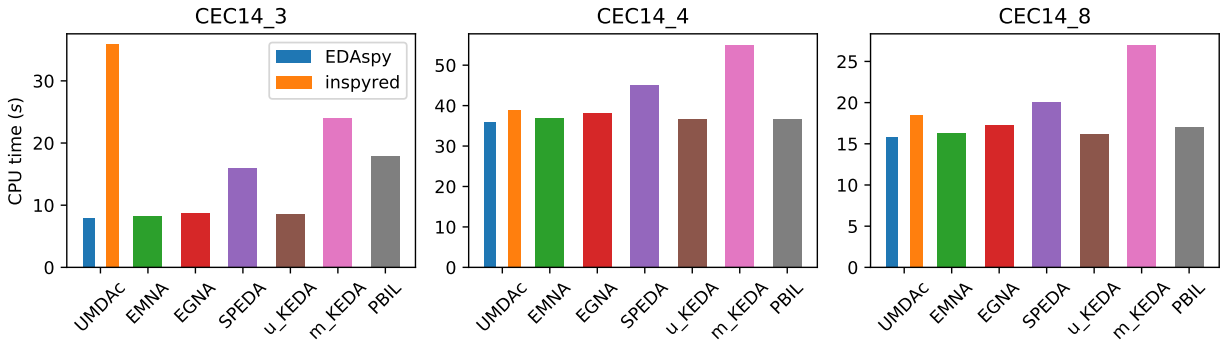


Figure 12.4: CPU runtime analysis of some EDA variants for continuous optimization. UMDA$_C$, EMNA, EGNA, SPEDA, univariate KEDA (u_KEDA), multivariate KEDA (m_KEDA) and PBIL are shown.

12.5 Illustrative examples

The following examples are available in the original documentation[1], where different EDAs are applied to different tasks:

- Using UMDA_C for continuous optimization. UMDA_C is tested on a IEEE CEC 2014 benchmark.
- Using SPEDA for continuous optimization. SPEDA is tested on a provided benchmark and several convergence plots are shown.
- Using EGNA for continuous optimization. SPEDA is tested on a provided benchmark and the plotting tools module is used to graphically show the probabilistic model embedded into the EDA approach.
- Using EMNA for continuous optimization. EMNA is tested on a IEEE CEC 2014 benchmark.
- Using UMDA_D for feature selection in a toy example. Given a dataset and a forecasting model, UMDA_D is used to select the best subset of variables that optimizes the accuracy of the prediction.
- Categorical optimization using EBNA and UMDA_D. A categorical cost function is designed and optimized by EBNA and UMDA_D approaches.
- Building my own EDA implementation. A tutorial on how to customize an EDA implementation is provided.
- CPU time analysis. All the continuous domain EDA variants are tested against the same IEEE CEC 2014 benchmark.

12.6 Conclusions

In this chapter we presented the first `python` library entirely dedicated to EDA implementations. `EDAspy` has been shown to be easy to use, and to integrate with custom implementations. Therefore, we hope that `EDAspy` can speed up the development of research on EDAs and their applications.

In addition to maintaining the code and solving bugs found by `EDAspy` users, future work would include the following lines:

- Visualization tools to analyze EDAs convergence.

Part IV

CONCLUSIONS

Chapter 13

Conclusions and Future Work

In this chapter, we list the main contributions achieved in this thesis and discuss possible future research lines. The publications and submissions are also listed in this chapter.

Chapter outline

In Section 13.1 we review the main contributions of this work. Section 13.2 provides a list of works published or submitted during this PhD thesis. Section 13.3 describes the software implemented in this research for the methodologies implementation and its respective experimentation. To sum up, Section 13.4 rounds the document off with future open research lines.

13.1 Summary of contributions

The main contributions achieved in this thesis are listed below:

- Chapter 5 studies the behavior the EGNA approach facing an optimization problem from the industry 4.0. The proposed approach introduces the concept of the *environment variables*, which restricts the search space in which the EDA is sampling the GBN. Moreover an additional configurable parameter is added in the algorithm to regulate the level of exploration vs. exploitation of the algorithm. Our results show that the proposed approach is able to imitate the proposals delivered by the experts in the industry when the level of exploitation is high. On the other hand, increasing the level of exploration allows our approach to find economically cheaper solutions to the optimization problem and outperform other state-of-the-art algorithms. This contribution satisfies objective **O7**.
- Chapter 6 extends the concept of *environment variables* with the EGNA approach to a multi-objective optimization problem in the industry 4.0. Our approach is used for the experimental design of a chemical laboratory which produces fuel. The EDA approach allows to reduce the number of experiments carried out in the laboratory by learning the dependencies and behavior of the chemical components and their relations in the

main formulas proposed. A regression model assists the optimizer by predicting some of the characteristics of the fuel formula that cannot be computed analytically and have to be evaluated in the laboratory. Our results show that the proposed approach is able to provide a set of optimal solutions that outperform the ones found by other optimizers. Moreover, the probabilistic model used finds dependencies between the variables which were unknown by the experts in the field. This contribution satisfies objective **O8**.

- Chapter 7 introduces a new methodology (SPEDA) in which the EDA is not restricted to assume a given probability distribution. SPEDA decides itself during the algorithm runtime whether to use Gaussians or KDE to each of the nodes in the embedded BN, and allows to combine different types of nodes within the same model. Moreover, the algorithm uses the concept of archive in which the probabilistic model is learned not only from the best solutions in the last iteration but also from those found in the last l iterations, where l is defined by the user. Numerical results show that our approach is able to outperform the state-of-the-art EDA approaches and overcome the premature convergence found in traditional EDA variants in different benchmark suites and a portfolio optimization problem. This contribution satisfies objective **O1**.
- Chapter 8 combines the EDA approach and QC technology by proposing an EDA variant in which the probabilistic model is replaced by a quantum circuit. The approach was applied to the TSP problem and the results simulating quantum noise outperform the results found by the approach without noise and other optimizers. We also analyze the quantum circuit depth after transpilation for different quantum devices, and propose the ideal quantum topology for our algorithm. This contribution satisfies objective **O5**.
- Chapter 9 applies the QAOA for learning the structure of a BN given some data by reformulating a QUBO problem into a QAOA *ansatz*. Numerical results show competitive results compared to other classical approach although the size of the learned models is limited by the size of the NISQ era quantum devices. To complement these results, a quantum-inspired technology has been applied to solve this problem facing much bigger BNs and again providing competitive results compared to classical approaches and outperforming them in some of the experiments. This contribution satisfies objective **O2**.
- Chapter 10 proposes the use of EDA approaches to optimize the parameters of the VQAs *ansatz*. We compare the performance of three different EDA variants for the parameter optimization of a QAOA, a VQE and a VQE with quantum noise for different optimization problems. The results show that the EDA variants are able to outperform the state-of-the-art approaches in terms of CPU time and quality of solutions depending on the hyper-parameter tuning of the EDA. This contribution satisfies objective **O3**.
- Chapter 11 goes one step forward by not only optimizing the parameters of a given *ansatz* but also its architecture. The proposed methodology is assisted by a ML model that ranks the proposed *ansatzes* skipping the parameter optimization step. Moreover, the optimization procedure is formulated as a multi-objective problem in which the possibility of BP existence in the proposed *ansatz* is reduced. Our results compare the found solutions for some given Hamiltonians to the ones provided in a dataset, where

competitive results were found. This contribution satisfies objective **O4**.

- Chapter 12 presents `EDAspy` python library, in which the state-of-the-art EDA variants and new methodologies introduced during this document are included. To the best of our knowledge, the identified libraries in which EDA implementations are available are very limited, or outdated, so we present this library in order to facilitate future research and applications on the topic. This contribution satisfies objective **O6**.

13.2 List of publications

Peer-reviewed JCR journals

- Soloviev, V. P., Larrañaga, P.,& Bielza, C. (2022). Estimation of distribution algorithms using Gaussian Bayesian networks to solve industrial optimization problems constrained by environment variables. *Journal of Combinatorial Optimization*, 44(2), 1077-1098.
- Soloviev, V. P., Bielza, C.,& Larrañaga, P. (2022). Quantum approximate optimization algorithm for Bayesian network structure learning. *Quantum Information Processing*, 22(1), 19.
- Soloviev, V. P., Bielza, C.,& Larrañaga, P. (2023). Semiparametric estimation of distribution algorithms for continuous optimization. *IEEE Transactions on Evolutionary Computation*.
- Soloviev, V. P., Larrañaga, P.,& Bielza, C. (2024). EDAspy: An extensible python package for estimation of distribution algorithms. *Neurocomputing*, 128043.
- Soloviev, V. P., Larrañaga, P., Bernabei, M., Chirita, M.A., Seoane, J.M., Fontán, P., & Bielza, C. (2024). A multi-objective framework based on estimation of distribution algorithms for data-driven fuel experimental design. *Submitted*.
- Soloviev, V. P., Dunjko, V., Bielza, C., Larrañaga, P., & Wang, H. (2024). Trainability maximization using estimation of distribution algorithms assisted by surrogate modelling for quantum architecture search. *EPJ Quantum Technology*, 11(1), 69.

Peer-reviewed conferences

- Soloviev, V. P., Bielza, C.,& Larrañaga, P. (2021). Quantum-inspired estimation of distribution algorithm to solve the travelling salesman problem. In *IEEE Congress on Evolutionary Computation* (pp. 416-425). IEEE. Online
- Soloviev, V. P., Larrañaga, P.,& Bielza, C. (2022). Quantum parametric circuit optimization with estimation of distribution algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 2247-2250). Boston, USA.
- Soloviev, V. P., Larrañaga, P.,& Bielza, C. (2023). Variational quantum algorithm parameter tuning with estimation of distribution algorithms. In *IEEE Congress on Evolutionary Computation* (pp. 1-9). IEEE. Chicago, USA.

Technical reports

- Ceberio, J., Doerr, B., Witt, C., & Soloviev, V. P. (2022). Estimation-of-distribution algorithms: Theory and applications (Dagstuhl Seminar 22182).

Poster sessions

- Soloviev, V. P. Bayesian network structure learning using quantum computing. Poster presented at: ELLIS Doctoral Symposium. 2022. Alicante, Spain.
- Soloviev, V. P. A probabilistic perspective for optimizing the parameters of quantum heuristics using evolutionary algorithms. Poster presented at Quantum Information in Spain ICE-8. 2023 (Santiago de Compostela).

13.3 Software

All the code implemented for the experimentation of each contribution is publicly available through the respective GitHub repository and all of them can be found in this GitHub account[1]. However, two main contributions are worth to be mentioned:

- `EDAspy` library is the first python library dedicated to the EDAs research. It compiles most of the state-of-the-art EDA approaches and is extended with the proposed methodologies presented in this document. Moreover, the library has been carefully implemented in order to ease the future extensibility of the current implementations. `EDAspy` is currently been used in several private companies and academic research centers as well.
- Several contributions have been done to `Qiskit` library in which the EDAs have been introduced, among other bugs solved.

13.4 Future work

In this section we provide a list of open research lines identified along the document.

- **Automatic tuning of exploration vs. exploitation parameter**. Chapter 5 and Chapter 6 introduce a novel parameter which regulates the level of exploration versus exploitation in a given landscape of solutions. During this document we have shown an analysis of the parameter as a function of the quality of the solutions. An automatic refinement of this hyper-parameter during runtime is still an open question in this research. Configuring it a priori may restrict the algorithm from reaching optimal solutions, however, some patterns should be identified in the sampled solutions in order to configure it during runtime.
- **Geometric metrics for regulating exploration vs. exploitation**. Chapter 5 and Chapter 6 control the level of exploration and exploitation by computing the log-

[1] https://github.com/VicentePerezSoloviev/

likelihood. Here we propose using different metrics to compute this, such as geometric distances in the landscape. Although this computation may be much more expensive, it may result in better results; however, this is still an open question.

- **Reduce CPU time of SPEDA**. SPEDA has shown to outperform the EDA state-of-the-art variants, however, the CPU runtime does not easily scale with the size of the problem. Reducing the CPU runtime is still an open question in this research. We propose not learning the BN structure within a window of iterations in which only the parameters of the BN are learned. Our intuition is that a big window will lead to local convergence, and a small window will compensate the CPU savings with an increasing number of needed function evaluations.
- **Multi-objective SPEDA**. Extending SPEDA to multi-objective optimization tasks is still and open research line. Several options may apply here, such as: (i) adding the cost functions as nodes in the BN, (ii) or transforming the multi-objective tasks into single-objective optimization through metrics such as the hypervolume.
- ***Environment variables* for SPEDA**. SPEDA has shown to outperform the EDA state-of-the-art variants for different benchmark suites. However, it would be very interesting to showcase the value of this approach for real optimization problems in which the data do not fit Gaussians. Problems in the industry 4.0. usually involve setting specific values to some of the variables. Thus, introducing *environmental variables* to SPEDA would allow to expand the usability of this approach in real problems. On the other hand no exact formulas are proposed in the literature for computing the conditional probability of the decision variables given the *environmental variables* in a SPBN.
- **QIEDA without PLS**. Our quantum-inspired EDAs have shown to achieve good results for the TSP problem, however, their CPU time remains slow. The sampling process of the algorithm heavily depends on the number of cities present in the TSP, which grows linearly with this number. Sampling each row of the codification depends on the previous row sampling. Classical communication in between the quantum circuit execution may allow to sample the full individual for the TSP, and thus saving a lot of computational resources. However, this may reduce the effect of the quantum noise that we have found to be beneficial to our approach. Analyzing this trade-off is still an open research question.
- **Extend QIEDA for other tasks**. QIEDA has been explicitly designed for the TSP problem in which the sampling procedure is valid for the matrix representation of the problem. Extending this method for a general optimization purpose is still an open research line.
- **Dynamic environments**. All the approaches proposed in this PhD Thesis are focused on optimization tasks in which the cost function remains constant during all the runtime. Dynamic optimization is a type of optimization problem where the objective is to find the best possible solution over runtime, considering the changes and dynamics of the problem. Unlike static optimization, where the parameters and constraints are fixed, dynamic optimization deals with problems where the parameters, constraints,

or objectives can change over time [Nguyen et al., 2012]. Extending the proposed methodologies in Chapters 5-7 is still an open research line.

- **AutoEDA**. We have found that combining different types of nodes (Gaussian and KDE) outperforms the state of the art. Here we propose an approach in which in each iteration the EDA itself decides whether to use different probabilistic models. By this way, the EDA may initially use univariate approaches such as UMDA, and end the execution exploiting the patterns found by the SPEDA approach, or viceversa. This is still an open research line.
- **Explicitly analyze EDAs with VQAs with BPs**. We have proposed in Chapter 10 the use of EDAs for different VQA parameter optimization. The main issue encountered in the parameter optimization is the existence of BPs in the landscape of solutions [Ragone et al., 2023]. A deep understanding on how the EDA approaches avoid getting stuck in BPs is still an open research line.
- **Dynamic definition of HV reference point**. Chapter 6 and Chapter 11 propose different extensions of the EDA approaches for multi-objective tasks. The HV metric has been used combining the objective functions. However, this involves defining a reference point with which being compared during all runtime. Here we propose a dynamic definition in which in each iteration the algorithm adjusts the reference point according to the solutions sampled [Ishibuchi et al., 2018]. This adjustment also has to be considered when comparing solutions from different iterations.
- **Visualization tools**. We have identified several research frameworks in which visualization tools are proposed analyze the algorithm trajectory within the landscape of solutions [Shine and Eick, 1997, De Lorenzo et al., 2019, Fyvie et al., 2023]. Adding a visualization module to `EDAspy` would add value to the framework.

Part V

APPENDIX

Appendix A

Benchmarking Functions

This appendix describes the benchmark suites mentioned in both Chapter 7 and Chapter 12. CEC2014 [Liang et al., 2013] and CEC2017 [Wu et al., 2017] benchmarks are usually used for numerical experimentation. Note that all the rotational and shift matrices were obtained from the original benchmarks.

A.1 CEC2014 benchmark

Table A.1 describes the CEC2014 functions tested. The functions are divided into unimodal, multimodal, hybrid, and composition functions, in the first, second, third, and fourth groups from top to bottom, respectively. Additionally, cec14_8 and cec14_10 are separable functions, while the rest are not. Hybrid functions are built in a way in which the search space is divided into different types of functions, and composition functions are similar to hybrid ones, but adding more complex modifications and being able to combine hybrid functions with simpler ones.

A.2 CEC2017 benchmark

Table A.2 describes the CEC2017 functions tested. The functions are divided in-between unimodal, multimodal and hybrid functions, in the first, second, and third groups from top to bottom, respectively. All of them are non-separable functions, and most of them have a large number of local optima.

Table A.1: CEC2014 single objective minimization test benchmarks used in the experiments. cec14_1 - cec14_3 benchmarks are uni-modal functions, cec14_4 - cec14_16 are multi-modal, cec14_17 - cec14_22 are hybrid functions that combine previously defined functions, and cec14_23 - cec14_30 are composition functions that combine previously defined functions and additional modifications over them.

	Description	Characteristics
cec14_1	Rotated High Conditioned Elliptic function	Quadratic ill-conditioned and non-separable
cec14_2	Rotated Bent Cigar function	Smooth but narrow ridge and non-separable
cec14_3	Rotated Discus function	With one sensitive direction and non-separable
cec14_4	Shifted and rotated Rosenbrock's function	Very narrow valley from local to global optimum and non-separable
cec14_5	Shifted and rotated Ackley's function	Epistasis and non-separable
cec14_6	Shifted and rotated Weierstrass function	Continuous but differentiable only on a set of points and non-separable
cec14_7	Shifted and rotated Griewank's function	Rotated and non-separable
cec14_8	Shifted Rastrigin's function	Local optima's number is huge and separable
cec14_9	Shifted and rotated Rastrigin's function	Local optima's number is huge and non-separable
cec14_10	Shifted Schwefel's function	Many local optimas far from the global optima and separable
cec14_11	Shifted and rotated Schwefel's function	Many local optimas far from the global optima and non-separable
cec14_12	Shifted and rotated Katsuura function	Continuous everywhere yet differentiable nowhere and non-separable
cec14_13	Shifted and rotated HappyCat function	Non-separable
cec14_14	Shifted and rotated HGBat function	Non-separable
cec14_15	Shifted and rotated Expanded Griewank's plus Rosenbrock's function	Non-separable
cec14_16	Shifted and rotated Expanded Scaffer's F6 function	Deceptive and non-separable
cec14_17	Hybrid function 1	Hybridization of 3 different functions
cec14_18	Hybrid function 2	Hybridization of 3 different functions
cec14_19	Hybrid function 3	Hybridization of 4 different functions
cec14_20	Hybrid function 4	Hybridization of 4 different functions
cec14_21	Hybrid function 5	Hybridization of 5 different functions
cec14_22	Hybrid function 6	Hybridization of 5 different functions
cec14_23	Composition function 1	Multi-modal, non-separable, asymmetric
cec14_24	Composition function 2	Multi-modal, non-separable
cec14_25	Composition function 3	Multi-modal, non-separable, asymmetric
cec14_26	Composition function 4	Multi-modal, non-separable, asymmetric
cec14_27	Composition function 5	Multi-modal, non-separable, asymmetric
cec14_28	Composition function 6	Multi-modal, non-separable, asymmetric
cec14_29	Composition function 7	Multi-modal, non-separable, asymmetric, different properties for different D
cec14_30	Composition function 8	Multi-modal, non-separable, asymmetric, different properties for different D

Table A.2: CEC2017 single objective minimization test benchmarks used in the experiments. cec17_1 - cec17_2 benchmarks are uni-modal functions, cec17_3 - cec17_9 are multi-modal, and cec17_10 - cec17_19 are hybrid functions that combine previously defined functions. All the functions in this benchmark are non-separable.

	Description	Characteristics
cec17_1	Shifted and rotated Bent Cigar Function	Non-separable, with a smooth but narrow ridge
cec17_2	Shifted and rotated Zakharov Function	Non-separable
cec17_3	Shifted and rotated Rosenbrock's Function	Non-separable with huge number of local optima
cec17_4	Shifted and rotated Rastrigin's Function	Non-separable with huge number of local optima
cec17_5	Shifted and rotated Expanded Scaffer's F6 Function	Non-separable with huge number of local optima
cec17_6	Shifted and rotated Lunacek Bi_Rastrigin Function	Non-separable, asymmetrical, continuous everywhere yet differentiable nowhere
cec17_7	Shifted and rotated Non-Continuous Rastrigin's Function	Non-separable with huge number of local optima
cec17_8	Shifted and rotated Levy Function	Non-separable with huge number of local optima
cec17_9	Shifted and rotated Schwefel's Function	Non-separable with huge number of local optima
cec17_10	Hybrid function 1	Hybridization of 3 different functions
cec17_11	Hybrid function 2	Hybridization of 3 different functions
cec17_12	Hybrid function 3	Hybridization of 3 different functions
cec17_13	Hybrid function 4	Hybridization of 4 different functions
cec17_14	Hybrid function 5	Hybridization of 4 different functions
cec17_15	Hybrid function 6	Hybridization of 4 different functions
cec17_16	Hybrid function 7	Hybridization of 5 different functions
cec17_17	Hybrid function 8	Hybridization of 5 different functions
cec17_18	Hybrid function 9	Hybridization of 5 different functions
cec17_19	Hybrid function 10	Hybridization of 6 different functions

Appendix B

Exploration Data Analysis

This appendix describes the dataset used in Chapter 6. It was provided by the experts in the industry and has been used for the EDA approach initialization.

B.1 Ingredients and properties description

Table B.1 shows the properties $P1, \ldots, P14$ associated to each of the $n = 24$ ingredients for the fuel fabrication. Note that the names of the ingredients are denoted referring to the category to which they belong. For example, $A1$ is the first ingredient of category A and $B2$ the second one of category B.

Figure B.1 shows a histogram for each of the properties $P1, \ldots, P14$.

B.2 Optimization constraints

Table B.2 shows the lower and upper bounds, respectively, for analytical functions $W^i_{calc}(i = 1, \ldots, 7)$ and laboratory experimentation $W^i_{lab}(i = 1, \ldots, 4)$.

		P1	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	*P9*	*P10*	*P11*	*P12*	*P13*	*P14*
I_1	*A1*	678.6	76.5	42.0	100.0	100.0	0.0	0.0	99.4	0.0	85.6	14.4	6.0	12.0	0.0
I_2	*A2*	667.0	97.7	102.2	100.0	100.0	0.0	0.0	99.9	0.1	85.6	14.4	5.0	10.0	0.0
I_3	*A3*	816.3	83.9	26.8	0.0	100.0	0.0	0.0	99.9	0.1	87.7	12.3	6.0	10.0	0.0
I_4	*A4*	777.1	93.4	68.4	100.0	100.0	0.0	0.0	98.9	0.0	87.6	11.8	4.9	7.9	0.0
I_5	*B1*	652.6	85.6	78.5	100.0	100.0	0.0	0.0	0.0	0.0	68.7	13.5	4.3	10.1	0.0
I_6	*B2*	652.5	88.6	78.4	100.0	100.0	0.0	0.0	0.0	0.0	83.6	16.4	5.7	13.3	0.0
I_7	*C1*	784.1	83.0	22.0	0.0	100.0	0.0	0.0	0.0	0.0	85.6	14.4	6.0	12.0	0.0
I_8	*C2*	746.6	100.6	66.0	100.0	100.0	0.0	0.0	0.0	0.0	85.5	14.5	5.0	10.1	0.0
I_9	*C3*	754.0	91.0	31.6	0.0	100.0	0.0	0.0	0.0	0.1	85.6	14.4	6.0	12.0	0.0
I_{10}	*D1*	723.0	105.0	33.0	100.0	100.0	0.0	15.7	0.0	0.0	70.5	13.8	6.0	14.0	1.0
I_{11}	*D2*	751.6	113.4	35.7	32.0	100.0	100.0	14.7	3.3	0.1	71.1	13.9	5.8	13.6	0.9
I_{12}	*D3*	751.6	113.4	35.7	32.0	100.0	50.0	14.7	3.3	0.1	71.1	13.9	5.8	13.6	0.9
I_{13}	*D4*	793.7	105.4	17.1	0.0	100.0	100.0	34.7	0.0	0.0	52.1	13.1	2.0	6.0	1.0
I_{14}	*D5*	800.0	105.1	8.0	0.0	0.0	0.0	21.6	0.0	0.0	64.8	13.6	4.0	10.0	1.0
I_{15}	*D6*	746.6	117.0	60.1	100.0	100.0	39.0	18.1	0.0	0.0	68.1	13.7	5.0	12.0	1.0
I_{16}	*D7*	796.9	106.0	32.8	100.0	100.0	100.0	49.9	0.0	0.0	37.5	12.6	1.0	4.0	1.0
I_{17}	*D8*	770.9	114.0	19.5	0.0	100.0	0.0	15.1	0.8	0.3	70.9	13.8	5.8	13.4	0.9
I_{18}	*E1*	625.2	91.5	135.1	100.0	100.0	0.0	0.0	0.0	0.0	83.2	16.8	5.0	12.0	0.0
I_{19}	*E2*	699.4	96.7	12.7	0.0	64.0	0.0	0.0	0.0	0.0	81.9	15.5	7.7	17.4	0.0
I_{20}	*E3*	707.3	77.4	30.3	26.3	67.4	100.0	0.0	2.5	1.9	50.8	9.0	3.1	6.5	0.0
I_{21}	*E4*	756.8	90.3	54.4	23.9	40.0	100.0	0.0	5.3	38.1	89.7	10.3	3.3	6.9	0.0
I_{22}	*F1*	871.4	107.4	3.4	0.0	0.0	0.0	0.0	0.0	100.0	90.5	9.5	8.0	10.0	0.0
I_{23}	*F2*	871.4	120.1	6.4	0.0	0.0	0.0	0.0	0.0	99.9	91.1	8.7	7.0	8.0	0.0
I_{24}	*F3*	868.4	117.5	2.7	0.0	0.0	0.0	0.0	0.0	100.0	90.5	9.5	8.0	10.0	0.0

Table B.1: Properties $P1, \ldots, P14$ associated to each of the ingredients $I_1, \ldots, I_{24}$ used for the fuel fabrication and its name denoting the category to which they belong.

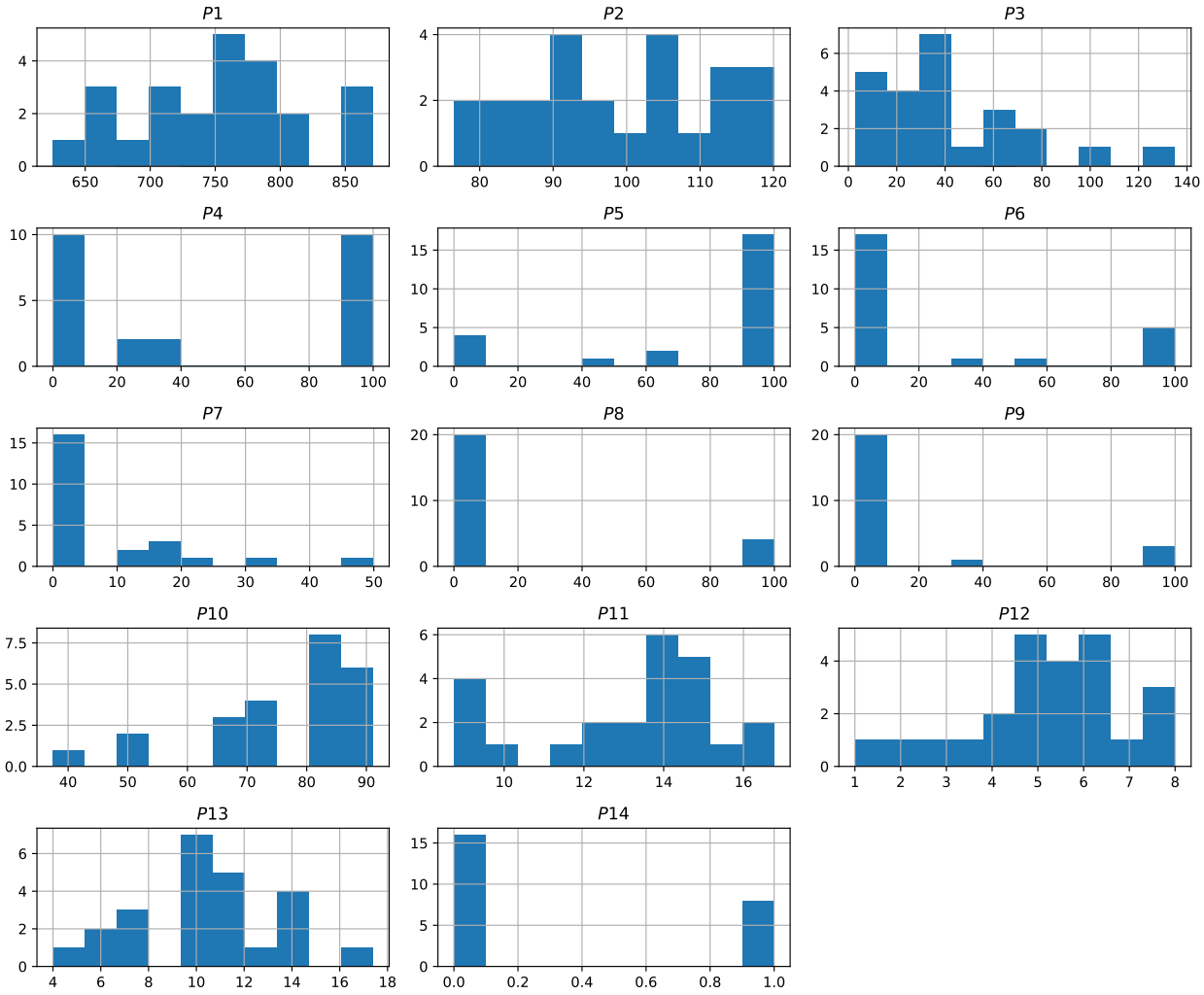


Figure B.1: Frequency of each property $P1, \ldots, P14$.

	Formula	LB	UB
$W^1_{calc}(\boldsymbol{x})$	$\frac{100}{\sum_{i=1}^{n}(x_i/P1(I_i))}$	LB^1_{calc}	UB^1_{calc}
$W^2_{calc}(\boldsymbol{x})$	$\sum_{i=1}^{n}\left(x_i\frac{1000(\frac{32.8P10(I_i)}{100}+141.8(\frac{P11(I_i)}{100}-\frac{P7(I_i)}{800}))-\frac{2440*9P11(I_i)}{100}}{\frac{P12(I_i)+P13(I_i)/4-P14(I_i)/2}{6.09/(12P12(I_i)+P13(I_i)+16P14(I_i))}}\right)$	LB^2_{calc}	UB^2_{calc}
$W^3_{calc}(\boldsymbol{x})$	$\frac{\sum_{i=1}^{n}x_iP8(I_i)/P1(I_i)}{\sum_{i=1}^{n}x_i/P1(X_i)}$	LB^3_{calc}	UB^3_{calc}
$W^4_{calc}(\boldsymbol{x})$	$\sum_{i=1}^{n}x_iP7(I_i)$	LB^4_{calc}	UB^4_{calc}
$W^5_{calc}(\boldsymbol{x})$	$\frac{\sum_{i=1}^{n}x_iP9(I_i)/P1(I_i)}{\sum_{i=1}^{n}x_i/P1(I_i)}$	LB^5_{calc}	UB^5_{calc}
$W^6_{calc}(\boldsymbol{x})$	$\sum_{i=1}^{n}x_iP6(I_i)$	LB^6_{calc}	UB^6_{calc}
$W^7_{calc}(\boldsymbol{x})$	x_{16}	LB^7_{calc}	UB^7_{calc}
$W^1_{lab}(\boldsymbol{x})$	-	LB^1_{lab}	UB^1_{lab}
$W^2_{lab}(\boldsymbol{x})$	-	LB^2_{lab}	UB^2_{lab}
$W^3_{lab}(\boldsymbol{x})$	-	LB^3_{lab}	UB^3_{lab}
$W^4_{lab}(\boldsymbol{x})$	-	LB^4_{lab}	UB^4_{lab}

Table B.2: Formula, lower bound (LB) and upper bound (UB) associated to each analytical descriptor W^i_{calc} and each laboratory descriptor W^i_{lab}; x_i is the percentage of each ingredient I_i in the total mixture $\boldsymbol{x}$, and $Pj(I_i)$ is the property j of ingredient I_i (see Table B.1).

Appendix C

Algorithms Configuration

This appendix shows the algorithms configuration used in Chapter 6.

C.1 Configuration of regression models

In this section, we detail the hyperparameter configurations applied to the regression models tested to predict W^1_{lab}. Tables C.1-C.2 show the hyperparameters used in the experimental results for Bayesian ridge, Lasso, Ridge and Kernel ridge models. Note that categorical hyperparameters were used as default in scikit-learn-1.3.0 [Pedregosa et al., 2011]. Bayesian ridge model was selected as the best regression model to predict W^1_{lab} values, so both the default and after tuning hyperparameters are shown. Hyperparameter tuning was performed by Bayesian optimization. All the experiments use the same maximum number of iterations set as default.

	tol	α_1	α_2	λ_1	λ_2
default	1e-3	1e-6	1e-6	1e-6	1e-6
hyperopt	94.64	50.16	18.59	81.89	0.20

Table C.1: Bayesian ridge model hyperparameters by default and after hyperparameter tuning (hyperopt), where *tol* is the tolerance error to consider convergence; α_1 and α_2 are the shape parameter for the Gamma distribution and the inverse scale parameter for α, respectively; λ_1 and λ_2 are the shape parameter for the Gamma distribution and the inverse scale parameter for λ, respectively.

C.2 Competitors configuration

In this section, we detail the hyperparameter configuration applied to the other optimizers used in the comparison against the EDA approach presented in this article. NSGA-II and MOEAD implementations have been obtained from PYMOO Python library [Blank and Deb,

	α_3	*tol*	degree
Lasso	0.5	1e-4	-
Ridge	0.5	1e-4	-
Kernel ridge	10	-	3

Table C.2: Lasso, Ridge and Kernel ridge regression models hyperparameter configurations used. Parameter α_3 is the constant that multiplies L1, L2 terms in Lasso and Ridge regression models, respectively, and *tol* represents the tolerance error to consider algorithm convergence. In the case of Kernel ridge *degree* is the polynomial degree used.

2020] and both algorithms were initialized from the historical data. Tables C.3-C.4 show the hyperparameter configurations used for both algorithms in experimental results.

			Crossover		Mutation	
	N	*offs*	P	*eta*	P	*eta*
tested	200	10	0.9	15	1.0	20

Table C.3: Hyperparameters used for NSGAII algorithm [Deb et al., 2002], where N is the population size, *offs* regards the number of *offsprings* performed; P is probability of crossover and mutation occurrence; and *eta* parameter controls the shape of the distribution used for both crossover and mutation operators.

	n_neig	P
tested	15	0.7

Table C.4: Hyperparameters used for MOEAD algorithm [Zhang and Li, 2007], where *n_neig* is the number of neighbors considered and P is the probability of neighbor mating occurrence.

Appendix D

Large BNSL using Digital Annealing

D.1 Problem size fitted into device

See Figure D.1.

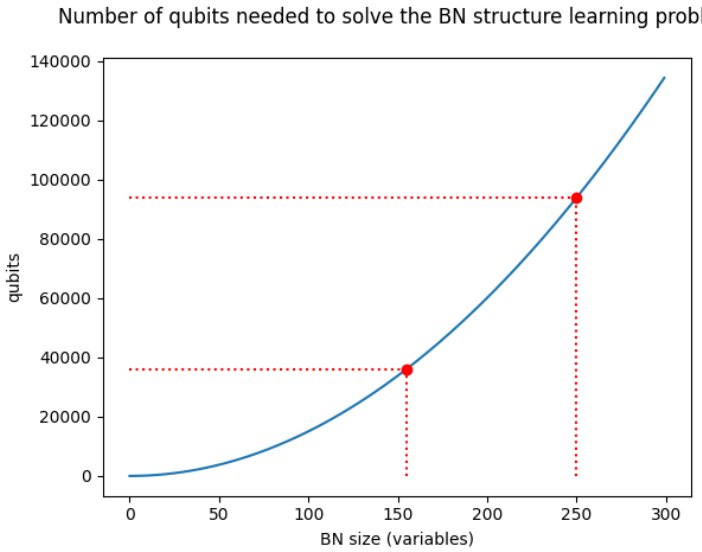


Figure D.1: In horizontal axis, the number of nodes, and in y axis, the number of quantum/classical bits needed for the QUBO formulation.

D.2 20 nodes BNSL

See Figure D.2.

D.3 50 nodes BNSL

See Figure D.3.

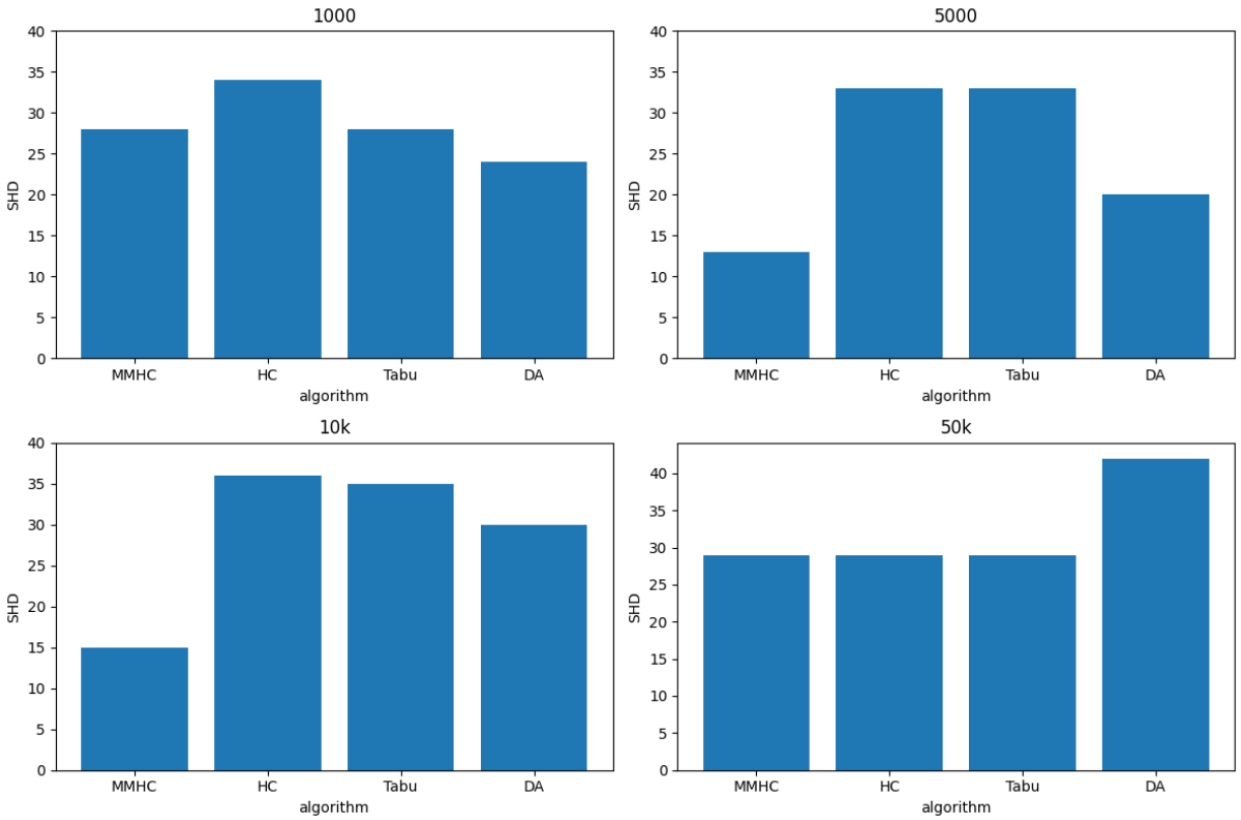


Figure D.2: Experiment for 20 nodes. Each subplot is a different size of dataset: 50000, 10000, 5000 and 1000 instances. DA is compared with the Hill Climbing algorithm (HC), Tabu search algorithm (Tabu), and Max-Min Hill Climbing algorithm (MMHC).

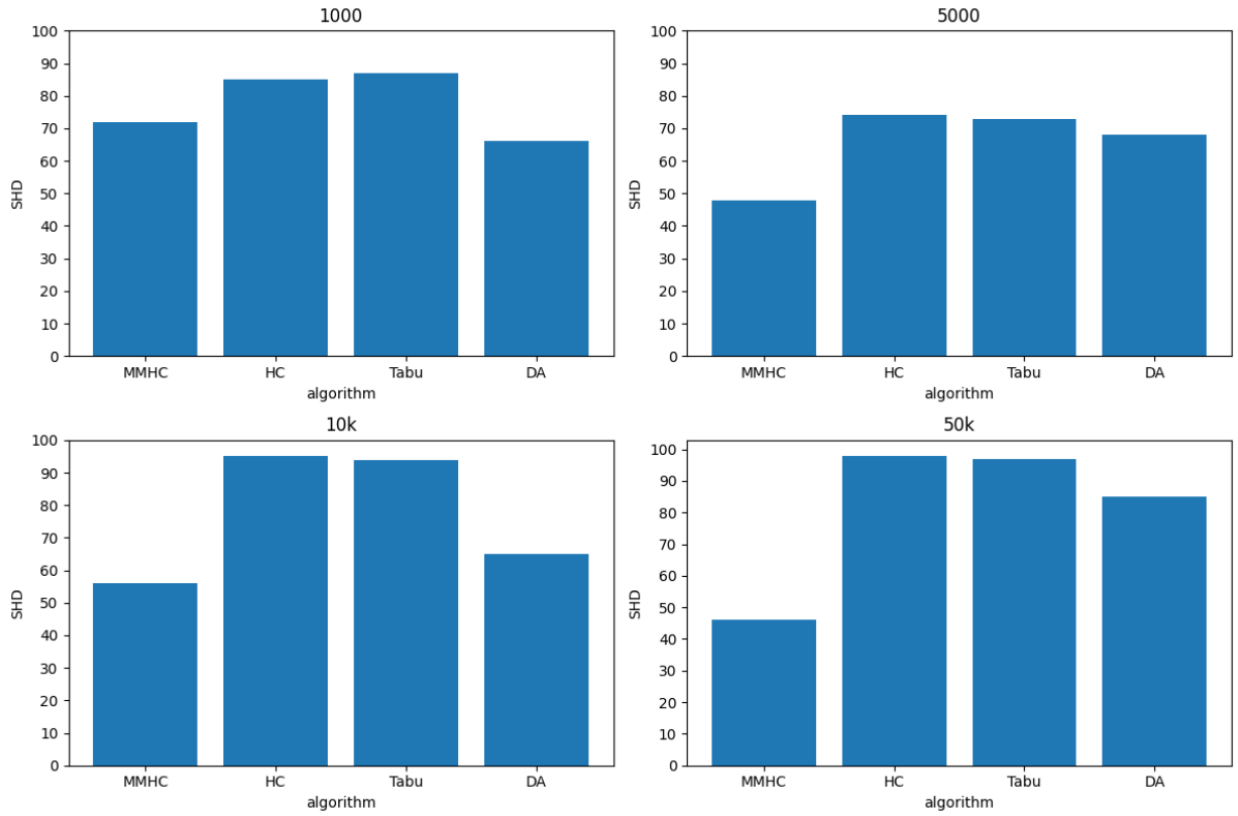


Figure D.3: Experiment for 50 nodes. Each subplot is a different size of dataset: 50000, 10000, 5000 and 1000 instances. DA is compared with the Hill Climbing algorithm (HC), Tabu search algorithm (Tabu), and Max-Min Hill Climbing algorithm (MMHC).

Appendix E

Complementary Materials for Quantum Architecture Search

E.1 Hamiltonians

This section describes the Hamiltonians used for the experimental results. Note that the following benchmarks and coefficients have been used in order to compare the results with the ones found in Nakayama et al. [2023].

1D transverse-field Ising model:

$$H_1 = \sum_{i=1}^{n-1} Z_i Z_{i+1} + 2\sum_{i=1}^{n} X_n$$

1D Heisenberg model:

$$H_2 = \sum_{i=1}^{n-1} (X_i X_{i+1} + Y_i Y_{i+1} + Z_i Z_{i+1}) + 2\sum_{i=1}^{n} Z_n$$

Su-Schrieffer-Heeger model:

$$H_3 = \sum_{i=1}^{n-1} \left(1 + \frac{3}{2}(-1)^{i-1}\right) (X_i X_{i+1} + Y_i Y_{i+1} + Z_i Z_{i+1}) + 2\sum_{i=1}^{n} X_n$$

J_1 - J_2 model:

$$H_4 = \sum_{i=1}^{n-1} (X_i X_{i+1} + Y_i Y_{i+1} + Z_i Z_{i+1}) + 3\sum_{i=1}^{n-2} (X_i X_{i+2} + Y_i Y_{i+2} + Z_i Z_{i+2})$$

E.2 Surrogate model prediction

Here we compare the performance of different surrogate models by comparing different *ansatz* by pairs in a given initial data for different number of qubits.

Different architectures have been built for problems described in Appendix E.1 and different values of n. The number of architectures have been set to $N = 37.5n$, and the circuit depth to $m = 60$. Table E.1 shows the accuracy found for different models with different configurations. Results show that support vector classifier (SVC) achieves the best metrics, and thus, is used as surrogate model in our approach.

model	$n = 4$	$n = 8$	$n = 12$
Random_forest_20	0.76	0.77	0.75
Random_forest_50	0.81	0.82	0.80
Random_forest_80	0.82	0.83	0.80
KNN_2	0.64	0.66	0.68
KNN_5	0.72	0.74	0.75
KNN_15	0.78	0.79	0.79
SVC	**0.91**	**0.92**	**0.90**
Decision tree	0.64	0.65	0.65
Naive Bayes	0.69	0.76	0.78

Table E.1: Accuracy found after evaluating each model in a set of initial architectures using cross-validation with 15 folds. Independently of n, all the *ansatz* have been restricted to $m = 60$, and $N = 37.5n$. Random forest with different numbers of estimators, k-nearest neighbors (KNN) with different numbers of neighbors, support vector classifier (SVC), decision tree, and naive Bayes have been tested.

E.3 Distance computation

Here we detail the distance comparison between all the proposed solutions within EDA_i^n and each of the clusters D_i^n by computing Equation 11.11. Note that index j denotes each of the 5 best results found by the EDA. Table E.2-E.4 show the distance computations for $n \in [4, 8, 12]$, respectively.

E.4 Pareto frontier approximations

Figure E.1 shows the Pareto frontier approximation for different H and number of qubits. The columns refer to the problem instances, while the rows refer to the number of qubits (n). Each subplot shows all the evaluated *ansatz* (blue spots) from which the non-dominated solutions are highlighted (orange spot).

E.5 IC and expectation values comparison

Table E.5 describes the mean expectation value (Equation 4.9) and IC (Equation 4.15) for the *ansatz* available in the dataset (D_i^n) for different values of n.

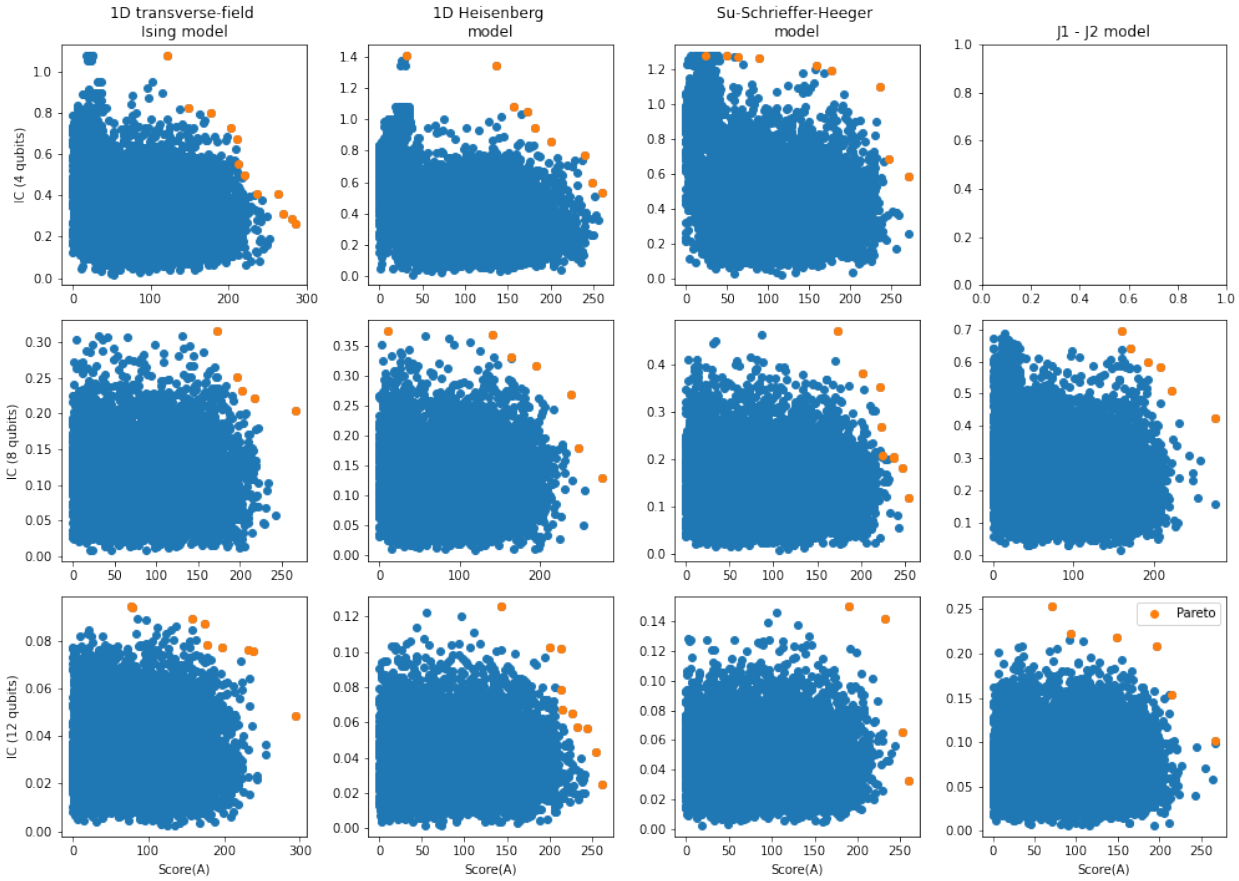


Figure E.1: Pareto frontier approximation (orange spots) over all the *ansatz* considered (blue spots) during optimization process. Columns refer to problem instances, while rows refer to number of qubits (n).

Table E.6 describes the best expectation value and IC found by the EDA approach for different H_i and values of n, where the HV is maximized. That is, the solutions which maximize HV within EDA_i^n.

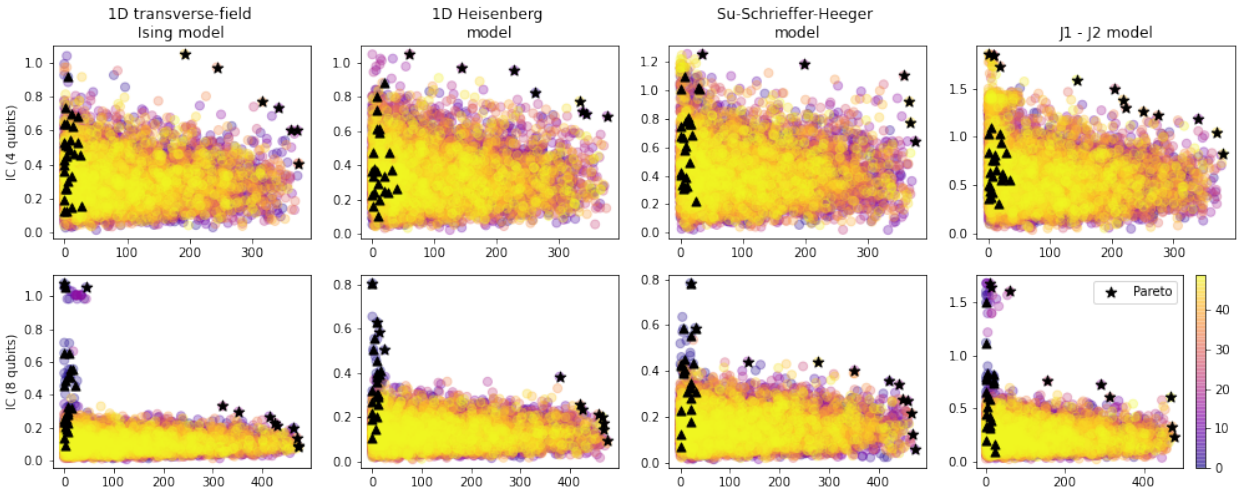


Figure E.2: Pareto frontier approximation (black stars) over all the *ansatz* considered (colored spots) during the optimization process. Black triangles regard the *ansatz* included in the dataset. Columns refer to problem instances, while rows refer to number of qubits (n).

ansatz ($\mathrm{EDA}^4_{i_j}$)	H_i	dist($\mathrm{EDA}^4_{1_j}, D^4_1$)	dist($\mathrm{EDA}^4_{2_j}, D^4_2$)	dist($\mathrm{EDA}^4_{3_j}, D^4_3$)	dist($\mathrm{EDA}^4_{4_j}, D^4_4$)
$\mathrm{EDA}^4_{1_1}$	H_1	**0.018**	0.998	0.990	0.999
$\mathrm{EDA}^4_{1_2}$	H_1	**0.011**	0.999	0.995	0.995
$\mathrm{EDA}^4_{1_3}$	H_1	**0.011**	0.999	0.989	0.999
$\mathrm{EDA}^4_{1_4}$	H_1	**0.027**	0.990	0.991	0.995
$\mathrm{EDA}^4_{1_5}$	H_1	**0.011**	0.999	0.989	0.999
$\mathrm{EDA}^4_{2_1}$	H_2	0.999	**0.038**	0.982	0.997
$\mathrm{EDA}^4_{2_2}$	H_2	0.999	**0.049**	0.993	0.999
$\mathrm{EDA}^4_{2_3}$	H_2	0.993	0.954	**0.233**	0.880
$\mathrm{EDA}^4_{2_4}$	H_2	0.999	**0.035**	0.976	0.990
$\mathrm{EDA}^4_{2_5}$	H_2	0.970	**0.374**	0.794	0.965
$\mathrm{EDA}^4_{3_1}$	H_3	0.993	0.999	**0.051**	0.660
$\mathrm{EDA}^4_{3_2}$	H_3	0.992	0.999	**0.058**	0.648
$\mathrm{EDA}^4_{3_3}$	H_3	0.988	0.998	**0.064**	0.646
$\mathrm{EDA}^4_{3_4}$	H_3	0.995	0.997	**0.069**	0.631
$\mathrm{EDA}^4_{3_5}$	H_3	0.987	0.999	**0.056**	0.637
$\mathrm{EDA}^4_{4_1}$	H_4	0.991	0.995	0.691	**0.077**
$\mathrm{EDA}^4_{4_2}$	H_4	0.993	0.992	0.752	**0.061**
$\mathrm{EDA}^4_{4_3}$	H_4	0.998	0.991	0.811	**0.081**
$\mathrm{EDA}^4_{4_4}$	H_4	0.992	0.997	0.702	**0.099**
$\mathrm{EDA}^4_{4_5}$	H_4	0.990	0.993	0.329	**0.011**

Table E.2: Distance (Equation 11.11) between each *ansatz* in $\mathrm{EDA}^4_{i_j}$ and D^4_i, where i denotes the Hamiltonian index and $n = 4$. Bold values represent those instances in which the closest cluster to $\mathrm{EDA}^4_{i_j}$ is D^4_i.

ansatz ($\text{EDA}^8_{i_j}$)	H_i	$\text{dist}(\text{EDA}^8_{1_j}, D^8_1)$	$\text{dist}(\text{EDA}^8_{2_j}, D^8_2)$	$\text{dist}(\text{EDA}^8_{3_j}, D^8_3)$	$\text{dist}(\text{EDA}^8_{4_j}, D^8_4)$
$\text{EDA}^8_{1_1}$	H_1	**0.973**	0.995	0.995	0.997
$\text{EDA}^8_{1_2}$	H_1	**0.950**	0.996	0.996	0.994
$\text{EDA}^8_{1_3}$	H_1	**0.830**	0.998	0.998	0.998
$\text{EDA}^8_{1_4}$	H_1	**0.553**	0.999	0.999	0.999
$\text{EDA}^8_{1_5}$	H_1	**0.942**	0.995	0.990	0.997
$\text{EDA}^8_{2_1}$	H_2	0.990	**0.926**	0.968	0.991
$\text{EDA}^8_{2_2}$	H_2	0.998	**0.906**	0.998	0.999
$\text{EDA}^8_{2_3}$	H_2	0.998	**0.963**	0.989	0.995
$\text{EDA}^8_{2_4}$	H_2	0.996	**0.992**	0.998	0.998
$\text{EDA}^8_{2_5}$	H_2	0.999	**0.991**	0.999	0.999
$\text{EDA}^8_{3_1}$	H_3	0.999	0.999	**0.957**	0.995
$\text{EDA}^8_{3_2}$	H_3	0.999	**0.958**	0.983	0.985
$\text{EDA}^8_{3_3}$	H_3	0.999	0.999	**0.522**	0.949
$\text{EDA}^8_{3_4}$	H_3	0.998	0.996	**0.958**	0.983
$\text{EDA}^8_{3_5}$	H_3	0.999	**0.922**	0.999	0.996
$\text{EDA}^8_{4_1}$	H_4	0.999	0.999	**0.971**	0.981
$\text{EDA}^8_{4_2}$	H_4	0.999	0.998	0.992	**0.945**
$\text{EDA}^8_{4_3}$	H_4	0.998	0.998	**0.988**	0.996
$\text{EDA}^8_{4_4}$	H_4	0.999	0.999	**0.982**	0.994
$\text{EDA}^8_{4_5}$	H_4	0.999	0.999	0.999	**0.988**

Table E.3: Distance (Equation 11.11) between each *ansatz* in $\text{EDA}^8_{i_j}$ and D^8_i, where i denotes the Hamiltonian index and $n = 8$. Bold values represent those instances in which the closest cluster to $\text{EDA}^5_{i_j}$ is D^8_i.

ansatz ($\mathrm{EDA}_{i_j}^{12}$)	H_i	$\mathrm{dist}(\mathrm{EDA}_{1_j}^{12}, D_1^{12})$	$\mathrm{dist}(\mathrm{EDA}_{2_j}^{12}, D_2^{12})$	$\mathrm{dist}(\mathrm{EDA}_{3_j}^{12}, D_3^{12})$	$\mathrm{dist}(\mathrm{EDA}_{4_j}^{12}, D_4^{12})$
$\mathrm{EDA}_{1_1}^{12}$	H_1	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{1_2}^{12}$	H_1	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{1_3}^{12}$	H_1	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{1_4}^{12}$	H_1	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{1_5}^{12}$	H_1	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{2_1}^{12}$	H_2	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{2_2}^{12}$	H_2	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{2_3}^{12}$	H_2	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{2_4}^{12}$	H_2	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{2_5}^{12}$	H_2	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{3_1}^{12}$	H_3	0.999	0.998	0.999	0.999
$\mathrm{EDA}_{3_2}^{12}$	H_3	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{3_3}^{12}$	H_3	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{3_4}^{12}$	H_3	0.999	0.999	0.998	0.999
$\mathrm{EDA}_{3_5}^{12}$	H_3	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{4_1}^{12}$	H_4	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{4_2}^{12}$	H_4	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{4_3}^{12}$	H_4	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{4_4}^{12}$	H_4	0.999	0.999	0.999	0.999
$\mathrm{EDA}_{4_5}^{12}$	H_4	0.999	0.999	0.999	0.999

Table E.4: Distance (Equation 11.11) between each *ansatz* in $\mathrm{EDA}_{i_j}^{12}$ and D_i^{12}, where i denotes the Hamiltonian index and $n = 12$.

	$n = 4$		$n = 8$	
	E	IC	E	IC
H_1	-8.37 ± 0.01	0.47 ± 0.14	-16.89 ± 0.01	0.46 ± 0.16
H_2	-7.83 ± 0.01	0.51 ± 0.16	-15.92 ± 0.02	0.45 ± 0.06
H_3	-14.19 ± 1.87	0.63 ± 0.15	-30.07 ± 0.01	0.51 ± 0.07
H_4	-17.18 ± 2.20	0.80 ± 0.09	-39.05 ± 0.04	0.82 ± 0.15

Table E.5: Mean and standard deviation of expectation value (E) (Equation 4.9) and information content (IC) (Equation 4.15), respectively, found in the *ansatz* in the dataset whose depth is in the range $m \pm \sqrt{m}$, for different number of qubits n and Hamiltonian H_i.

	$n = 4$		$n = 8$	
	E	IC	E	IC
H_1	-7.81	0.97	-16.18	0.56
H_2	-6.74	0.73	-13.58	0.45
H_3	-14.03	1.00	-29.28	0.43
H_4	-17.21	1.47	-26.87	1.57

Table E.6: Best expectation value (E) (Equation 4.9) and information content (IC) (Equation 4.15) found by the EDA approach (assisted by COBYLA) for different number of qubits (n) and Hamiltonians (H_i), where HV is maximized in the best Pareto approximation.

Appendix F

Library Required Metadata

F.1 Current executable software version

See Table F.1.

N	Software metadata description	Software metadata information
S1	Current software version	1.1.4
S2	Permanent link to executables of this version	https://github.com/VicentePerezSoloviev/EDAspy/releases/tag/1.1.3
S3	Legal Software License	MIT License
S4	Computing platform/Operating System	Linux, OS X, Windows
S5	Installation requirements & dependencies	python 3.8–3.11, pybnesian, numpy, pandas, scikit_learn, scipy, pgmpy pyarrow, multiprocessing
S6	Link to user manual	https://edaspy.readthedocs.io/en/latest/
S7	Support email for questions	vicente.perez.soloviev@gmail.com

Table F.1: Software metadata.

F.2 Current code version

See Table F.2.

N	Software metadata description	Software metadata information
C1	Current code version	1.1.4
C2	Permanent link to code/repository used of this code version	https://github.com/VicentePerezSoloviev/EDAspy
C3	Legal Software License	MIT License
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	python 3.8–3.11
C6	Compilation requirements, operating environments & dependencies	Compatible `python` pybnesian, numpy, pandas, scikit_learn, scipy, pgmpy, pyarrow, multiprocessing
C7	Link to developer documentation/manual	https://edaspy.readthedocs.io/en/latest/
C8	Support email for questions	vicente.perez.soloviev@gmail.com

Table F.2: Code metadata.

Part VI

REFERENCES

Bibliography

C. W. Ahn and R. S. Ramakrishna. Multiobjective real-coded bayesian optimization algorithmrevisited: diversity preservation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 593–600, 2007.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte. Parameter concentrations in quantum approximate optimization. *Physical Review A*, 104(1):L010401, 2021.

H. Alibrahim and S. A. Ludwig. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In *2021 IEEE Congress on Evolutionary Computation*, pages 1551–1559. IEEE, 2021.

A. Anand, M. Degroote, and A. Aspuru-Guzik. Natural evolutionary strategies for variational quantum computation. *Machine Learning: Science and Technology*, 2(4):045012, 2021.

E. R. Anschuetz and B. T. Kiani. Quantum variational algorithms are swamped with traps. *Nature Communications*, 13(1):7760, 2022.

S. Aouay, S. Jamoussi, and Y. B. Ayed. Particle swarm optimization based method for Bayesian network structure learning. In *5th International Conference on Modeling, Simulation and Applied Optimization*, pages 1–6. IEEE, 2013.

D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Frontiers in Physics*, 7:48, 2019.

D. Atienza, C. Bielza, and P. Larrañaga. PyBNesian: An extensible Python package for Bayesian networks. *Neurocomputing*, 504:204–209, 2022a.

D. Atienza, C. Bielza, and P. Larrañaga. Semiparametric Bayesian networks. *Information Sciences*, 584:564–582, 2022b.

ATOS. Quantum Learning Machine. https://atos.net/en/solutions/quantum-learning-machine, 2021. [Online; accessed 26-January-2022].

S. Baluja. *Population-based Incremental Learning: A Method for Integrating Genetic Search*

based Function Optimization and Competitive Learning. School of Computer Science, Carnegie Mellon University Pittsburgh, PA, 1994.

S. Baluja and S. Davies. *Combining Multiple Optimization Runs with Optimal Dependency Trees*. Carnegie-Mellon University. Department of Computer Science, 1997.

P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner. Improving variational quantum optimization using CVaR. *Quantum*, 4:256, 2020.

J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, pages 115–123, 2013.

N. Beume, C. M. Fonseca, M. Lopez-Ibanez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

K. Bharti, A. Cervera-Lierta, T. H. Kyaw, et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, 94(1):015004, 2022.

C. Bielza and P. Larrañaga. Bayesian networks in neuroscience: A survey. *Frontiers in Computational Neuroscience*, 8:131, 2014.

R. Blanco, I. Inza, and P. Larrañaga. Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, 18 (2):205–220, 2003.

J. Blank and K. Deb. PYMOO: Multi-objective optimization in Python. *IEEE Access*, 8: 89497–89509, 2020.

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

X. Bonet-Monroig, H. Wang, D. Vermetten, B. Senjean, C. Moussa, T. Bäck, V. Dunjko, and T. E. O'Brien. Performance comparison of optimization methods on variational quantum algorithms. *arXiv:2111.13454*, 2021.

P. A. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.

P. A. N. Bosman and D. Thierens. Expanding from Discrete to Continuous Estimation of Distribution Algorithms: The IDEA. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 767–776. Springer, 2000.

F. G. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven. For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances. *arXiv:1812.04170*, 2018.

A. E. I. Brownlee. *Multivariate Markov Networks for Fitness Modelling in an Estimation of Distribution Algorithm.* PhD thesis, University of Stirling, 2009.

W. Buntine. Theory refinement on Bayesian networks. In *Uncertainty Proceedings 1991*, pages 52–60. Elsevier, 1991.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

B. Calvo, O. M. Shir, J. Ceberio, C. Doerr, H. Wang, T. Bäck, and J. A. Lozano. Bayesian performance analysis for black-box optimization benchmarking. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1789–1797, 2019.

J. Ceberio, B. Doerr, C. Witt, and V. P. Soloviev. Estimation-of-Distribution Algorithms: Theory and Applications (Dagstuhl Seminar 22182). In *Dagstuhl Reports*, volume 12. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022a.

J. Ceberio, A. Mendiburu, and J. A. Lozano. A roadmap for solving optimization problems with estimation of distribution algorithms. *Natural Computing*, pages 1–15, 2022b.

M. Cerezo, A. Arrasmith, R. Babbush, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021a.

M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1):1791, 2021b.

C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data*, pages 121–130. Springer, 1996.

D. Chivilikhin, A. Samarin, V. Ulyantsev, I. Iorsh, A. Oganov, and O. Kyriienko. MoG-VQE: Multiobjective genetic variational quantum eigensolver. *arXiv:2007.04424*, 2020.

C. K. Chow and S. Y. Yuen. An evolutionary algorithm that makes decision based on the entire previous search history. *IEEE Transactions on Evolutionary Computation*, 15(6): 741–769, 2011.

M. A. Coletti, E. O. Scott, and J. K. Bassett. Library for evolutionary algorithms in Python (LEAP). In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1571–1579, 2020.

G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

M. Costa and E. Minisci. Moped: a multi-objective parzen-based estimation of distribution algorithm for continuous problems. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 282–294. Springer, 2003.

N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of an International Conference on Genetic Algorithms and the Applications*, pages 183–187, 1985.

M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys*, 45(3):1–33, 2013.

D. Cruz, R. Fournier, F. Gremion, A. Jeannerot, K. Komagata, T. Tosic, J. Thiesbrummel, C. L. Chan, N. Macris, M.-A. Dupertuis, et al. Efficient quantum algorithms for GHZ and W states, and implementation on the IBM Quantum computer. *Advanced Quantum Technologies*, 2(5-6):1970031, 2019.

L. R. da Silveira, R. Tanscheit, and M. M. Vellasco. Quantum inspired evolutionary algorithm for ordering problems. *Expert Systems with Applications*, 67:71–83, 2017.

P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

D. Dasgupta and Z. Michalewicz. *Evolutionary Algorithms in Engineering Applications*. Springer, 2014.

S. Dasgupta. Learning mixtures of Gaussians. In *40th Annual Symposium on Foundations of Computer Science*, pages 634–644. IEEE, 1999.

J. S. De Bonet, C. L. Isbell Jr, and P. A. Viola. MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, pages 424–430, 1997.

A. De Lorenzo, E. Medvet, T. Tušar, and A. Bartoli. An analysis of dimensionality reduction techniques for visualizing evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1864–1872, 2019.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

J. Deutch. Is net zero carbon 2050 possible? *Joule*, 4(11):2237–2240, 2020.

B. Doerr and W. Zheng. Sharp bounds for genetic drift in estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 24(6):1140–1149, 2020.

Y. Du, T. Huang, S. You, M.-H. Hsieh, and D. Tao. Quantum circuit architecture search: Error mitigation and trainability enhancement for variational quantum solvers. *arXiv:2010.10217*, 2020.

D. J. Egger, J. Mareček, and S. Woerner. Warm-starting quantum optimization. *Quantum*, 5: 479, 2021.

T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

L. Eshelman. On crossover as an evolutionarily viable strategy. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann Publishers San Francisco, 1991.

R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In *Proc. 2nd Symposium on Artificial Intelligence (CIMAF-99)*, pages 332–339, 1999.

J. G. Falcón-Cardona, R. H. Gómez, C. A. C. Coello, and M. G. C. Tapia. Parallel multi-objective evolutionary algorithms: A comprehensive survey. *Swarm and Evolutionary Computation*, 67:100960, 2021.

H. Fang, A. Zhou, and G. Zhang. An estimation of distribution algorithm guided by mean shift. In *IEEE Congress on Evolutionary Computation*, pages 3268–3275. IEEE, 2016.

K.-T. Fang, R. Li, and A. Sudjianto. *Design and Modeling for Computer Experiments*. CRC Press, 2005.

E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2014.

L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, 1966.

E. Fontana, N. Fitzpatrick, D. M. Ramo, R. Duncan, and I. Rungger. Evaluating the noise resilience of variational quantum algorithms. *Physical Review A*, 104(2):022403, 2021.

T. Fösel, M. Y. Niu, F. Marquardt, and L. Li. Quantum circuit optimization with deep reinforcement learning. *arXiv:2103.07585*, 2021.

J. Fox. *Applied Regression Analysis, Linear Models, and Related Methods.* Sage Publications, Inc, 1997.

N. Friedman and I. Nachman. Gaussian process networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, page 211–219. Morgan Kaufmann Publishers, 2000.

M. Fyvie, J. A. McCall, L. A. Christie, A. E. Brownlee, and M. Singh. Towards explainable metaheuristics: Feature extraction from trajectory mining. *Expert Systems*, page e13494, 2023.

M. Gallagher, M. R. Frean, and T. Downs. Real-valued evolutionary optimization using a flexible probability density estimator. In *Proceedings of the conference on Genetic and evolutionary computation*, volume 99, pages 840–846, 1999.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1):106–148, 2011.

B. Gao and I. Wood. TAM-EDA: Multivariate t distribution, archive and mutation based estimation of distribution algorithm. *ANZIAM Journal*, 54:C720–C746, 2012.

A. Garcia-Saez and J. Riu. Quantum observables for continuous control of the quantum approximate optimization algorithm via reinforcement learning. *arXiv:1911.09682*, 2019.

J. Grahl, P. A. Bosman, and F. Rothlauf. The correlation-triggered adaptive variance scaling IDEA. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 397–404, 2006.

S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh. Bayesian optimization for adaptive experimental design: A review. *IEEE Access*, 8:13937–13948, 2020.

H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 10(1):3007, 2019.

L. Gyongyosi. Quantum state optimization and computational pathway evaluation for gate-model quantum computers. *Scientific Reports*, 10(1):1–12, 2020.

L. Gyongyosi and S. Imre. A survey on quantum computing technology. *Computer Science Review*, 31:51–71, 2019.

J. A. Gámez, J. Mateo, and J. M. Puerta. Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22:106–148, 2011.

S. K. Hadia, A. H. Joshi, C. K. Patel, and Y. P. Kosta. Solving city routing issue with particle swarm optimization. *International Journal of Computer Applications*, 47(15), 2012.

K.-H. Han and J.-H. Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6(6):580–593, 2002.

K. Hanaoka. Bayesian optimization for goal-oriented multi-objective inverse material design. *iScience*, 24(7):102781, 2021.

K. Hanaoka. Comparison of conceptually different multi-objective Bayesian optimization methods for material design problems. *Materials Today Communications*, 31:103440, 2022.

N. Hansen. The CMA evolution strategy: A comparing review. *Towards a New Evolutionary Computation*, pages 75–102, 2006.

G. Harik. Linkage learning via probabilistic modeling in the ECGA (IlliGAL Report No. 99010). *Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory*, 1999.

G. R. Harik, F. G. Lobo, and D. E. Goldberg. The Compact Genetic Algorithm. *IEEE Transactions of Evolutionary Computation*, 3(4):287–297, 1999.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver. Perspectives of quantum annealing: Methods and implementations. *Reports on Progress in Physics*, 83(5): 054401, 2020.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Uncertainty in Artificial Intelligence*, volume 5, pages 149–163. Elsevier, 1988.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–435, 1952.

J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, 1975.

W.-J. Hong, P. Yang, and K. Tang. Evolutionary computation for large-scale multi-objective optimization: A decade of progresses. *International Journal of Automation and Computing*, 18(2):155–169, 2021.

K. Ickstadt, B. Bornkamp, M. Grzegorczyk, J. Wieczorek, M. R. Sheriff, H. E. Grecco, and E. Zamir. Nonparametric Bayesian networks. *Bayesian Statistics*, 9:283–316, 2010.

H. Ishibuchi, R. Imada, N. Masuyama, and Y. Nojima. Dynamic specification of a reference point for hypervolume calculation in SMS-EMOA. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2018.

A. Jankovic, G. Chaudhary, and F. Goia. Designing the design of experiments (DOE)–An investigation on the influence of different factorial designs on the characterization of complex systems. *Energy and Buildings*, 250:111298, 2021.

J.-Z. Ji, H.-X. Zhang, R.-B. Hu, and C.-N. Liu. A tabu-search based Bayesian network structure learning algorithm. *Journal of Beijing University of Technology*, 37:1274–1280, 2011.

D. Kalpić, N. Hlupić, and M. Lovrić. Student's t-tests. *International Encyclopedia of Statistical Science*, 2011.

A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, 567(7749):491–495, 2019.

H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Multiobjective estimation of distribution algorithm based on joint modeling of objectives and variables. *IEEE Transactions on Evolutionary Computation*, 18(4):519–542, 2013.

J. Kaufmann and A. Schering. Analysis of variance ANOVA. *Wiley Encyclopedia of Clinical Trials*, 2007.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

N. Khan, D. E. Goldberg, and M. Pelikan. Multi-objective Bayesian optimization algorithm. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 684–684, 2002.

T. D. Kieu. The travelling salesman problem and adiabatic quantum computation: An algorithm. *Quantum Information Processing*, 18(3):90, 2019.

D. P. Kingma and J. Ba. ADAM: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

N. K. Kitson, A. C. Constantinou, Z. Guo, Y. Liu, and K. Chobtham. A survey of Bayesian Network structure learning. *Artificial Intelligence Review*, 56(8):8721–8814, 2023.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT press, 2009.

M. S. Krejca and C. Witt. Lower bounds on the run time of the univariate marginal distribution algorithm on onemax. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, pages 65–79, 2017.

H. Langseth, T. D. Nielsen, R. Rumı, and A. Salmerón. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53(2):212–227, 2012.

P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana. A review on probabilistic graphical models in evolutionary computation. *Journal of Heuristics*, 18(5):795–819, 2012.

P. Larrañaga and C. Bielza. Estimation of distribution algorithms in machine learning: A survey. *IEEE Transactions on Evolutionary Computation*, 2024.

P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization by learning and simulation of Bayesian and Gaussian networks. *Technical Report, Department of Computer Science and Artificial Intelligence, University of the Basque Country*, 1999.

P. Larrañaga, M. Poza, Y. Yurramendi, R. H. Murga, and C. M. H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9): 912–926, 1996.

P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. *Proceedings Genetic and Evolutionary Computation Congress*, 2000.

H.-K. Lau, R. Pooser, G. Siopsis, and C. Weedbrook. Quantum machine learning over infinite dimensions. *Physical Review Letters*, 118(8):080501, 2017.

M. Laumanns and J. Ocenasek. Bayesian optimization algorithms for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 298–307. Springer, 2002.

S. Lee and S. B. Kim. Parallel simulated annealing with a greedy algorithm for Bayesian network structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(6): 1157–1166, 2019.

M. Li and X. Yao. Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys*, 52(2):1–38, 2019.

J. J. Liang, B. Y. Qu, and P. N. Suganthan. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Technical Report, Nanyang Technological University, Singapore*, 635:490–523, 2013.

Y. Liang, Z. Ren, X. Yao, Z. Feng, A. Chen, and W. Guo. Enhancing Gaussian estimation of distribution algorithm by exploiting evolution direction with archive. *IEEE Transactions on Cybernetics*, 50(1):140–152, 2018.

K. Linghu, Y. Qian, R. Wang, M.-J. Hu, Z. Li, X. Li, H. Xu, J. Zhang, T. Ma, P. Zhao, et al. Quantum circuit architecture search on a superconducting processor. *arXiv:2201.00934*, 2022.

H. Liu, Y.-S. Ong, and J. Cai. A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. *Structural and Multidisciplinary Optimization*, 57:393–416, 2018a.

H. Liu, K. Simonyan, and Y. Yang. Darts: Diferentiable architecture search. *arXiv:1806.09055*, 2018b.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, 1982.

Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf. NSGA-NET: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 419–427, 2019.

N. Luo and F. Qian. Evolutionary algorithm using kernel density estimation model in continuous domain. In *7th Asian Control Conference*, pages 1526–1531. IEEE, 2009.

L. Martí, J. García, A. Berlanga, and J. M. Molina. Multi-objective optimization with an adaptive resonance theory-based estimation of distribution algorithm. *Annals of Mathematics and Artificial Intelligence*, 68:247–273, 2013.

L. Martí, J. García, A. Berlanga, and J. M. Molina. Moneda: scalable multi-objective optimization with a neural network-based estimation of distribution algorithm. *Journal of Global Optimization*, 66:729–768, 2016.

R. A. Martin. PyPortfolioOpt: Portfolio optimization in Python. *Journal of Open Source Software*, 6(61):3066, 2021.

J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):1–6, 2018.

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, pages 55–61, 2000.

M. McKerns and M. Aivazis. Pathos: A framework for heterogeneous computing. *http://trac. mystic. cacr. caltech. edu/project/pathos*, 2010.

M. M. McKerns, L. Strand, T. Sullivan, A. Fang, and M. A. Aivazis. Building a framework for predictive science. *arXiv:1202.1056*, 2012.

J.-A. Mejía-de Dios, A. Rodríguez-Molina, and E. Mezura-Montes. Multiobjective bilevel optimization: A survey of the state-of-the-art. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

B. Mihaljević, C. Bielza, and P. Larrañaga. Bayesian networks for interpretable machine learning and optimization. *Neurocomputing*, 456:648–665, 2021.

S. Miki, D. Yamamoto, and H. Ebara. Applying deep learning and reinforcement learning to traveling salesman problem. In *2018 International Conference on Computing, Electronics & Communications Engineering*, pages 65–70. IEEE, 2018.

O. Montiel, Y. Rubio, C. Olvera, and A. Rivera. Quantum-inspired acromyrmex evolutionary algorithm. *Scientific Reports*, 9(1):1–10, 2019.

S. Moral, R. Rumí, and A. Salmerón. Mixtures of truncated exponentials in hybrid Bayesian networks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 156–167. Springer, 2001.

H. Mühlenbein and T. Mahnig. Fda-a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary computation*, 7(4):353–376, 1999.

H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distributions I. Binary parameters. In *International Conference on Parallel Problem Solving from Nature*, pages 178–187. Springer, 1996.

H. Mühlenbein, J. Bendisch, and H.-M. Voigt. From recombination of genes to the estimation of distributions II. Continuous parameters. In *International Conference on Parallel Problem Solving from Nature*, pages 188–197. Springer, 1996.

K. P. Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley & Sons, 2016.

A. Nakayama, K. Mitarai, L. Placidi, T. Sugimoto, and K. Fujii. VQE-generated quantum circuit dataset for machine learning, 2023.

G. Neumann and D. Cairns. Introducing intervention targeting into estimation of distribution algorithms. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 220–225, 2012.

T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

M. A. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2002.

M. Ostaszewski, E. Grant, and M. Benedetti. Structure optimization for parameterized quantum circuits. *Quantum*, 5:391, 2021a.

M. Ostaszewski, L. M. Trenkwalder, W. Masarczyk, E. Scerri, and V. Dunjko. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, 34:18182–18194, 2021b.

B. O'Gorman, R. Babbush, A. Perdomo-Ortiz, A. Aspuru-Guzik, and V. Smelyanskiy. Bayesian network structure learning using quantum annealing. *The European Physical Journal Special Topics*, 224(1):163–188, 2015.

T. K. Paul and H. Iba. Reinforcement learning estimation of distribution algorithm. In *Genetic and Evolutionary Computation Conference*, pages 1259–1270. Springer, 2003.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

M. Pelikan and D. E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In *International Conference on Parallel Problem Solving from Nature*, pages 385–394. Springer, 2000.

M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In *Advances in Soft Computing: Engineering Design and Manufacturing*, pages 521–535. Springer, 1999.

M. Pelikan, D. E. Goldberg, E. Cantú-Paz, et al. BOA: The Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 525–532, 1999.

M. Pelikan, K. Sastry, and D. E. Goldberg. Multiobjective hBOA, clustering, and scalability. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 663–670, 2005.

J. M. Peña, J. A. Lozano, and P. Larrañaga. Benefits of data clustering in multimodal function optimization via EDAs. In *Estimation of Distribution Algorithms*, pages 101–127. Springer, 2002.

A. Pérez, P. Larrañaga, and I. Inza. Bayesian classifiers based on kernel density estimation: Flexible classifiers. *International Journal of Approximate Reasoning*, 50(2):341–362, 2009.

A. Pérez-Salinas, H. Wang, and X. Bonet-Monroig. Analyzing variational quantum landscapes with information content. *arXiv:2303.16893*, 2023.

A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):1–7, 2014.

M. Pirhooshyaran and T. Terlaky. Quantum circuit design search. *Quantum Machine Intelligence*, 3:1–14, 2021.

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

M. D. Platel, S. Schliebs, and N. Kasabov. Quantum-inspired evolutionary algorithm: A multimodel EDA. *IEEE Transactions on Evolutionary Computation*, 13(6):1218–1232, 2008.

P. Pošík. Preventing premature convergence in a simple EDA via global step size setting. In *International Conference on Parallel Problem Solving from Nature*, pages 549–558. Springer, 2008.

M. J. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer, 1994.

M. J. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7: 287–336, 1998.

M. Probst. Generative adversarial networks in estimation of distribution algorithms for combinatorial optimization. *arXiv:1509.09235*, 2015.

M. Probst and F. Rothlauf. Harmless overfitting: Using denoising autoencoders in estimation of distribution algorithms. *Journal of Machine Learning Research*, 21(78):1–31, 2020.

C. Puerto-Santana, P. Larrañaga, and C. Bielza. Autoregressive asymmetric linear Gaussian hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.

D. Quesada, C. Bielza, and P. Larrañaga. Structure learning of high-order dynamic Bayesian networks via particle swarm optimization with order invariant encoding. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 158–171. Springer, 2021.

M. Ragone, B. N. Bakalov, F. Sauvage, A. F. Kemper, C. O. Marrero, M. Larocca, and M. Cerezo. A unified theory of barren plateaus for deep parametrized quantum circuits. *arXiv:2309.09342*, 2023.

P. Rakshit, A. Konar, and S. Das. Noisy evolutionary optimization algorithms–a comprehensive survey. *Swarm and Evolutionary Computation*, 33:18–45, 2017.

A. G. Rattew, S. Hu, M. Pistoia, R. Chen, and S. Wood. A domain-agnostic, noise-resistant, hardware-efficient evolutionary variational quantum eigensolver. *arXiv:1910.09694*, 2019.

I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, University of Cambridge, 1971.

R. W. Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V*, pages 28–43. Springer, 1977.

V. Robles, P. De Miguel, and P. Larrañaga. Solving the traveling salesman problem with EDAs. In *Estimation of Distribution Algorithms*, pages 211–229. Springer, 2002.

O. H. M. Ross. A review of quantum-inspired metaheuristics: Going from classical computers to real quantum computers. *IEEE Access*, 8:814–838, 2019.

Y. Ruan, S. Marsh, X. Xue, Z. Liu, and J. Wang. The quantum approximate algorithm for solving traveling salesman problem. *Computers, Materials and Continua*, 63(3):1237–1247, 2020.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

S. Rudlof and M. Koppen. Stochastic hill climbing by vectors of normal distributions. *Proceedings of the First Online Workshop on Soft Computing*, 1996.

R. Santana, C. Bielza, P. Larranaga, J. A. Lozano, C. Echegoyen, A. Mendiburu, R. Armananzas, and S. Shakya. Mateda-2.0: A MATLAB package for the implementation and analysis of estimation of distribution algorithms. *Journal of Statistical Software*, 35:1–30, 2010.

M. Scanagatta, A. Salmerón, and F. Stella. A survey on Bayesian network structure learning from data. *Progress in Artificial Intelligence*, 8(4):425–439, 2019.

M. Schuld. *Supervised Learning with Quantum Computers*. Springer, 2018.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, pages 461–464, 1978.

J. Schwarz and J. Ocenasek. Multiobjective bayesian optimization algorithm for combinatorial problems: Theory and practice. *Neural Network World*, 11(5):423–442, 2001.

D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

K. C. Seto, G. Churkina, A. Hsu, M. Keller, P. W. Newman, B. Qin, and A. Ramaswami. From low-to net-zero carbon cities: The next global agenda. *Annual Review of Environment and Resources*, 46:377–415, 2021.

K. Sharma, S. Khatri, M. Cerezo, and P. J. Coles. Noise resilience of variational quantum compiling. *New Journal of Physics*, 22(4):043006, 2020.

W. F. Sharpe. The Sharpe ratio. *The Journal of Portfolio Management*, 21:49–58, 1994.

R. Shaydulin and Y. Alexeev. Evaluating quantum approximate optimization algorithm: A

case study. In *Tenth International Green and Sustainable Computing Conference*, pages 1–6. IEEE, 2019.

P. P. Shenoy and J. C. West. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657, 2011.

R. Shi, J. Luo, and Q. Liu. Fast evolutionary neural architecture search based on Bayesian surrogate model. In *IEEE Congress on Evolutionary Computation*, pages 1217–1224. IEEE, 2021.

Y. Shikuri. Efficient conversion of Bayesian network learning into quadratic unconstrained binary optimization. *arXiv:2006.06926*, 2020.

V. A. Shim, K. C. Tan, and K. K. Tan. A hybrid adaptive evolutionary algorithm in the domination-based and decomposition-based frameworks of multi-objective optimization. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

V. A. Shim, K. C. Tan, C. Y. Cheong, and J. Y. Chia. Enhancing the scalability of multi-objective optimization via restricted boltzmann machine-based estimation of distribution algorithm. *Information Sciences*, 248:191–213, 2013.

W. B. Shine and C. F. Eick. Visualizing the evolution of genetic algorithm search processes. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 367–372. IEEE, 1997.

L. R. Silveira, R. Tanscheit, and M. Vellasco. Quantum-inspired genetic algorithms applied to ordering combinatorial optimization problems. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2012.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 2018.

H. Soh and M. Kirley. mopga: Towards a new generation of multi-objective genetic algorithms. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1702–1709. IEEE, 2006.

V. P. Soloviev, C. Bielza, and P. Larrañaga. Quantum-inspired estimation of distribution algorithm to solve the travelling salesman problem. In *IEEE Congress on Evolutionary Computation*, pages 416–425. IEEE, 2021.

V. P. Soloviev, C. Bielza, and P. Larrañaga. Quantum approximate optimization algorithm for bayesian network structure learning. *arXiv:2203.02400*, 2022a.

V. P. Soloviev, P. Larrañaga, and C. Bielza. Estimation of distribution algorithms using Gaussian Bayesian networks to solve industrial optimization problems constrained by environment variables. *Journal of Combinatorial Optimization*, 44(2):1077–1098, 2022b.

V. P. Soloviev, P. Larrañaga, and C. Bielza. Quantum parametric circuit optimization with estimation of distribution algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2247–2250, 2022c.

V. P. Soloviev, C. Bielza, and P. Larrañaga. Semiparametric estimation of distribution

algorithms for continuous optimization. *IEEE Transactions on Evolutionary Computation*, 2023a.

V. P. Soloviev, P. Larrañaga, and C. Bielza. Variational quantum algorithm parameter tuning with estimation of distribution algorithms. In *IEEE Congress on Evolutionary Computation*, pages 1–9. IEEE, 2023b.

V. P. Soloviev, V. Dunjko, C. Bielza, P. Larrañaga, and H. Wang. Trainability maximization using estimation of distribution algorithms assisted by surrogate modelling for quantum architecture search. *EPJ Quantum Technology*, 11(1):69, 2024a.

V. P. Soloviev, P. Larrañaga, and C. Bielza. EDAspy: An extensible Python package for estimation of distribution algorithms. *Neurocomputing*, page 128043, 2024b.

V. P. Soloviev, P. Larrañaga, M. Bernabei, M. Bernabei, M. A. Chirita, J. M. Seoane, P. Fontán, and C. Bielza. A multi-objective framework based on estimation of distribution algorithms for data-driven fuel experimental design. *(in review)*, 2024c.

M. Soto, Y. González-Fernández, and A. Ochoa. Modeling with copulas and vines in estimation of distribution algorithms. *arXiv:1210.5500*, 2012.

J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, 2000.

R. Storn. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report, International Computer Science Institute*, 11, 1995.

M. Streif and M. Leib. Comparison of QAOA with quantum and simulated annealing. *arXiv:1901.01903*, 2019.

J. Sun, X. Yuan, T. Tsunoda, V. Vedral, S. C. Benjamin, and S. Endo. Mitigating realistic noise in practical noisy intermediate-scale quantum devices. *Physical Review Applied*, 15 (3):034026, 2021.

X. Sun, Z. Shi, G. Lei, Y. Guo, and J. Zhu. Multi-objective design optimization of an IPMSM based on multilevel strategy. *IEEE Transactions on Industrial Electronics*, 68(1):139–148, 2020.

L. Sünkel, D. Martyniuk, D. Mattern, J. Jung, and A. Paschke. GA4QCO: Genetic algorithm for quantum circuit optimization. *arXiv:2302.01303*, 2023.

R. Tanabe and A. Fukunaga. Success-history based parameter adaptation for differential evolution. In *IEEE Congress on Evolutionary Computation*, pages 71–78. IEEE, 2013.

R. Tanabe and A. S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation*, pages 1658–1665. IEEE, 2014.

B. Thiesson, C. Meek, D. M. Chickering, and D. Heckerman. Learning mixtures of DAG models. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, page 504–513. Morgan Kaufmann Publishers, 1998.

A. Tonda. Inspyred: Bio-inspired algorithms in Python. *Genetic Programming and Evolvable Machines*, 21(1-2):269–272, 2020.

I. Tsamardinos and C. F. Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In *International Workshop on Artificial Intelligence and Statistics*, pages 300–307, 2003.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 673–678, 2003.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

S. Tsutsui, M. Pelikan, D. E. Goldberg, et al. Evolutionary algorithm using marginal histogram models in continuous domain. *IlliGAL Report*, 2001019(999):1050, 2001.

M. Urbanek, B. Nachman, V. R. Pascuzzi, A. He, C. W. Bauer, and W. A. de Jong. Mitigating depolarizing noise on quantum computers with noise-estimation circuits. *arXiv:2103.08591*, 2021.

L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

F. A. Viana. Things you wanted to know about the Latin hypercube design and were afraid to ask. In *10th World Congress on Structural and Multidisciplinary Optimization*, volume 19, pages 1–9, 2013.

H. Vieira Jr, S. M. Sanchez, K. H. Kienitz, and M. C. N. Belderrain. Efficient, nearly orthogonal-and-balanced, mixed designs: An effective way to conduct trade-off analyses via simulation. *Journal of Simulation*, 7(4):264–275, 2013.

J. Vovrosh, K. E. Khosla, S. Greenaway, C. Self, M. Kim, and J. Knolle. Simple mitigation of global depolarizing errors in quantum simulations. *Physical Review E*, 104(3):035309, 2021.

S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12 (1):6961, 2021.

S.-C. Wang, R. Gao, and L.-M. Wang. Bayesian network classifiers based on Gaussian kernel density. *Expert Systems with Applications*, 51:207–217, 2016.

Z. Wang, M. Li, and J. Li. A multi-objective evolutionary algorithm for feature selection based on mutual information with a new redundancy measure. *Information Sciences*, 307: 73–88, 2015.

N. Wiebe, D. Braun, and S. Lloyd. Quantum algorithm for data fitting. *Physical Review Letters*, 109(5):050505, 2012.

G. Wu, R. Mallipeddi, and P. N. Suganthan. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. *Technical Report, Nanyang Technological University, Singapore*, 2017.

W. Wu, G. Yan, X. Lu, K. Pan, and J. Yan. QuantumDARTS: Differentiable quantum architecture search for variational quantum algorithms. pages 37745–37764, 2023.

C. Xue, Z.-Y. Chen, Y.-C. Wu, and G.-P. Guo. Effects of quantum noise on quantum approximate optimization algorithm. *Chinese Physics Letters*, 38(3):030302, 2021.

H. Yu. Optimized ant colony algorithm by local pheromone update. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 12, 2014.

B. Yuan and M. Gallagher. On the importance of diversity maintenance in estimation of distribution algorithms. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 719–726, 2005.

G. Zhang. Quantum-inspired evolutionary algorithms: A survey and empirical study. *Journal of Heuristics*, 17(3):303–351, 2011.

J. Zhang and A. C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.

Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2009.

R. Zhang, A. Prokhorchuk, and J. Dauwels. Deep reinforcement learning for traveling salesman problem with time windows and rejections. In *International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2020.

S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao. Differentiable quantum architecture search. *Quantum Science and Technology*, 7(4):045023, 2022.

W. Zheng and B. Doerr. From understanding genetic drift to a smart-restart mechanism for estimation-of-distribution algorithms. *Journal of Machine Learning Research*, 24(292):1–40, 2023.

X. Zhong and W. Li. A decision-tree-based multi-objective estimation of distribution algorithm. In *2007 International Conference on Computational Intelligence and Security (CIS 2007)*, pages 114–11. IEEE, 2007.