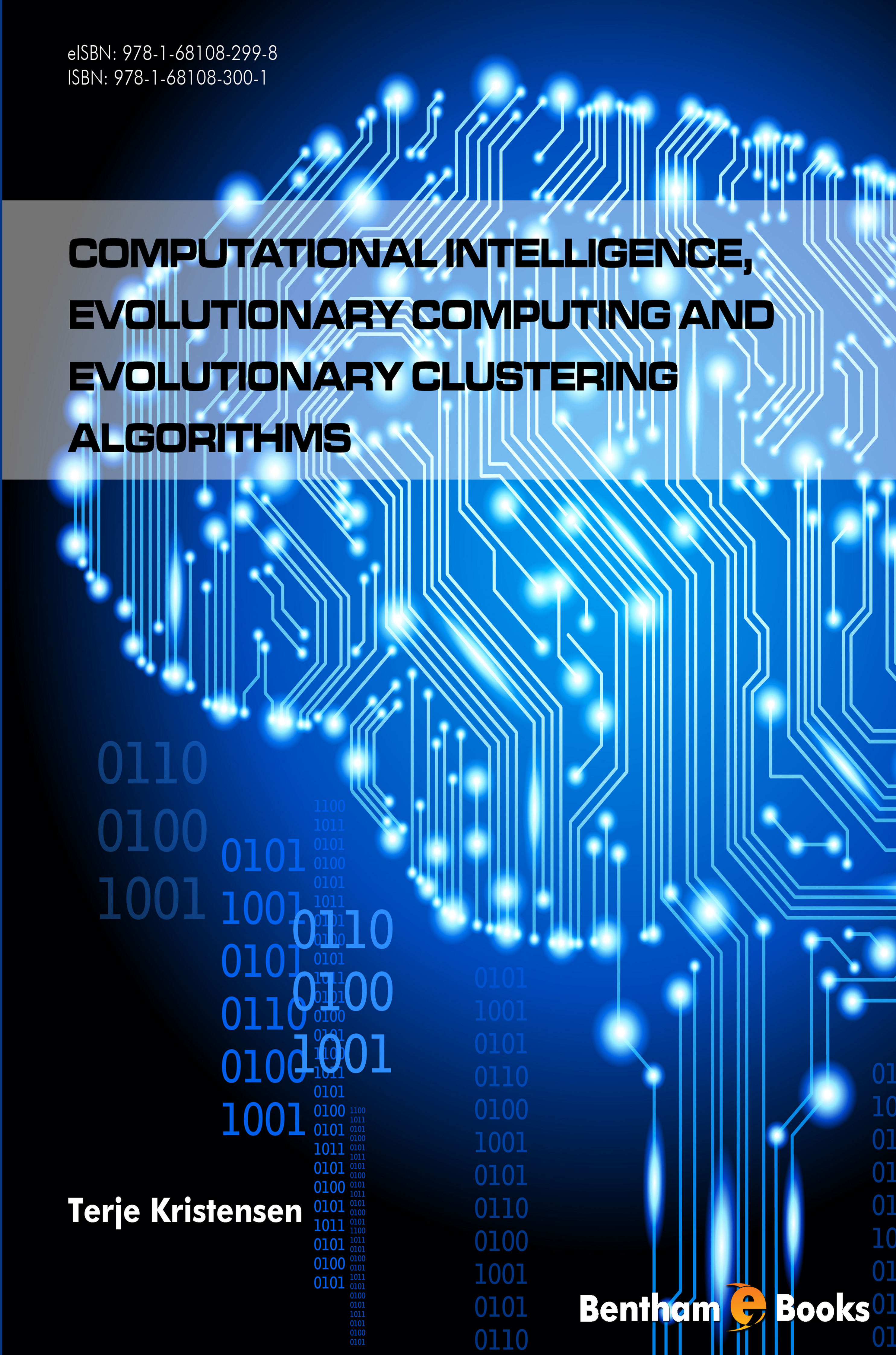
eISBN: 978-1-68108-299-8
ISBN: 978-1-68108-300-1

COMPUTATIONAL INTELLIGENCE, EVOLUTIONARY COMPUTING AND EVOLUTIONARY CLUSTERING ALGORITHMS



Terje Kristensen

Bentham e Books

COMPUTATIONAL INTELLIGENCE, EVOLUTIONARY COMPUTING AND EVOLUTIONARY CLUSTERING ALGORITHMS
Terje Kristensen
eISBN: 978-1-68108-299-8
ISBN: 978-1-68108-300-1

Computational Intelligence, Evolutionary Computing and Evolutionary Clustering Algorithms

Authored By

Terje Kristensen

CEO and Founder of Pattern Solutions ltd.

and

Bergen University College

Bergen

Norway

Computational Intelligence, Evolutionary Computing and Evolutionary Clustering Algorithms

Author: Terje Kristensen

ISBN (eBook): 978-1-68108-299-8

ISBN (Print): 978-1-68108-300-1 © 2016, Bentham eBooks imprint.

Published by Bentham Science Publishers – Sharjah, UAE. All Rights Reserved.

First published in 2016.

CONTENTS

“*In memory of my mother, Anna, for teaching me never to give up.*”

PREFACE

This book is about how to use new algorithm models to solve complex problems. The book presents one branch of a field in computer science that we today call computational intelligence. Big Data play already a great role in society and evolutionary algorithms may be one approach to do data mining of high-dimensional data. High-dimensional data are produced in scientific laboratories all over the world and are often difficult to interpret. Clustering is one possible technique that may help us to interpret these data.

Clustering is a well-known technique that is used in many areas of science. This book is about how to use such algorithms to solve clustering of huge sets of data. This subject may be introduced in the last year at the bachelor level in computer science or mathematics or at the graduate level. The intention of the book is to show how to use such computation models on classical clustering problems. Visualization is an important part of the clustering process. We therefore also want to visualize the result of the cluster analysis.

The most known clustering algorithm is the K-means algorithm that is dependent on the parameter value K and the initial position of K cluster centroids. An incorrect value will result in an inaccurate clustering structure. The configuration of cluster centroids determines if the algorithm converges to a local minimum or not. These limitations may be solved by using evolutionary algorithms.

Genetic algorithms and differential evolution algorithms are two paradigms in the book that are used to optimize the value of K and the initial configuration of cluster centroids. The correctness and quality of the solution are compared using both artificial and real-life data sets. Experiments have shown that the algorithms are able to classify the correct number of well-defined clusters, but fail to do so for overlapping data clusters. This is mainly because the Davies-Bouldin Index as a fitness measure has certain kind of limitations. The experiments carried out in the book also show that both Genetic and Differential evolution algorithms provide suboptimal positions of initial configuration of cluster centroids, reflected in higher values of the Davies-Bouldin Index.

ACKNOWLEDGEMENTS

I will thank Bergen University College for making it possible to write this book. I also thank Bentham Science Publishers for all help and specifically Manager for Publications, Salma Sarfaraz, for all the support during the publishing process. At last I will thank my prior master student Eirik Steine Frivåg who has contributed a lot in this book.

CONFLICT OF INTEREST

The author confirms that there is no conflict of interest to declare for this publication.

Terje Kristensen
Bergen University College
Bergen
Norway
E-mail: tkr@hib.no

CHAPTER 1

Introduction

Abstract: This chapter describes the main goal of the book, namely the use of evolutionary algorithms to optimize the K-means algorithm. The outline of the book is also given in the chapter.

Keywords: Background, Case study, Data visualization, Design and implementation, Discussion, Evolutionary algorithms, Introduction, System specification, User interface.

1.1. OVERVIEW

The *K-means algorithm* is one of the most applied algorithms of partitional clustering, where it aims to partition *n* data objects into *K* clusters (or groups) based on some dissimilarity measure. The number of clusters *K* is predefined. A major drawback of the algorithm is that its performance is dependent on the pre-defined value of *K*, because it produces a clustering structure under the assumption that the number of clusters in the data is *K*. If the predefined *K* is not accurate, the algorithm converge into a local minimum. This means that there exists a better solution to the clustering problem. Much can be gained in terms of performance by providing the optimal value of *K*. The purpose of clustering is to analyse unknown data, thus making the task of predefining a good value of *K* an impossible task. Various *Optimization* techniques have been applied to find the best value of *K* for a given clustering problem. Evolutionary algorithms are heuristic optimization algorithms that mimic the process of *natural selection* from biology. They have gained a lot of interest because of its abilities to find global optimal solutions to an optimization problem. *Genetic algorithms* define a paradigm within evolutionary algorithms that model genetic evolution by evolving a population of candidate solutions, to find the optimal solution of a problem. There has been extensive research on adapting genetic algorithms to K-means algorithm, with varying results.

Differential Evolution is another paradigm of evolutionary algorithms that focus on using information about the current population to perform a more intelligent search for an optimal solution. By utilizing information about the search space, we may guide the candidate solutions into more promising areas of the search space, which may produce a better result and also possible increase the convergence speed of the algorithm.

1.2. GOAL

The main goal of this project (book) is to explore the possibilities of adapting evolutionary algorithms to optimize the K-means algorithm. The purpose is to create an evolutionary clustering algorithm that locates the value of *K*, with corresponding cluster centroids, that produces an optimal clustering structure of a given data set. As a part of the research we may adapt both a *Genetic algorithm* and a *Differential Evolution* algorithm to the K-means algorithm, to compare their performance using benchmark tests. As a result, a system is created that performs cluster analysis using evolutionary clustering algorithms. *Visualization* is an important part of clustering to be able to visualize the result of the cluster analysis graphically. Analysing high-dimensional data impose restrictions on how we may visualize the data. We therefore need to present the data in a way that comply to these restrictions.

1.3. OUTLINE

Chapter 1 (Introduction)

This chapter gives a brief overview of the problem domain and presents the goals of this project.

Chapter 2 (Background)

This chapter presents relevant background theory of the main subjects of the book. The chapter is divided into two parts. The first part provides a detailed introduction to the field of cluster analysis, where we focus on presenting the main concepts of cluster analysis. The second part gives an introduction to mathematical optimization.

Chapter 3 (Evolutionary Algorithms)

This chapter gives an introduction to evolutionary algorithms, including an introduction to different paradigms of evolutionary algorithms.

Chapter 4 (System Specification)

This chapter presents the main objectives of the system developed, including both the functional and the non-functional requirements.

Chapter 5 (Design and Implementation)

This chapter provides a description of the overall design of the system and algorithms. First, we describe the overall architecture, including different design patterns applied for the implementation. Second, we will describe the design of relevant algorithms along with the time complexity of them.

Chapter 6 (Data Visualization)

We introduce here the subject of visualization and its application within the system.

Chapter 7 (User Interface)

In this chapter, we describe the user interface of the system.

Chapter 8 (Case Study)

Some cases are studied to compare different clustering algorithms using both artificial and real-life data sets.

Chapter 9 (Discussion)

In this chapter we discuss different aspects and design challenges of adapting evolutionary algorithms to optimize the K-means algorithm.

CHAPTER 2

Background

Abstract: The chapter describes three phases of a pattern recognition system; data acquisition, feature extraction and classification. In addition, different clustering methods are described.

Keywords: Cluster validation, Fuzzy clustering, Hierarchical methods, Partitional clustering.

2.1. CLUSTERING

2.1.1. Introduction

The amount of information available on the Internet is enormous, but only a fraction is relevant to each user. People may only dedicate much time in their busy schedule to consume information, if it is important and only the most valuable information should be presented. As a result, we need information filtering systems. A good example is an e-mail inbox. Without an automatic spam filter, users have to spend their time going through irrelevant email, containing malware, advertisement and other unwanted content. The purpose of an automatic spam filter is to remove unwanted e-mail before it reach the inbox so that only relevant information is presented to the user. A spam filter consists of different criteria, or rules, that are used to filter out spam e-mail. Some of the criteria are generally based on what the general public defines as spam. For instance, e-mail containing the phrase "*Money Back Guarantee"* is in most cases spam [1]. Other criteria are based on a training set generated through interaction with the user. The training set consists of e-mail that is manually identified as "*wanted"* and "*unwanted"*. The filter uses this set as a template for future classification [2].

Over time the training set will evolve, thus making the classification more accurate.

The example above describes a pattern recognition system. A pattern recognition system consists of three phases; *Data acquisition*, *Feature extraction* and *Classification* [3]. In the data acquisition phase the system gathers data by using some set of sensors depending on the environment of the data, to be acquired. In the example above the gathered data would be incoming emails. The system then extracts the features of the data. The purpose of this phase is to select a set of features to be used to classify the data. There are two critical aspects of this phase [3]:

1. One has to choose a criterion for evaluating the significance of a feature.
2. One needs to find the optimal size of the feature set.

In the classification phase the extracted data points are mapped into a set of classes or groups. To summarize, pattern recognition is a transformation from the measurement space M, containing the acquired data, to the feature space F and to the classification space D [3], *i.e.*

$$M \to F \to D. \qquad \textbf{(2.1)}$$

The spam filter classification process is an example of *supervised classification*. Supervised classification is possible when you have a priori information about the dataset, where there is already established a model of the problem domain [4]. There already exist labelled data, meaning that some patterns are already classified, and this labelled data could be used as a training set to generate the classification criteria. The actual training is done based on a classifier function, D. Given a N-dimensional input pattern $\mathbf{x} = [x_1, x_2,...,x_N]$ the function $D(x)$ will classify the pattern as member of one of k possible classes, C_1, C_2, ..., C_k. The training process consists of optimizing the parameters of D to best fit the training set [3], where one adjusts the parameters so that the classifier function is able to map each input pattern to the corresponding class of the training set. After optimizing the parameters of D, the system should be able to classify an unknown pattern into the correct class. The error rate depends on the maturity of the training set where more diversity in the training examples will give a more well-defined classifier function. More details on optimization will be discussed later in the book.

But how do you find patterns and hidden structures in unlabelled data? In biology this process is called *Taxonomy*. In the field of data mining this is known as *unsupervised classification*, *clustering* or *cluster analysis*. The outline of Section 2.1 is as follows: Section 2.1.2 will describe the general definition of clustering. In 2.1.3 one covers definition of similarity between objects. In 2.1.4 we give an introduction to the various clustering methods, and 2.1.5 covers *crisp* and *fuzzy* clustering. At last in 2.1.6 we present *cluster validation*, how to validate the result of the cluster analysis.

2.1.2. General Definition

In biology, *Taxonomy* is known as the science of identifying and naming unknown species and organizing them into a system [5]. The purpose is to organize species in groups based on some common characteristics. The same idea is brought into data mining, especially in clustering where the objective is the same, namely organizing and grouping data objects into different clusters based on common characteristic. There exist various definitions of a cluster, depending on the clustering method. Hierarchical methods, see section 2.1.4, organize clusters based on the distance between objects, and partitional methods creating clusters based on minimization of the mean square error of the clusters. However, what all definitions have in common are that they describe an internal homogeneity and an external separation [6]. This means that, based on some measure of similarity, an object is similar to other objects within the same cluster and dissimilar to objects that are not in the same cluster. The various clustering methods, including the previous two, will be discussed in section 2.1.4 of the book. The general mathematical definition is that you have a set S of n patterns, $S = \{x_1, x_2,...,x_N\}$ where each pattern is denoted as a vector in a N-dimensional feature space, $x = [x_1, x_2, . . . , x_j, . . . , x_N]$ [3]. These patterns are partitioned into K clusters, C_1, C_2, ..., C_K, where

$$C_i \neq \emptyset, \qquad \text{for i} = 1, \ldots, K, \qquad \textbf{(2.2a)}$$

CHAPTER 3

Evolutionary Algorithms

Abstract: The chapter describes different components of evolutionary algorithms and what is meant by mathematical optimization. In addition, genetic and evolutionary programming is defined.

Keywords: Chromosome, Crossover, Cultural algorithms, Differential evolution, Fitness function, Genetic and evolutionary programming, Mutation, Reproduction, Selection.

3.1. INTRODUCTION

Evolutionary algorithms are based on the principles of evolution and Darwin's theory on "natural selection" [13]. In an environment with limited resources individuals have to compete for survival. All individuals possess various properties (traits) and the individuals that are able to best adapt to the environment, *i.e.* individuals with the superior properties, will survive. As time goes on these individuals will mutate, creating individuals that are more adapted to the environment. This idea is brought into computational intelligence where one solves computer based optimization problems by using evolutionary principles from biology. A generic evolutionary algorithm consists of a population of individuals that all propose a solution to an optimization problem. Each individual has a level of fitness representing the strength of the individual or how good the solution is. *Mutation* and *reproduction* operators are used to evolve the population, where solutions with higher fitness are kept while others are discarded. The algorithm terminates when some stopping condition is fulfilled. In the next paragraphs we will describe all these components in detail.

3.1.1. Data Representation Chromosome

Finding the best solution of a mathematical function consists of optimizing each parameter value to produce an optimal value. This process is known as *mathematical optimization*, described in section 3.2. In an evolutionary algorithm the properties of an individual represent one combination of possible values of these parameters, and hence a possible solution to the optimization problem. Each possible solution is known as a *chromosome* where its parameter values are referred to as *genes* [13]. There exist two types of properties:

- *Genotype*, describes properties of an individual that are represented by genes that may be passed from one generation to the next.
- *Phenotype*, describes behavioural properties.

Genes can be represented by binary, discrete or continuous values. In a standard evolutionary algorithm the size of each chromosome is the same for all individuals of the population.

3.1.2. Initial Population

The first step of the algorithm is to generate the initial population. The initial population consists of individuals where gene values are randomly generated, with the purpose of creating a population where its members are well distributed across the entire search space. Each individual has a unique combination of genes to avoid redundant solutions to the optimization problem. The size of the initial population affects the computational complexity of the algorithm and its ability to explore the entire search space. A large population size will increase diversity and favour exploration, but also increases the computational complexity of each generation.

3.1.3. Fitness Function

The fitness function is a mathematical measure of how good an individual is, where the most fitted individuals are more likely to survive to the next generation. The fitness function f maps a chromosome representation into a scalar value

$$f : \Gamma^{n_x} \to \mathbb{R} \tag{3.1}$$

where Γ represents the data type of the elements of a n_x-dimensional chromosome. The formulation of the fitness function is problem dependent, *e.g.* if the parameters are *constrained/unconstrained*, and also if fitness is *absolute* or *relative*. If one has an unconstrained problem one could simply use the objective function as a fitness function, but for constrained problems the parameter constraints need to be included in the function. *Absolute fitness* values are not dependent on fitness of other individuals of the population, while *relative fitness* values are dependent.

3.1.4. Selection

When selecting individuals for reproduction one wants to select individuals with a high fitness level, and thus ensuring a high fitness level of the offspring. This also applies when selecting individuals for the next generation. There are various ways of selecting individuals, where each one is characterized by their *selective pressure*. Selective pressure is a measurement relating to how long it takes for a population to entirely contain the best solution when one only applies the selection operator. High selective pressure will reduce diversity, and thus disfavours exploration. This means that the algorithm converges faster and most likely get stuck in local minima. Too low selective pressure on the other hand results in a slow convergence. Some examples of selection operators are:

- Random Selection
- Proportional Selection
- Tournament Selection
- Rank-based Selection

Different algorithms use different selection operators. A more detailed description of the operators will be given in the chapters describing the different evolutionary algorithms. To ensure that the best individuals survive to the next generation, one could apply elitism where the best individuals are automatically kept for the next generation without being mutated. One could also apply *Hall of fame* where the best individuals in each generation are kept and used as a parent pool for reproduction, ensuring that only the best individuals gets selected for reproduction.

CHAPTER 4

System Specification

Abstract: The system requirements are divided into two types, functional and non-functional. Both of them are described in addition to their data visualization.

Keywords: Functional requirement, Non-functional requirements, System objective.

4.1. INTRODUCTION

The purpose of this chapter is to describe the objective of the system, including requirements and constraints that the system must satisfy. *System requirements* depicts the services that the system must provide and its operational constraints [18]. It should give a detailed description of the functionality that will be implemented, but one should avoid the discussion of specific design choices or implementation. System requirements are divided into two subcategories:

- Functional requirements
- Non-functional requirements

Functional requirements describe the requirements of each service that the system provides, and also the input/output of each service. They should reflect the customers' expectations of the system. Imprecision in the requirements may result in systems that fail to meet the needs of the customer, mainly because the developer misinterpreted the requirements [18]. The description of the functional requirements should contain enough details to avoid misinterpretation, without defining any boundaries on the implementation. *Non-functional requirements* do not describe requirements of a specific functionality, but for the system as a whole. However, the requirements define constraints on the properties of the system. Examples are security requirements, system performance requirements and reliability requirements. There is one important characteristic that makes the functional and the non-functional requirements be different: A system that violates a functional requirement may still be usable, but a system that violates a non-functional is not. For instance, if a web browser should be able to display both text and images (its functional requirement), but fails to display images, the user is still able to display text using the web browser. However, if an air traffic control system fails to meet its security and reliability requirements, it should never be used for air traffic control.

We want to be able to verify the requirements of the system. A requirement stating that the user interface needs to be simple is a poor requirement because it is difficult to test [18]. Therefore, one should restate the requirement to make it quantifiable, *e.g.* "*the total number of errors of an experienced user should not exceed two per day*". The *functional* and *non-functional requirements* of the system is presented in Section 4.3 and 4.4, respectively, and in Section 4.2 the *system objective* is described.

4.2. SYSTEM OBJECTIVE

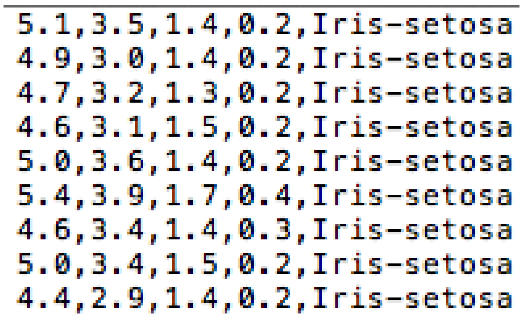
The main objective of the system is to analyse and visualize high-dimensional data, with the purpose of revealing underlying structures that otherwise would be hard to identify. Through visualization one presents the data in a manner that makes it easier for experts to further analyse them. The system provides two evolutionary clustering algorithms to analyse the data, where each one employs a different paradigm of *Evolutionary algorithms* to optimize the *K-means algorithm*.

The cluster analysis should not be dependent on any pre-defined information about the clustering structures to be able to analyse data. An increasing number of dimensions introduce restrictions on how one can visualize the data in a meaningful way. Thus, the system needs to provide a visualization regime that can handle high-dimensional data.

4.3. FUNCTIONAL REQUIREMENTS

4.3.1. System Input

The system should be able to import data from a text file. Each line of the text file represents a data object. The attributes of each data object must be separated by either a "*,*" or a *whitespace*. The relevant attributes must be *numerical*. Some data sets contain meta-information about the data objects in leading and/or trailing attributes. Hence, one should be able to ignore when importing them into the system. An example is the *Iris* data set (See Section 8.3.2), where the last attribute represents which of the Iris flower the data object belongs to. This information is a result of a classification process conducted on the data, and should be ignored, since it is not a feature of the original data (See Fig. **4.1**). The text file can be any ASCII file, as long as the data match the format presented earlier.



```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
```

Fig. (4.1). Data representation of the Iris data set.

4.3.2. Cluster Analysis

The user should be able to select which of the algorithms to do the clustering process and also adjust the specific parameters of the algorithm (*e.g. population size, maximum number of generations, maximum size of individuals*, *etc.*). In addition, the user should be able to select evolutionary operators and adjust their parameters (*e.g.mutation and crossover rate etc.*). After selecting and initializing the algorithms, the user should be able to perform the cluster analysis. After the analysis is finished, the user should be able to import new data sets, and re-initialize algorithms and the operators without the need to reboot the system.

CHAPTER 5

Design and Implementation

Abstract: In this chapter, the system architecture and different tools that have been used in the implementation process such as Netbeans and JavaFX, are described.

Keywords: Complexity of K-means algorithm, Davies-Bouldin Index, Evolutionary operators, Fitness evaluation, Genetic Algorithm, Junit, JavaFX, K-means algorithm, Maven, Netbeans.

5.1. INTRODUCTION

In this chapter, we describe the design and implementation of the system:

- In Section 5.2 one describes the architecture and different design patterns employed in the development of the system.
- Section 5.3 describes the tools employed in the development of the system.
- Section 5.4 deals with the internal data structure used to represent imported data of the system, and the implementation of the *K-means clustering algorithm*.
- Section 5.5 describes two evolutionary clustering algorithms.

The time-complexity of the algorithms are also analysed using *Big-O notation* [19].

5.2. SYSTEM ARCHITECTURE

Fig. (**5.1**) shows a *high-level* representation of the system architecture, where relationships and dependencies between classes are illustrated. The system is designed to satisfy the *Model-View-Controller* (MVC) architectural pattern to separate the User Interface (UI) and business logic. The controller classes and view classes of MVC are marked with pink. The MVC pattern is further described in Chapter 7. The classes marked with blue constitute the *Evolutionary Clustering algorithms* (ECA) (Section 5.5), the green classes represent the *K-Means algorithm* and its components (Section 5.4.2), and the orange represents the utility classes, *i.e.* classes for importing and representing data in the system (Section 5.4.1).

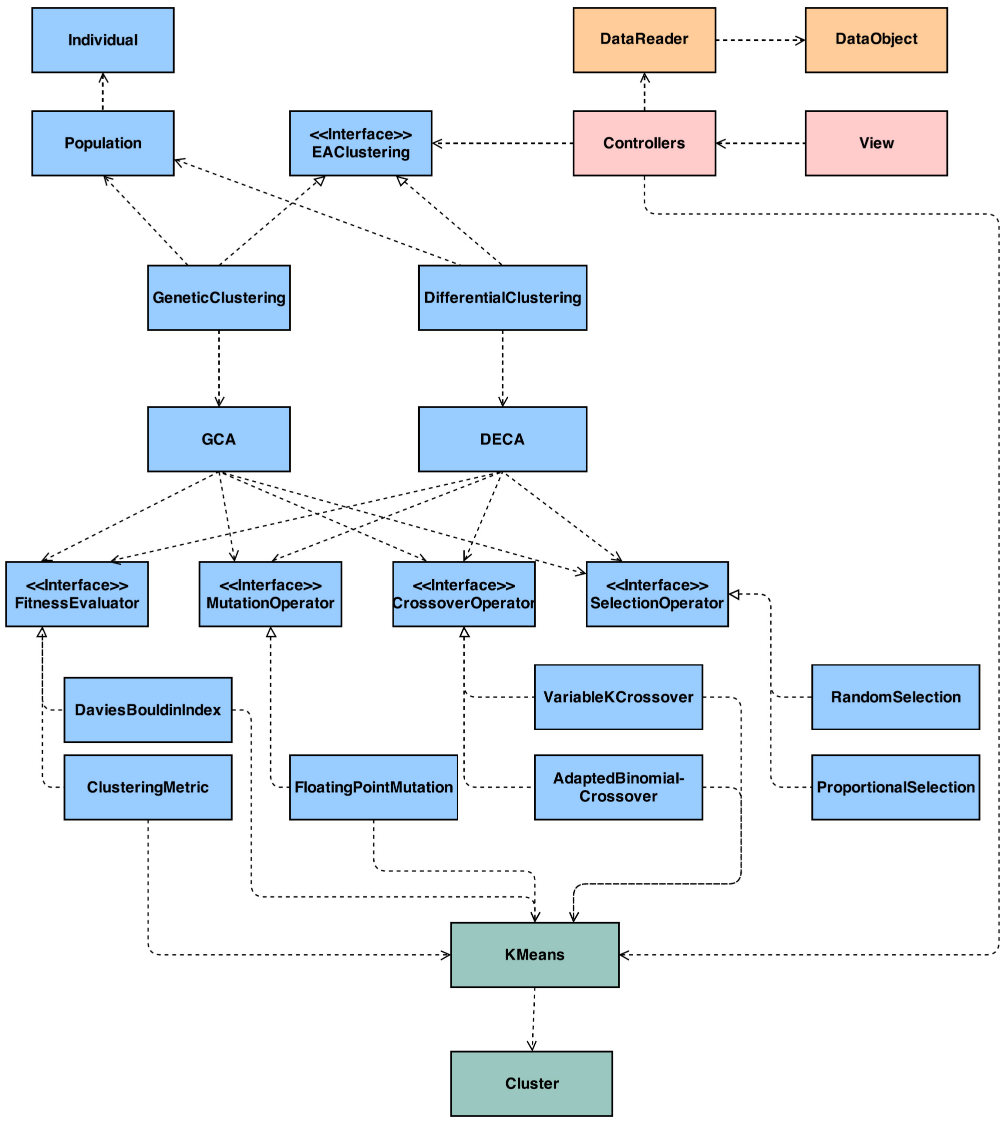


Fig. (5.1). UML diagram of the high-level architecture of the system.

5.2.1. Dependency Injection

The evolutionary clustering algorithms (ECA) are dependent on different evolutionary operators, such as mutation operators, selection operators and crossover operators. To make the ECAs loosely coupled from the operators, we pass a reference to each operator through the ECAs constructor, rather than creating the dependencies in the respective ECA. This is known as *Dependency Injection (DI)* pattern. For instance, the user specifies which of the available fit-ness evaluation methods he wants to employ in the relative ECA through the user interface. An instance of the selected fitness evaluation method is then created by the controller class instead of being created in the ECA, and as a result the ECA does not need to know anything about the fitness evaluation method. Fig. (**5.2**) illustrates an example of how the DI pattern is employed in our sys-tem. In this example, the user wants to employ the *Davies-Bouldin Index (DBI)* (Section 5.5.6) as a fitness evaluator of the *Genetic Clustering algorithm (GCA)*. The controller creates an instance of DBI and pass this object down to the GCA. The same case is illustrated in Fig. (**5.3**), but without the DI pattern. In this case, the GCA relies on knowledge about DBI to be able to create an instance of it. If we are going to alter the DBI, we must also update the GCA based on the alteration made to the DBI. In reference to the *maintainability* requirement of Section 4.4, developers can alter the DBI without the need to alter the GCA. Thus, making the system easier to maintain. Note that the DI pattern is employed for all evolutionary operators used in the ECA.

5.2.2. Open-Closed Principle

To be able to extend the system in the future with additional evolutionary operators or fitness evaluation methods, it is important that relevant components conform to the *Open-Closed Principle (OCP)* [20]. By conforming to the OCP, we create components that are *open* for extension and *closed* for modification. This can be achieved through *abstraction* [20]. In the system architecture presented earlier, the behaviour of the ECAs is closed for modification since they depend on the fixed behaviour of the evolutionary operators and the fitness evaluator, defined by their respective interfaces. At the same time, the behaviour of the ECAs is extendible by creating new derivatives of the different interfaces.

CHAPTER 6

Data Visualization

Abstract: Different methods for visualization of data are described in the chapter. One important aspect is how to handle colours. Colour are separated in three different sub-channels. Different methods to reduce the dimension of data are also described.

Keywords: Data types, Dissimilarity matrix, Multi-dimensional scaling, Normalization, Principal components, Visual channels.

6.1. INTRODUCTION

In computer science visualization is defined as the visual representation of a domain using graphics, images or animation, to present the data or the structures of large data sets [28]. Visualization is used to understand the meaning of data, in situations where the problem may be too vague for a computer to handle [29]. For instance, illustrating the benefits of visualisation, is a subway map. The purpose of a subway map is to illustrate a subway system, consisting of subway lines, stations and transit points. This makes it easier for travellers to find the best way to travel from location *A* to location *B*. We may imagine what kind of problems a traveller needs to cope with if presented by only the raw data, containing for instance, the GPS-coordinates of each subway station. Visualization provides us with two types of information [28]:

1. Answers to concrete questions and hypothesis related to a given data set.
2. Facts about data of a given data set that one is unaware of.

Visualization may be used to answer both *quantitative* and *qualitative* questions. For quantitative questions, *e.g.* *"what is the output of function f*(*x*) *for all values of x within a given interval?"*. Visualization is not indispensable, it is an intuitive

Terje Kristensen

way to represent intervals rather than displaying a list of values. But for qualitative questions, visualization is indispensable. An example of a qualitative question is *“given a medical scan of a patient, are there any anomalies that indicate medical problems?"*. A computer program could find patterns in a blood sample that differ from other patterns, but it is not able to classify it as an anomaly [28]. One needs a human expert, for instance a doctor, to analyse the pattern. His opinion is based on prior expertise and experience. However, human experts are not able to make good decisions without being presented with meaningful data. Thus, advocating the importance of using visualization.

6.2. DATA TYPES

In cluster analysis the data type determines which operations one may utilize when analysing the data. In visualization, the data types define boundaries on how one may visualize the data. If the purpose is to visualize spatial data, one may choose an approach where the spatial information is emphasized and not lost. T. Munzer defines three types of data [29]: *quantitative, ordered and categorical data*. Quantitative data is often numerical. We can use arithmetic operations on them. One cannot do arithmetical operations on ordered data, but they have a well-defined ordering that one can utilize. Categorical data has no ordering. The data may only be divided into groups. In some data sets there exists a specific relationship between its members, and it is important that the visualization scheme expresses these relationships. This is referred to as *relational data* or *graphs* [29]. Some visualization guidelines of certain types of data are illustrated in section 6.3.

6.3. VISUAL CHANNELS

Visual channels are used to encode information [29]. Examples of visual channels are spatial information, such as horizontal and vertical positions, color, size *and shape*. Multiple visual channels are applied to represent the different dimensions of some d-dimensional data object. Visualizing a 3-dimensional vector, $x = \{x_1, x_2, x_3\}$, in a scatter plot one could use vertical and horizontal spatial points to represent values of x_1 and x_2, and size of the plotted points to represent values for x_3. When selecting the type of a visual channel to represent some data, it is important to consider the characteristics of the visual channels [29]. Not all channels are equally *distinguishable*, and studies have shown that people's ability to distinguish information encoded in different visual channels from each other, is dependent on the *data type* one is using. Some visual channels work well for categorical data, but not for ordered data, where the order of objects needs to be emphasized. *Separability* is an important characteristic of a visual channel. When visualizing categorical data it is important that the user can separate two categories from each other. The positioning of different objects is highly separable, but distinguishing objects by their vertical and horizontal size may be difficult [29]. Color is a powerful channel, but to avoid confusing the viewer, one has to consider its properties before applying it to a visualization problem [29]. Color may be separated into three different sub-channels:

- *Hue*
- *Saturation*
- *Lightness*

All available colors are derivatives of the primary colors red, green and blue (RGB), and the *hue* represents a given combination of these three colors. The different values of hue are represented in Fig. (**6.1a**) (image obtained from [30]). *Saturation* defines how intense a color is, from a slight grey tone to a vivid color, illustrated in Fig. (**6.1b**) (image obtained from [31]). *Lightness* defines how light/dark the relative color is. The three sub-channels in combination are illustrated in Fig. (**6.1c**) (image obtained from [32]).

Hue is good for visualizing categorical data because one could generate a large set of hues that are separable from each other [29]. But it is not appropriate when representing ordered data because it does not have any implicit perceptual ordering [29]. For instance, it is impossible to order yellow, purple and pink. For ordered data, it would be more preferable to use lightness or saturation because of its implicit ordering. The number of channels to use for spatial layouts has been extensively discussed [29]. Compared to 2D visualization, three dimensions makes an additional perspective when visualizing data objects.

However, researchers have begun to experience the *cost* of 3-dimensional visualization of datasets. The costs are referred to as *Occlusion* and *Perspective*

CHAPTER 7

User Interface

Abstract: The chapter gives an introduction to the architecture based on Model-View - Controller (MVC) patterns. Such an architecture separates the user interface from the business logic. The MVC enables us to divide the system into three components. The user provides the parameter values. When an algorithm is terminated the results may be shown graphically.

Keywords: Evolutionary operators, JavaFX, MVC.

7.1. INTRODUCTION

This chapter gives an overview of the *User Interface* (UI) of the system. The UI facilitates the interaction between the user and the system and provides the following functionality:

1. **Import data to the system**
2. **Select/initialize evolutionary clustering algorithms**
3. **Select/initialize evolutionary operators**
4. **Visualize the clustering structure**

The different functionality will be described in Sections 7.3, 7.4, 7.5 and 7.6. As mentioned in Section 5.2, the architecture of the system complies to the MVC pattern which separates the UI from the business logic. The MVC pattern and its role in the system is presented in Section 7.2.

7.2. MODEL-VIEW-CONTROLLER

The *Model-View-Controller* enables us to divide the system into three different subsystems: *Model, View and Controller* [36]. By doing this we separate the business logic (the model) from the UI (the view), and as a result, modification in either subsystem will not affect the other. All communication between the two is performed using the controller subsystem. By employing JavaFX to develop the UI we enforce the MVC pattern on our system. The user interface is created by defining the different UI components using a XML-based language known as FXML. The behaviour of the different components is defined in a separate controller class. Fig. (**7.1**) illustrates a basic example of how the MVC pattern is employed in the system. All UI components are defined in the *UI* FXML document, where the specific behaviour of each component is defined in *UIController*. The model, *i.e.* business logic in this example, are the classes marked with blue. For instance, the user decides to use the DECA to perform the cluster analysis, and specifies this through the *UI* (See Section 7.4). The *UIController* then creates an instance of the selected algorithm and run the algorithm on behalf of the *UI*. The algorithm analyses the data and provides the *UIController* with the results from the analysis. The *UIController* may alter relevant *UI* components based on the results from the analysis.

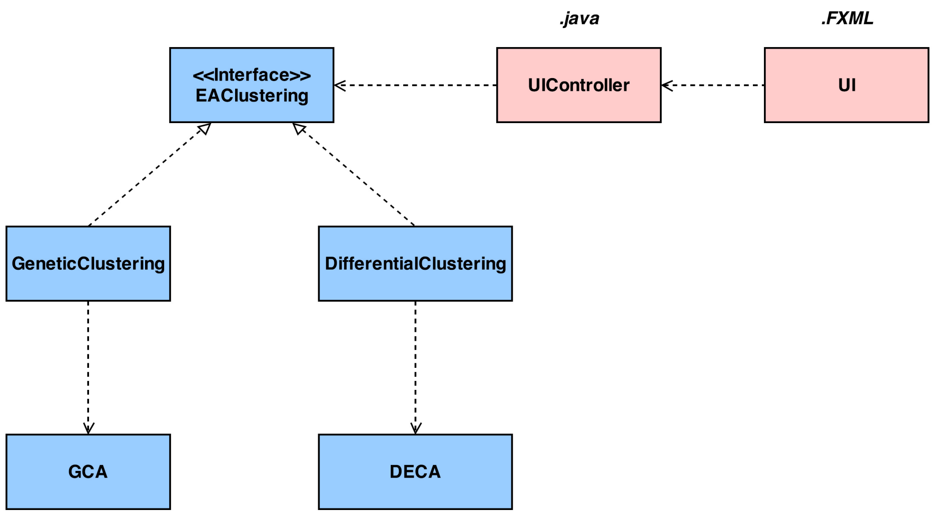


Fig. (7.1). Model-View-Controller using JavaFX.

7.3. IMPORT OF DATA

Fig. (**7.2**) shows the functionality that enables the user to upload data to the system. The user specifies *from/to* indices that the DataReader employs to create

DataObjects (see Section 5.4.1), represented by the *“From index:”* and *“To index:”* input fields in the figure.

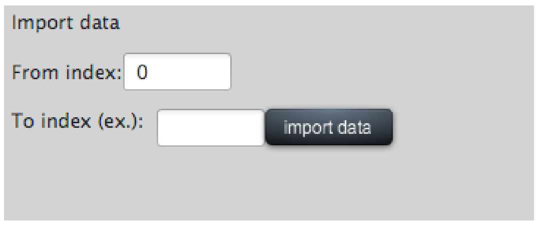


Fig. (7.2). Import of data.

When pressing the “import data”-button, a pop-up window lets the user browse the file system to locate the specific data set that the user wants to analyse (see Fig. **7.3**). Note that the indices must be specified before importing the data.

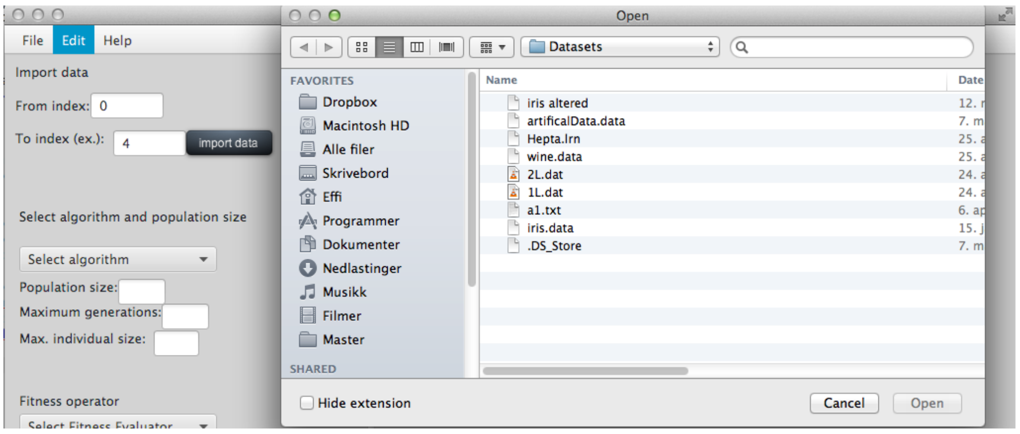


Fig. (7.3). Browsing the file system.

7.4. SELECTING AND INITIALIZING THE ALGORITHMS

From the drop-down menu in Fig. (**7.4**), the user may select which of the different ECAs from Section 5.5 to be used in the cluster analysis. The user must also provide parameter values, such as *population size*, *maximum number of generations and the maximum size of individuals*.

After providing the parameters, the user must press the “save”-button to create an

CHAPTER 8

A Case Study

Abstract: To compare the different algorithms three data sets have been used. Different benchmarking sets have been applied and the results of the experiments are presented in tables and illustrated graphically.

Keywords: DECA, GCA, Hepta, Iris and Wine, JavaFX, Median, Scatter Chart.

8.1. INTRODUCTION

The purpose of the case study is to compare the performance of the evolutionary clustering algorithms presented in Section 5.5 by using some *benchmarking criteria*. We want to explore if the algorithms are able to classify the correct number of clusters when presented with data of different complexity. In addition, we want to compare the quality of the solutions provided by the algorithms.

8.2. BENCHMARKING ENVIRONMENT AND CRITERIA

To perform benchmark comparisons of the different algorithms we use three data sets:

- **Hepta** – artificial data created for benchmarking purposes of the clustering algorithms.
- **Iris** – three species of the Iris flower.
- **Wine** – wines from three different cultivars from the same region in Italy.

The three data sets differ in size, both the number of data objects and the dimension of the data, and also how hard it is to classify the clusters. Some data sets contain clusters that overlap, meaning that the clusters are hard to separate.

This makes it harder to determine the correct number of clusters. The characteristics of the different data sets are described in Section 8.3.1, 8.3.2 and 8.3.3, respectively.

When we compare heuristic optimization algorithms, we need to run the algorithms more than once to remove the randomness from the benchmarking results. In the test environment, each algorithm is run *r* independent *trial runs* for each data set.

8.2.1. Benchmarking Criteria

The benchmarking criteria we have used are:

- **Average fitness of solutions**
  The average quality, Q_{average}, of the candidate solutions obtained from the trial runs is calculated as

$$Q_{\text{average}} = \frac{1}{r}\sum_{i=0}^{r} F_i(x) \tag{8.1}$$

  where $F_i(x)$ denotes the fitness level of the most fitted individuals of the *i*th trial run, and *r* is the total number of trial runs.

- **Average number of *K* clusters**
  The average number of *K* clusters, K_{average}, obtained from the trial runs is calculated as

$$K_{\text{average}} = \frac{1}{r}\sum_{i=0}^{r} S_i(x) \tag{8.2}$$

  where $S_i(x)$ denotes the size of the most fitted individuals, from the *i*th trial run. This value reflects how well the algorithms are able to determine the correct number of clusters.

In addition to the mean value, we employ the *median value* because the mean is sensitive to *extreme values* (or outliers). In our case, if the results from some of the trial runs are unusual compared to the results of the rest. In this case the mean will loose its ability to represent typical or most frequent results of the trial runs.

The median is less sensitive to extreme values since it represents the most frequent results of the algorithms. The median is the center value of a sorted list. For a list of n elements, where n is an odd number, the median m is defined as

$$m = \frac{n+1}{2} \qquad \textbf{(8.3)}$$

When n is an even number, the median is the average of the two middle numbers. The median of the K clusters obtained from each trial run is calculated by the same approach as applied to the median of the fitness levels. The median values are denoted as Q_{median} and K_{median}.

We may also measure the convergence speed, *i.e.* the average number of generations it takes for the algorithm to terminate. Termination criteria are described in Section 5.5.

8.2.2. Testing the Parameters

The number of trial runs of each algorithm is set to r =50. For both algorithms, the population size is p = 50, maximum size of individuals is K_{max}= 15 and maximum number of generations is t_{max}= 100. In GCA we employ *Variable K crossover*, *Floating-point mutation* and *Proportional Selection* as evolutionary operators. The mutation rate is set to p_m= 0.1 and crossover rate to p_c= 0.8. The evolutionary operators used in the DECA is *Adapted binomial crossover* and *Random selection*, where the crossover rate is set to p_c= 0.7 and the scaling factor β = 0.5. To evaluate fitness we apply Davies-Bouldin Index for both algorithms.

8.3. DATA SETS

This section describes the features of the data used for the benchmarking tests. The descriptors of the data sets are summarized in Table **8.3.3**.

8.3.1. Hepta Data Set

The *Hepta* data set is a part of the *Fundamental Clustering Problem Suite (FCPS)*

CHAPTER 9

Discussion

Abstract: In this chapter we discuss different challenges of using evolutionary algorithms to optimize the K-means algorithm. One problem is how to handle empty clusters. In addition, the time complexity of the different algorithms is shown.

Keywords: Convergence speed, Data representation, Empty clusters, Fitness measure, Invalid cluster structures, Time complexity.

9.1. INTRODUCTION

In this chapter, we want to discuss some of the design challenges that we have faced during the process of using evolutionary algorithms to optimize the K-means algorithm.

9.2. DATA REPRESENTATION

The data representation scheme described in Section 5.4, introduces new constraints that we need to handle. When a standard data representation scheme of evolutionary algorithm is used, genes of individuals represent a *finite* set distinct features of the data. This implies that individuals may contain equal values in its genetic material since each value represents a measurement of different dimension. When optimizing clustering structures, duplicates of the genetic material implies invalid clustering structures.

For instane, when an individual x_1, representing a clustering structure K = 3, is initialized with the genetic material

$$x_1 = \{c_1, c_1, c_1\} \qquad \textbf{(9.1)}$$

where c_1 is some arbitrary centroid. We see that all three clusters are assigned the same cluster centroid. As a result, when clustering the data (Algorithm **5.4.1**) all data objects will be assigned to the first cluster, leaving the other two clusters empty. This results in an invalid clustering structure. Invalid clustering structures result in incorrect fitness evaluation and must be avoided. Invalid clustering structures are further discussed in Section 9.3.

During the initialization process we want to create a population of distinct individuals to avoid individuals that explore the same parts of the search space. The data representation scheme is described in Section 3.1. When two individuals of size n are compared, we only need to verify that none of the n gene-pairs are equal. This has a time-complexity of $O(n)$. Thus initializing a population of p distinct individuals has the time-complexity of $O(p^2 \cdot n)$. When the representation scheme described in Section 5.4 is used, the initialization of a population becomes a bit more complicated. For instance, two individuals x_1 and x_2, where both represent the clustering structure of $K = 2$, are initialized with the following genetic material

$$\begin{aligned} x_1 &= \{c_1, c_2\} \\ x_2 &= \{c_2, c_1\} \end{aligned} \tag{9.2}$$

where c_1 and c_2 represent some arbitrary centroids. Since each gene represents a cluster centroid, the relative ordering of the genetic material is trivial. As a result, x_1 and x_2 represent the same clustering structure. To check that two individuals of size n are equal we have to cross-check their genetic material, which has the time-complexity of $O(n^2)$. For a population of size p this results in a time-complexity of $O(p^2 \cdot n^2)$. Some redundancy may be allowed in the initial population, but a population that solely consists of equal individuals will impair the exploration abilities of the algorithm. On the other hand, large data sets reduce the probability of producing redundant individuals. By doing redundancy check we are adding unnecessary computational complexity.

9.3. INVALID CLUSTERING STRUCTURES

To initialize the centroids of K clusters by randomly generating data objects within the boundaries of the search space, one could end up with cluster centroids that are far away from the data undergoing clustering. By this scenario, some of the clusters could end up without members, thus failing to satisfy one of the requirements of partitional clustering, that all clusters have to be non-empty. In a standard K-means algorithm the limitation is handled by randomly selecting data objects from the data set to serve as cluster centroids [42]. Since centroids are sampled from the data set, each cluster will at least have one member. The same approach is used during the initialization process of the ECA, where the genetic material of individuals is randomly sampled from the data set. This ensures that all clustering structures are valid. When evolving the population, the genetic material of the individuals is altered. As a result, it is no longer guaranteed that the clustering structures of the individuals consist of entirely non-empty clusters.

The fitness evaluation of the individuals will be compromised if we allow individuals to represent invalid clustering structures. For instance, the Davies-Bouldin Index of a clustering structure that contains empty clusters is *undefined*, because the intra- and inter-cluster distances of empty clusters cannot be computed. The same situation occurs when we use the clustering metric as a fitness measure. This illustrates that it is necessary to add mechanisms to handle empty clusters of clustering structures, before we evaluate how good they are. Various methods for handling empty clusters in a standard K-means algorithm are described in [42]:

- *Removing empty clusters:* if a cluster is empty after performing clustering, we remove the relevant cluster from the clustering structure.
- *Reinitialize cluster centroids:* if a cluster is empty after performing clustering, one reinitialize the cluster centroid by randomly selecting a data object from one of the other clusters.

With respect to the exploration of the search space, the consequence of *removing* empty clusters, one biases the exploration abilities of the algorithm towards smaller individuals. When we remove empty clusters, we reduce the number of

CHAPTER 10

Summary and Future Directions

Abstract: In this chapter we make a summary of how to optimize the K-means clustering algorithm based on evolutionary computing. The system is still missing a user interface to handle invalid user input. Parallel coordinates that may be used as a tool to visualize data in high-dimensional spaces is only given a short introduction. In addition, Particle Swarm Optimization (PSO) is also mentioned to find global solutions to optimization problems.

Keywords: Clustering metric, Invalid cluster structures, Overplotting, Parallel coordinates, PSO.

10.1. SUMMARY

The main goal of this book is to explore the possibilities of adapting an evolutionary algorithm to optimize the K-means clustering algorithm. As a part of our research we wanted to create two evolutionary clustering algorithms, each one using a different paradigm of evolutionary algorithms, and compare their performance. Since there is much research on adapting genetic algorithms to optimize the K-means algorithm, we also want to test out another paradigm of evolutionary algorithm, known as Differential Evolution. Differential Evolution represents a paradigm of evolutionary algorithms that utilizes information about the search space in order to perform a more intelligent search. To compare the performance of the two algorithms, some benchmark tests using data with different characteristics have been done.

A system has been developed where the user may upload data and perform cluster analysis using the two evolutionary clustering algorithms. In addition, two evolutionary operators and fitness evaluation methods have been used, the user

Terje Kristensen

may specify the parameters of the algorithms. In order to visualize high-dimensional data in a two-dimensional scatter plot, classical multi-dimensional scaling is used to perform dimension reduction. The data is presenting by two principal components of highest variance. A complete description of the system was given in Chapter 5, with a detailed description of the time-complexity of algorithms used.

The main goal of this book (project), discussed in Chapter 9 is to present different aspects of evolutionary algorithms to optimize the K-means algorithm. This may also result in using different constraints when employing an evolutionary algorithm to optimize the K-means algorithm. The possibility of producing individuals that represent invalid clustering structures implies that all evolutionary operators must contain a constraint that inhibits them from creating invalid individuals. This results in increase of the time-complexity of the GCA, but does not affect the time-complexity of the DECA. We also discussed possible effects the constraints have on the convergence rate.

A case study revealed that both algorithms perform well on data that contain clearly defined clusters. However, both algorithms are struggling with data containing messy clustering structures, due to the limitation of using the Davies-Bouldin Index as a fitness measure. In addition, even though the algorithms are able to find good values of *K*, the quality level of the solutions was quite low compared to solutions provided by a standard K-means algorithm. This indicates that both algorithms have problems with exploiting the search space to find optimal positions for the cluster centroids.

10.2. ALGORITHM IMPROVEMENTS

10.2.1. Exploitation Abilities

As described in Section 3.1 the purpose of the crossover operator is to *exploit* promising parts of the search space, while the mutation operator should move individuals in different directions with the purpose of *exploring* all parts of the search space. The GCA uses the Variable K crossover operator to explore different values of *K*, while the Floating-point mutation operator is used to find good positions for the *K* cluster centroids. One may argue that the behaviour of the two operators does not comply with the behaviour described in Section 3.1, because the Variable K crossover operator explores the search space while the Floating-point mutation operator exploits the space of *K*. A possible approach is to make the mutation operator to explore the space of *K*, while the crossover operator is used to exploits the space of the *K*, with the purpose of locating the optimal position of the *K* cluster centroids. This may be accomplished by adjusting the crossover operator to produce offspring of equal size of the most fitted parent, while creating a mutation operator that alters the size of the individual.

10.2.2. Invalid Clustering Structures

In Chapter 9 we discussed how empty clusters resulted in incorrect fitness evaluation of individuals. *Reinitialization* of cluster centroids of empty clusters during the fitness evaluation is proposed as a possible solution, but was discarded because one should rather prevent invalid clustering structures being created by the evolutionary operators. We also discussed the possibility that the prevention of invalid clustering structures may lead to a smaller convergence rate since a lot of crossover and mutation operations may be rejected. Rather than rejecting invalid individuals, we may apply the reinitialization approach presented in the discussion, to correct invalid individuals produced by the operators. Rejected individuals may contain good genetic material which is lost when these individuals are rejected.

10.3. SYSTEM IMPROVEMENTS

10.3.1. Error Dialog Boxes

A future improvement is to add functionality that handles invalid input of the user interface. If the user provides a string to the mutation rate instead of a number, the system will crash when the analysis is done. The system should rather display error dialog boxes so the user could provide correct input.

10.3.2. System Output

The system presents the result of the cluster analysis during run-time, both

Bibliography

[1] MailChimp, *How spam filters think,* 2013. Available at: http://kb.mailchimp.com/article/ how-spa--filters-think accessed: 06/09/2013

[2] D. Cook, D. Swayne, and A. Buja, *Interactive and Dynamic Graphics for Data Analysis: With R and GGobi, ser. Use R!.* Springer, 2007, pp. 63-64. [Online] Available at: http://books.google.no/books?id=sC0Ij2r_KLcC [http://dx.doi.org/10.1007/978-0-387-71762-3_4]

[3] S. Bandyopadhyay, and S. Pal, "Classification and Learning Using Genetic Algorithms: Applications in Bioinformatics and Web Intelligence", In: *ser. Natural Computing Series.* Springer: Berlin Heidelberg, 2007. [Online] Available at: http://books.google.no/books?id=jHMGohuVHhgC

[4] M. Seeger, "Learning with labeled and unlabeled data", In: *Tech. Rep.* University of Ed-Inburgh, 2002.

[5] Natural History Museum, *What is taxonomy?,* 2013. Available at: http://www.nhm.ac. uk/nature-online/science-of-natural-history/taxonomy-systematics/ what-is-taxonomy/index.htmlaccessed: 23/09/2013

[6] R. Xu, and D. Wunsch, "Clustering", *IEEE Press Series on Computational Intelligence. Wiley,* 2008. [Online]. Available at: http://www.google.no/ books?id=kYC3YCyl_tkC

[7] G. Gan, C. Ma, and J. Wu, *Data clustering: theory, algorithms, and applications.,* vol. 20. Siam, 2007. [http://dx.doi.org/10.1137/1.9780898718348]

[8] N. K. Visalakshi, and K. Thangavel, "Impact of normalization in distributed K-means clustering", *Int. J. Soft Computing,* vol. 4, 2009.

[9] J.V. Oliveira, and W. Pedrycz, *Advances in Fuzzy Clustering and Its Applications..* John Wiley & Sons, Inc.: New York, NY, USA, 2007. [http://dx.doi.org/10.1002/9780470061190]

[10] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms..* Kluwer Academic Publishers: Norwell, MA, USA, 1981. [http://dx.doi.org/10.1007/978-1-4757-0450-1]

[11] D.L. Davies, and D.W. Bouldin, "A cluster separation measure", *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. PAMI-1, no. 2, pp. 224-227, 1979. [http://dx.doi.org/10.1109/TPAMI.1979.4766909] [PMID: 21868852]

[12] N. Bolshakova, and F. Azuaje, "Cluster validation techniques for genome expression data", *Signal Process.,* vol. 83, no. 4, pp. 825-833, 2003. [http://dx.doi.org/10.1016/S0165-1684(02)00475-9]

[13] A. Engelbrecht, *Computational Intelligence: An Introduction.* 2nd ed Wiley, 2007. [Online]. Available at: http://books.google.no/books?id=IZosIcgJMjUC [http://dx.doi.org/10.1002/9780470512517]

[14] D. Simon, *Evolutionary Optimization Algorithms..* Wiley, 2013. [Online]. Available at: http://books.google.no/books?id=gwUwIEPqk30C

[15] R. Adams, and C. Essex, *Calculus: A Complete Course.*. 7th ed Pearson Canada, 2010.

[16] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", In: *ser. Artificial intelligence.* Springer, 1996. [Online]. Available at: http://books.google.no/books?id=vlhLAobsK68C

[17] K.P. Wang, and J. Yuryevich, "Evolutionary-programming-based algorithm for environmentally-constrained economic dispatch", In: *Power Systems.,* vol. 13. IEEE Transactions on, 1998.

[18] I. Sommerville, "Software Engineering", In: *International computer science series.* 8th ed Addison-Wesley, 2007. [Online]. Available at: http://books. google.no/books?id=B7idKfL0H64C

[19] S. Dasgupta, C. Papadimitriou, and U. Vazirani, *Algorithms,* McGraw-Hill Education, 2008. [Online]. Available at: http://books.google.no/books?id=LaIqnwEACAAJ

[20] R. Martin, "Agile Software Development: Principles, Patterns, and Practices", *ser. Alan Apt Series,* Prentice Hall/Pearson Education, 2003. [Online]. Available:http://books.google.no/ books?id=0HYhAQAAIAAJ

[21] Oracle, "What is javafx", Available at: http://docs.oracle.com/javafx/2/overview/ jfxpub-overview.htm accessed: 06/03/2014.

[22] The Apache Software Foundation, *Apache maven project,* 2014. Available at: http:// maven.apache. org/accessed: 07/01/2014.

[23] Git, *About,* 2014. Available at: http://git-scm.com/about/ accessed: 07/01/2014.

[24] GitHub, Inc., *Github,* 2014. Available at: https://github.com/accessed: 07/01/2014.

[25] D.S. Kent Beck, "Erich Gamma and M. Clark", *Junit,* 2012. Available at: http://junit.org/accessed: 07/11/2013.

[26] U. Maulik, and S. Bandyopadhyay, "Genetic algorithm-based clustering technique", *Pattern Recognit.,* vol. 33, no. 9, pp. 1455-1465, 2000.
[http://dx.doi.org/10.1016/S0031-3203(99)00137-5]

[27] H. Yuan, and J. He, "Evolutionary design of operational amplifier using variable-length differential evolution algorithm", *Computer Application and System Modeling (ICCASM)* International Conference, vol. 4, pp. V4-610-614, 2010.

[28] A. Telea, "Data Visualization: Principles and Practice", In: *ser. Ak Peters Series.* A K PETERS Limited (MA), 2008. [Online]. Available: http://books.google.no/books?id=OFEmGQAACAAJ

[29] T. Munzer, "Visualization", In: *Fundamentals of Computer Graphics, ser. Ak Peters Series.*. Taylor & Francis, 2009, pp. 675-712.

[30] D. Mize, *Compose,* 2014. Available at: http://3.bp.blogspot.com/_vHdM2yoeMlM/ SZsomzDKYlI/ AAAAAAAAD1s/mfuhg6cuARU/s1600/colorwheel_1.jpgaccessed: 08/02/2014

[31] OrgeBot, *Scale of saturation,* 2014. Available at: http://commons.wikimedia.org/wiki/File:Saturat iondemo.pngaccessed: 09/02/2014

[32] D. Shark, *The hsl color model mapped to a cylinder*. Available at: http://commons.wikimedia.org/wiki/File:HSL_color_solid_cylinder_alpha_ lowgamma.pngthis file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

[33] I. Borg, and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications, ser. Springer Series in Statistics,* Springer, 2005. [Online]. Available at: http://www.google.no/books?id=duTODldZzRcC

[34] Algorithmics Group, *MDSJ: Java library for multidimensional scaling (version 0.2),* 2009. Available at: http://www.uni-marburg.de/fb12/datenbionik/data? language_sync=1

[35] S. Stober, C. Hentschel, and A. Nurnberger, "Multi-facet exploration of image collections with an adaptive multi-focus zoomable interface", *Neural Networks (IJCNN), The 2010 International Joint Conference on. IEEE,* 2010pp. 1-8 [http://dx.doi.org/10.1109/IJCNN.2010.5596747]

[36] S. Ramnath, and B. Dathan, "Object-Oriented Analysis and Design", *ser. Undergraduate Topics in Computer Science,* Springer, 2010. [Online] Available at: http://books.google.no/books?id=2oLOia08YZ0C

[37] A. Ultsch, *Fundamental clustering problem suite,* 2014. Available at: http://www. uni-marburg.de/fb12/datenbionik/data?language_sync=1 Accessed: 05/04/2014

[38] E. Anderson, "The species problem in iris", *Annals of the Missouri Botanical Garden,* vol. 23, pp. 457-509, 1936. Available: http://biostor.org/reference/11559 [http://dx.doi.org/10.2307/2394164]

[39] R.A. Fisher, "The use of multiple measurements in taxonomic problems", *Ann. Eugen.,* vol. 7, no. 2, pp. 179-188, 1936. [http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x]

[40] K. Bache, and M. Lichman, *UCI machine learning repository,* 2013. Available at: http://archive.ics.uci.edu/ml

[41] S. Saitta, B. Raphael, and I.F. Smith, "A bounded index for cluster validity", In: *Machine Learning and Data Mining in Pattern Recognition..* Springer, 2007, pp. 174-187. [http://dx.doi.org/10.1007/978-3-540-73499-4_14]

[42] F. Torrent-Fontbona, V. Muñoz Solà, and B. López Ibáñez, *Solving large location-allocation problems by clustering and simulated annealing.* 2013.

[43] A. Inselberg, "Parallel Coordinates: Visual Multidimensional Geometry and Its Applications", *ser. Advanced series in agricultural sciences,* Springer, 2009. [Online]. Available at: http://books.google.no/books?id=DkQlVpv6kzsC [http://dx.doi.org/10.1007/978-0-387-68628-8]

[44] M. Theus, "High-dimensional data visualization", In: *Handbook of Data Visualization, ser. Springer Handbooks Comp.Statistics.* Springer: Berlin Heidelberg, 2008, pp. 151-178. [Online]. [http://dx.doi.org/10.1007/978-3-540-33037-0_7]

[45] E.J. Wegman, "Hyperdimensional data analysis using parallel coordinates", *J. Am. Stat. Assoc.,* vol. 85, no. 411, pp. 664-675, 1990. [http://dx.doi.org/10.1080/01621459.1990.10474926]

[46] S. Bandyopadhyay, and U. Maulik, "Nonparametric genetic clustering: comparison of validity indices", *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 31, no. 1, pp. 120-125, 2001.

SUBJECT INDEX