Universidade Estadual de Campinas
Instituto de Computação

David Aparco Cardenas

Iterative Optimum-Path Forest: A Graph-Based Data Clustering Framework

Floresta de Caminhos Ótimos Iterativa: Um Arcabouço para Agrupamento de Dados Baseado em Grafos

CAMPINAS
2021

David Aparco Cardenas

Iterative Optimum-Path Forest: A Graph-Based Data Clustering Framework

Floresta de Caminhos Ótimos Iterativa: Um Arcabouço para Agrupamento de Dados Baseado em Grafos

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Pedro Jussieu de Rezende
Co-supervisor/Coorientador: Prof. Dr. Alexandre Xavier Falcão

Este exemplar corresponde à versão final da Dissertação defendida por David Aparco Cardenas e orientada pelo Prof. Dr. Pedro Jussieu de Rezende.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Ap12i
Aparco Cardenas, David, 1991-
Iterative optimum-path forest : a graph-based data clustering framework / David Aparco Cardenas. – Campinas, SP : [s.n.], 2021.

Orientador: Pedro Jussieu de Rezende.
Coorientador: Alexandre Xavier Falcão.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Aprendizagem não-supervisionada. 2. Floresta de caminhos ótimos. 3. Reconhecimento de padrões. 4. Análise por agrupamento. 5. Aprendizado de máquina. I. Rezende, Pedro Jussieu de, 1955-. II. Falcão, Alexandre Xavier, 1966-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Floresta de caminhos ótimos iterativa : um arcabouço para agrupamento de dados baseado em grafos
Palavras-chave em inglês:
Unsupervised learning
Optimum-path forest
Pattern recognition
Cluster analysis
Machine learning
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Pedro Jussieu de Rezende [Orientador]
Alexandru Cristian Telea
Sandra Eliza Fontes de Avila
Data de defesa: 20-08-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-2537-7050
- Currículo Lattes do autor: http://lattes.cnpq.br/8611273873930217

Universidade Estadual de Campinas
Instituto de Computação

David Aparco Cardenas

Iterative Optimum-Path Forest: A Graph-Based Data Clustering Framework

Floresta de Caminhos Ótimos Iterativa: Um Arcabouço para Agrupamento de Dados Baseado em Grafos

Campinas, 20 de agosto de 2021

Agradecimentos

I would like to express my most profound appreciation to my advisors, Prof. Pedro de Rezende and Prof. Alexandre Falcão, for their patience, motivation, support and mentorship during my entire MSc study. Without their dedication, technical insight and encouragement, none of this would have been possible. Their guidance and motivation were of the utmost importance to overcome difficulties I came across during research and writing.

I am also grateful to my colleagues from the Laboratory of Image Data Science (LIDS), whom I admire very much and shared various wonderful moments. I have to thank all the friends I made at the Institute of Computing for their support, collaboration and company during my master's years.

I would like to thank my family, particularly my parents, Fermin and Alejandra, and my sister Maria Alejandra, for their unconditional love and support.

I am thankful to Walter and his family for their warm welcome, support and affection during the first months of my arrival to Brazil.

Lastly, I would like to extend my sincere thanks to Gabriel, Bruno, George, Joaquim and Vinicius for their friendship and for being my family during my stay in Brazil.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo

Agrupamento de dados é amplamente reconhecido como uma técnica fundamental em reconhecimento de padrões e mineração de dados, sendo extensivamente utilizado em um vasto espectro de aplicações em diversos campos das ciências, negócios e engenharia. Atualmente, apesar do grande número de métodos de agrupamento já conhecidos, apenas um pequeno conjunto deles aproveita a conectividade ótima entre amostras visando obter um agrupamento mais efetivo.

Neste trabalho, apresentamos um arcabouço para agrupamento de dados baseado em grafos, chamado *Floresta de Caminhos Ótimos Iterativa* (IOPF), a qual explora a conectividade ótima para o projeto de métodos de agrupamento aprimorados. O arcabouço IOPF consiste em quatro componentes fundamentais: (*i*) amostragem de um conjunto de sementes, (*ii*) partição do grafo induzido pelas amostras da base de dados através da *Floresta de Caminhos Ótimos* (OPF) enraizada no conjunto de sementes, (*iii*) recomputação do conjunto de sementes a partir de partição prévia do grafo e, após várias iterações das duas últimas etapas, (*iv*) seleção da floresta com o menor custo total entre todas as iterações.

O arcabouço IOPF pode ser visto como uma generalização da *Floresta Geradora Iterativa* (ISF), uma metodologia proposta para segmentação de superpixels que consiste de uma sequência de *Transformadas de Imagem-Floresta* (IFTs), do domínio da imagem para o domínio do espaço de características. Além disso, exploramos o uso da estimação dinâmica de peso de arco enquanto as árvores de caminhos ótimos crescem – uma estratégia que demonstrou fornecer um delineamento mais preciso para segmentação de superpixels e segmentação interativa de objetos em trabalhos recentes.

Nesse contexto, a abordagem proposta é utilizada para projetar métodos de agrupamento aprimorados. Apresentamos quatro soluções de agrupamento baseadas no IOPF para ilustrar escolhas distintas de seus componentes constituintes. Esses métodos são subsequentemente usados na abordagem de três aplicações diferentes, a saber, segmentação de objetos não-supervisionada, análise de redes rodoviárias e agrupamento de bases de dados sintéticas bidimensionais, de modo a avaliar a efetividade dos métodos sob várias topologias de grafo, assim como para determinar sua eficácia e robustez quando comparados com baselines competitivos. Além disso, introduzimos um procedimento de seleção de sementes baseado em uma sequência de execuções do OPF, o qual fornece um conjunto apropriado de sementes iniciais que melhoram a precisão dos métodos baseados no IOPF.

Abstract

Data clustering is widely recognized as a fundamental technique in pattern recognition and data mining, being extensively used in many fields of the sciences, business and engineering, covering a broad spectrum of applications. Currently, despite the large number of clustering methods, only a few of them take advantage of optimum connectivity between samples for more effective clustering.

In this work, we present a graph-based data clustering framework, named *Iterative Optimum-Path Forest* (IOPF), that exploits optimum connectivity for the design of improved clustering methods. The IOPF framework consists of four fundamental components: (*i*) sampling of a seed set, (*ii*) partition of the graph induced from the dataset samples by an *Optimum-Path Forest* (OPF) rooted on the seed set, (*iii*) recomputation of the seed set based on the previous graph partition and, after multiple iterations of the last two steps, (*iv*) selection of the forest with the lowest total cost across all iterations.

IOPF can be seen as a generalization of the *Iterative Spanning Forest* (ISF), a framework proposed for superpixel segmentation consisting of a sequence of *Image Foresting Transforms* (IFTs), from the image domain to the feature space. Moreover, we explore the use of dynamic arc-weight estimation, as the optimum-path trees grow – a strategy that has been demonstrated to provide more accurate delineation for superpixel segmentation and interactive object segmentation in recent works.

In this context, our approach is employed to design improved clustering methods. We present four IOPF-based clustering solutions to illustrate distinct choices of its constituent components. These methods are subsequently used to address three different applications, namely, unsupervised object segmentation, road network analysis and clustering of two-dimensional synthetic datasets, in order to assess their effectiveness under various graph topologies and to ascertain their efficacy and robustness against competitive baselines. Furthermore, we introduce a seed selection procedure based on a sequence of OPF executions, which provides a suitable set of initial seeds that improve the accuracy of the IOPF-based methods.

List of Figures

List of Tables

Contents

Chapter 1

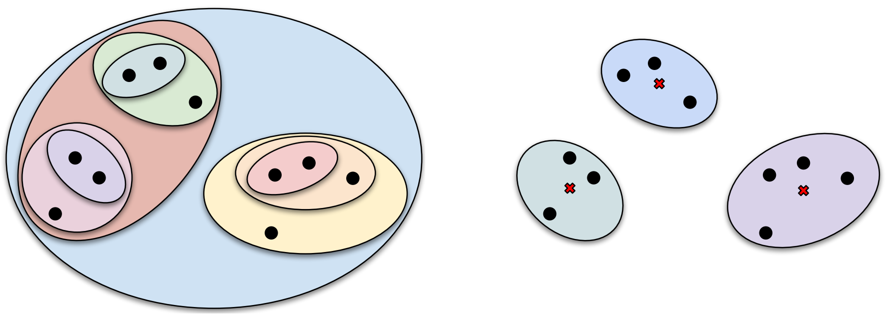
Introduction

A vast amount of data is generated by a wide range of sources in the current digital age. This data needs to be processed, analyzed, and transformed into valuable insights to support decision-making tasks. However, intensive computing resources and sophisticated techniques are required to efficiently and effectively extract the necessary information. Conventionally, these techniques are categorized into supervised, unsupervised and semi-supervised in accordance to their dependency on labeled data. Supervised and semi-supervised methods depend on labeled datasets whose construction might be time-consuming and tedious in some situations. Unsupervised approaches can considerably alleviate this problem. Among the available unsupervised techniques, data clustering has become a crucial and widely used technique to discover hidden patterns and relationships in the data. By assuming that samples in a same cluster have the same label, it can considerably reduce the dependency of labeled data.

Clustering is a fundamental process that seeks to identify the intrinsic grouping in a set of unlabeled data based on some similarity measure. The goal of clustering is to partition a set of unlabeled objects into subsets (clusters) so that those falling within the same subset are more closely related (similar) to each other than to those in different subsets. Therefore, a relevant research topic is to design practical clustering algorithms aiming to maximize intra-subset similarity and inter-subset dissimilarity according to a similarity rule. Clustering has a variety of applications in a wide range of domains, including plant and animal ecology, sequence analysis, human genetic clustering, medical imaging, market research, social network analysis, image segmentation, evolutionary algorithms, crime analysis, petroleum geology, physical geography, and so forth [26].

A significant number of algorithms addressing this problem through different approaches can be found in [69, 83]. Clustering algorithms can be comprehensively categorized into *hierarchical* and *partitional*, based on their solving strategies. Hierarchical clustering algorithms attempt to recursively find nested clusters representing the final result by means of dendrograms or trees. On the other hand, partitional clustering algorithms aim to simultaneously discover clusters by decomposing the dataset into disjoint subsets [43]. Figure 1.1 illustrates each of these categories with an example.

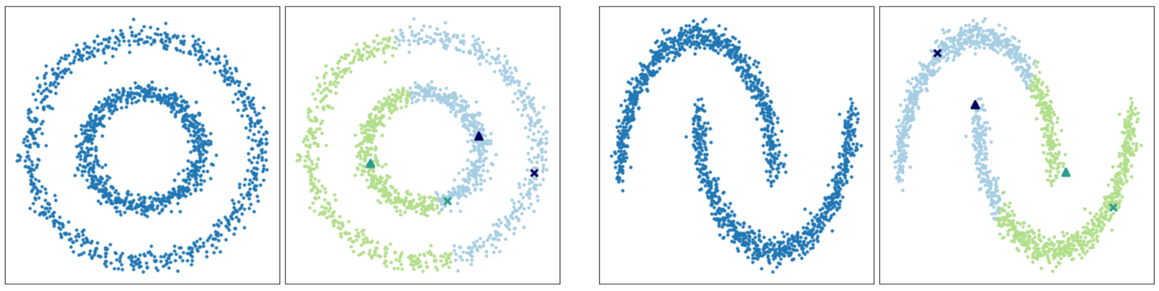
K-means, a partitional algorithm, is one of the most commonly used algorithms for clustering found in the literature, mainly because of its simplicity of implementation and intuitiveness. It is a numerical, non-deterministic, and iterative method that approximates



(a) Hierarchical clustering (b) Partitional clustering

Figure 1.1: Categorization of clustering algorithms. An example of hierarchical clustering following an agglomerative approach is shown in (a). In (b), we illustrate an example of partitional clustering through k-means, where centroids are marked with red crosses (x).

each cluster's center by representing the objects as data points in Euclidean space and measuring the dissimilarity between two points by their Euclidean distance [43]. The objective of k-means is to minimize the sum of the squared distances between every point and its nearest centroid, also known as *sum-of-squared-errors* (SSE). However, variants of k-means aim to minimize other objective functions. Despite it being widely used, k-means presents some shortcomings, such as: (*i*) the clustering result depends heavily on the initial centers as it is prone to convergence to local optima, (*ii*) it can only identify spherical-shaped clusters, and (*iii*) scales poorly for large datasets [59]. Several extensions have been proposed to overcome these limitations [29, 65], addressing, however, only a subset of these issues. Figure 1.2 illustrates how k-means is not capable of detecting nonspherical clusters.



(a) Noisy circles (b) Noisy moons

Figure 1.2: Clustering results of k-means on the noisy circles (a) and the noisy moons (b) datasets. It can be seen that k-means fails to identify nonspherical clusters.

1.1 Motivation

Recently, a graph-based iterative framework called *Iterative Spanning Forest* (ISF) [78] was proposed for superpixel segmentation. An ISF-based solution entails selecting four fundamental components: (*i*) a seed set sampling strategy, (*ii*) a connectivity function, (*iii*) an adjacency relation, and (*iv*) a seed set recomputation scheme. It consists of a sequence of executions of the *Image Foresting Transform* (IFT) [31] framework from enhanced seed sets spawned by (*iv*). Depending on the selection of the components, the user can design different ISF-based methods suitable for a particular application. The

ISF framework has proven to be effective at creating superpixel segmentation methods that are either competitive or superior to several other state-of-the-art solutions [11, 48].

On the other hand, a graph-based clustering algorithm called *Iterated Watersheds*, hereafter referred to as IW, was introduced as an extension of k-means, introducing the notion of connectivity among the samples in a dataset. As with ISF, it also works through multiple executions of the IFT framework following an iterative procedure until convergence is reached. The initial seed set is chosen randomly, and a seed set recomputation procedure is carried out at the end of each IFT execution. It has been shown to achieve superior performance compared to other state-of-the-art clustering algorithms on image segmentation and emergency station allocation problems. The IW algorithm can be regarded as an ISF-based method generalized to the feature space from its design.

Another relevant work is the *Dynamic IFT* (DynIFT) [14] framework, which extends the general IFT algorithm by dynamically estimating arc weights while extracting object information as the trees evolve from the seed set. This approach has been shown to improve object delineation in comparison to its graph-based counterparts significantly.

Accordingly, the motivation for creating a graph-based clustering framework that follows an iterative approach stems from the following considerations: in [76], the study of IW is restricted to the use of the additive connectivity function (f_{sum}) and the employment of the general IFT algorithm as the main component of the iterative solution. Thus, the use of a different connectivity function (*e.g.*, the maximum connectivity function ($f_{\max}$)) and the IFT algorithm with dynamic arc-weight estimation under an iterative clustering scheme remains unexplored. We believe that its generalization to encompass a broader set of configurations is a plausible research direction. Moreover, its applicability is restricted to graphs whose adjacency relation is either already defined or can be intuitively derived from the nature of the problem (*e.g.*, image processing). Hence, defining a graph topology for datasets with no evident relationship among the samples becomes a challenging task. Furthermore, since the clustering result of seed-based algorithms (*e.g.*, k-means) relies heavily on the choice of the initial seeds, the formulation of a seed selection strategy is necessary.

1.2 Objectives

In this context, the present work aims to explore graph-based clustering solutions for different applications through the proposal of a novel graph-based iterative clustering framework based on a sequence of *Optimum-Path Forest* executions. The first objective of this work is to formally present *Iterative Optimum-Path Forest* (IOPF), which can be regarded as a generalization of the ISF framework from the image domain to the feature space. IOPF, through the different selection of its components, is capable of creating a variety of clustering solutions that preserve connectivity among the samples of a dataset.

Once the IOPF framework has been established and explained, our second objective is to study and analyze the application of IOPF-based solutions under different graph topologies while showcasing its flexibility, extensibility, and applicability to a wide variety of problems. Furthermore, we plan to explore different approaches towards defining a graph topology based on the context of the problem. For instance, road networks or image processing problems offer enough information about the relationship among the samples in the dataset to build a graph. In contrast, generic datasets do not present an intuitive way to identify the relationships between samples, requiring the introduction of strategies to build a suitable graph topology.

Moreover, we also aim to analyze the effect of using OPF with dynamic arc-weight estimation under an unsupervised iterative scheme. Previous works [14, 12] have shown the effectiveness of using IFT with dynamic arc-weight estimation in superpixel and object segmentation. However, the generalization of this strategy to the feature space is still unaddressed.

1.3 Contributions

Our main contribution is the proposal of a graph-based clustering framework, namely Iterative Optimum-Path Forest (IOPF), which, through a sequence of OPF executions, each followed by a seed recomputation stage, aims to partition a dataset while preserving connectivity within each cluster. This framework can be regarded as a generalization of the ISF framework from the image domain to the feature space. A previous work [76] presented an algorithm with a similar formulation, which, can be viewed as an IOPF-based solution. However, it restricted its choice of components to the general IFT algorithm with the f_{sum} connectivity function. Thus, the inclusion of IFT with dynamic arc-weight estimation along with the f_{max} connectivity function as part of the set of framework components are also part of our contribution.

We explore the effectiveness of IOPF-based solutions in various applications, including allocation of emergency stations in road networks, clustering of datasets of arbitrary shapes, and object delineation. These applications will allow us to show the flexibility of the framework under different graph configurations. Therefore, our contributions also include the analysis of IOPF-based solutions under the following graph settings: (*i*) the adjacency relation and arc-weights are established from the problem definition; (*ii*) only the adjacency relation comes from the problem definition; (*iii*) neither the adjacency relation nor the arc-weights are determined from the problem definition. For the graph setting given in (*iii*), we also introduce strategies to build suitable graph topologies.

1.4 Document Organization

The remainder of this work is structured as follows: Chapter 2 starts by introducing some notations and definitions that set the ground for presenting the theoretical background to provide context to forthcoming chapters. Chapter 3 presents the related work found in the literature. The proposed framework and its constituent components are described in detail in Chapter 4. The experimental analysis and discussion of the results are presented in Chapter 5. Lastly, the conclusion and future work is stated in Chapter 6.

1.5 Concluding Remarks

In this chapter, we present a brief introduction to clustering while pointing out some of the disadvantages of k-means, one of the most popular clustering algorithms, which narrows its applicability to certain kinds of problems. Next, the motivation, objectives and contributions of this work are introduced. Lastly, we end the chapter with the document's organization of this master's work. In the next chapter, we introduce the theoretical background, which establishes the essential concepts to present the Iterative Optimum-Path Forest (IOPF) framework in Chapter 4.

Chapter 2

Theoretical Background

This chapter presents essential concepts that set the groundwork for describing the methodology proposed in future chapters. We start off by introducing some notations and definitions in Section 2.1. Next, the *Image Foresting Transform* and the *Iterative Spanning Forest*, two important graph-based techniques used for image and superpixel segmentation, are presented in Sections 2.2 and 2.3, respectively. In Section 2.4, we describe the use of Image Foresting Transform with dynamic arc-weight estimation for superpixel and interactive object segmentation. Finally, in Section 2.5, we include the description of the *Optimum-Path Forest* framework and its variants for supervised, semi-supervised, and unsupervised machine learning.

2.1 Notations and Definitions

An image is a pair $(\mathcal{D}_\mathcal{I}, \mathbf{I})$, whose domain $\mathcal{D}_\mathcal{I} \subset \mathbb{Z}^n$, for $n \in \mathbb{N}^*$, is comprised of spatial elements, called pixels (*picture elements*), or more generically known as *spels* (*spatial elements*), and $\mathbf{I} : \mathcal{D}_\mathcal{I} \to \mathbb{R}^m$, for $m \in \mathbb{N}$, a function that maps each pixel $p \in \mathcal{D}_\mathcal{I}$ to a vector of image features (*e.g.*, color space components). In this work, we use two-dimensional colored images. Therefore, each color pixel $p \in \mathbb{Z}^2$ is a bidimensional position vector and $\mathbf{I}(p) \in \mathbb{R}^3$ assigns to each pixel a triplet containing the components of a color space (*e.g.*, CIELAB).

The norm $\| \cdot \|_i, i \in \mathbb{N}$, of an m-dimensional vector $v = \{x_1, \dots, x_m\}$, is denoted as

$$\|v\|_i = \left(\sum_{j=0}^{m} (x_j)^i \right)^{\frac{1}{i}} \tag{2.1}$$

Thus, the notations $\|p-q\|_1$ and $\|p-q\|_2$ stand for the *Manhattan* and *Euclidean* distance, respectively, between two pixels $p, q \in \mathcal{D}_\mathcal{I}$. The same notation applies to the feature vectors of the image – *i.e.*, $\|\mathbf{I}(p) - \mathbf{I}(q)\|_1$ and $\|\mathbf{I}(p) - \mathbf{I}(q)\|_2$.

An image I can be rendered as a directed graph $(\mathcal{N}, \mathcal{A})$ under various configurations, depending upon how nodes $\mathcal{N} \subseteq \mathcal{D}_\mathcal{I}$ and edges are defined. In the present work, we define nodes as pixels ($\mathcal{N} = \mathcal{D}_\mathcal{I}$) and edges are determined by $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$, an irreflexive binary relation on $\mathcal{N}$ (*i.e.*, *adjacency relation*); thus $p, q \in \mathcal{D}_\mathcal{I}$ are considered adjacent pixels if $(p, q) \in \mathcal{A}$.

An adjacency relation can be established in many different ways. However, in this work, we use the *Euclidean adjacency relation* $\mathcal{A}^r$, for a given radius $r \in \mathbb{R}$, defined as

$$\mathcal{A}^r = \{(p, q) \in \mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}} \mid \|q - p\|_2 \leq r\} \tag{2.2}$$

A *path* π is defined as a finite sequence of distinct adjacent pixels in the image graph – *i.e.*, $\pi = \langle p_1, p_2, \ldots, p_n \rangle$, such that $(p_i, p_{i+1}) \in \mathcal{A}$ for $1 \leq i < n$. A path is *simple* if $n > 1$, otherwise, it is *trivial*. The pixels p_1 and p_n are designated as *start* and *terminus* of path π, and this can be denoted as either $\pi_{p_1 \rightsquigarrow p_n}$ or π_{p_n} with root $\mathbf{R}(p_n) = p_1$. A *cycle* is a simple path where the start and terminus pixels are the same – *i.e.*, $\pi_{p \rightsquigarrow p}$. Let Π be the set of all possible paths in the image graph. A pixel q is *connected* to a pixel $p \neq q$ if there is a path $\pi_{p \rightsquigarrow q} \in \Pi$. Furthermore, the *concatenation* of two paths $\pi_p \in \Pi$ and $\tau_{p \rightsquigarrow q} \in \Pi$ is denoted by $\pi_p \cdot \tau_{p \rightsquigarrow q}$ with the two occurrences of p merged into one. An edge with endpoints p and q is given by $\langle p, q \rangle$, while the concatenation of π_p and $\langle p, q \rangle$ is denoted as $\pi_p \cdot \langle p, q \rangle$.

A *subgraph* of ($\mathcal{N}$, $\mathcal{A}$) is a pair ($\mathcal{N}'$, $\mathcal{A}'$), such that $\mathcal{N}' \subseteq \mathcal{N}$ and $\mathcal{A}' \subseteq \mathcal{A}$. Let Π' the set of all possible paths in $(\mathcal{N}', \mathcal{A}')$, then the subgraph ($\mathcal{N}'$, $\mathcal{A}'$) is said to be *connected* if $\forall p, q \in \mathcal{N}'$ there is a path $\pi_{p \rightsquigarrow q} \in \Pi'$. A subgraph is *spanning* if $\mathcal{N}' = \mathcal{N}$ and is *acyclic* if it contains no cycles. A connected and acyclic subgraph is a *tree* and a collection of trees is called a *forest*. A *partition* of the pixels of an image graph is a set of connected subgraphs ($\mathcal{N}'_i$, $\mathcal{A}'_i$), $i \in \{1, \ldots, n\}$, such that $\cup_{i=1}^{n} \mathcal{N}'_i = \mathcal{N}$ and $\cap_{i=1}^{n} \mathcal{N}'_i = \emptyset$.

2.2 Image Foresting Transform

The *Image Foresting Transform* (IFT) [31] is a framework for the design, implementation and evaluation of image processing operators based on optimum connectivity between pixels. The IFT reduces optimal image partition problems to a shortest-path forest problem in a graph derived from the image (*image graph*), leading to a unified and efficient approach for the design of image processing operators.

In most applications, the implementation of IFT algorithms can achieve linear time complexity (proportional to the number of vertices in the graph). A *path-cost function* associates a value to any path $\pi \in \Pi$, measuring how strongly connected the start and terminus nodes are. Given an image graph ($\mathcal{N}$, $\mathcal{A}$) and a path-cost function f, the IFT minimizes a path-cost map $C : \mathcal{N} \rightarrow \mathbb{R}$,

$$C(q) = \min_{\forall \pi_q \in \Pi_q} \{f(\pi_q)\}, \tag{2.3}$$

where Π_q denotes the set of all possible paths in the graph with terminus q.

The IFT is a practical tool whose effectiveness, efficiency and flexibility have been widely demonstrated in various applications. In [51], Lotufo *et al.* introduced the IFT-watersheds from gray-scale marker exploiting the IFT to improve the efficiency of watershed transforms. A work by Falcão *et al.* [32] leverages the IFT for the design of the following connected image operators: cutting-off-domes and filling-up-basins. Moreover, the IFT has also been used to address the problem of one-pixel-wide connected multiscale skeletonization in [33]. In [16], the IFT is used for the characterization of the cortex in cerebral MR images using a combination of two different procedures: an IFT-based watershed-from-markers transform and IFT-based skeletonization.

The definition of a path-cost function suitable to the addressed problem is a critical step towards creating an effective IFT-based solution. For instance, in image segmentation, it is expected that pixels located in a region of uniform texture be strongly connected in the image graph.

Let $\mathcal{S} \subset \mathcal{D}_{\mathcal{I}}$ be a set of seed pixels. To define a path-cost function (connectivity function) $f_* : \Pi \to \mathbb{R}$ for any path $\pi \in \Pi$, the IFT algorithm starts with trivial paths for all nodes $q \in \mathcal{D}_{\mathcal{I}}$, whose path-cost $f_*(\langle p \rangle)$ is given by the following rule:

$$f_*(\langle q \rangle) = \begin{cases} 0 & \text{if } q \in \mathcal{S} \subset \mathcal{D}_{\mathcal{I}} \\ +\infty & \text{otherwise.} \end{cases} \tag{2.4}$$

During the execution of the algorithm, the current path π_q is replaced by $\pi_p \cdot \langle p, q \rangle$ for $(p, q) \in \mathcal{A}$, whenever $f(\pi_q) > f(\pi_p \cdot \langle p, q \rangle)$ and the cost map $C(q)$ is updated to $f(\pi_p \cdot \langle p, q \rangle)$. Accordingly, since all paths stem from $\mathcal{S}$, the seeds become the roots of the trees comprising the forest.

The similarity of color between two adjacent pixels can be defined by a non-negative function $w : \mathcal{A} \to \mathbb{R}$. For instance, in Equation 2.5, the arc-weight for $(p, q) \in \mathcal{A}$ is determined from the Euclidean distance between the pixels' feature vectors.

$$w(p, q) = \|\mathbf{I}(p) - \mathbf{I}(q)\| \tag{2.5}$$

It is easy to see that a *connectivity function* f_* can then recursively assign a cost to any $\pi \in \Pi$. As an example to illustrate this assignment, we use the function $f_{\max}$, defined in Equation 2.6, which returns the largest arc-weight in a path.

$$\begin{aligned} f_{\max}(\langle q \rangle) &= \begin{cases} 0 & \text{if } q \in \mathcal{S} \subset \mathcal{D}_{\mathcal{I}} \\ +\infty & \text{otherwise} \end{cases} \\ f_{\max}(\pi_p \cdot \langle p, q \rangle) &= \max\{f_{\max}(\pi_p), w(p, q)\}, \end{aligned} \tag{2.6}$$

A path π_p is called *optimum* if $f_*(\pi_p) \leq f_*(\tau_p)$ for any other path $\tau_p \in \Pi$, regardless of its starting node. The IFT algorithm partitions the graph into an *optimum-path forest* P through the minimization of the path-cost map $C(q) = \min_{\forall \pi_q \in \Pi_q}\{f_*(q)\}$ using the recursive rule above. $P : \mathcal{D}_{\mathcal{I}} \to \mathcal{D}_{\mathcal{I}} \cup \{nil\}$ is an *acyclic predecessor map* that attributes to each node $p \in \mathcal{D}_{\mathcal{I}}$ either a predecessor $P(p) \in \mathcal{D}_{\mathcal{I}}$ in the optimum-path π_p or a distinguished marker $P(q) = nil \notin \mathcal{D}_{\mathcal{I}}$ if $p \in \mathcal{S}$. A *root map* $R : \mathcal{D}_{\mathcal{I}} \to \mathcal{D}_{\mathcal{I}}$ may also be generated either recursively from the predecessor map or iteratively during the execution of the algorithm.

$$R(q) = \begin{cases} q & \text{if } P(q) = nil \\ R(p) & \text{if } P(q) = p \neq nil \end{cases} \tag{2.7}$$

If the connectivity function f_* does not satisfy the properties of a *smooth-function* [20], the generated forest P is not an optimum-path forest but a spanning forest whose every tree $\mathcal{T} = (\mathcal{N}^{\mathcal{T}}, \mathcal{A}^{\mathcal{T}})$, of root $p \in \mathcal{D}_{\mathcal{I}}$, has its vertex set and adjacency relation given by:

$$\begin{aligned} \mathcal{N}^{\mathcal{T}} &: \{q \in \mathcal{D}_{\mathcal{I}} \mid R(q) = p\} \\ \mathcal{A}^{\mathcal{T}} &: \{(P(q), q) \in \mathcal{A} \mid p \neq q \in \mathcal{D}_{\mathcal{I}}\}, \end{aligned} \tag{2.8}$$

Mansilla *et al.* [55] studied the application of the IFT with non-smooth connectivity functions for the 3D segmentation of MR images, demonstrating outstanding results and suggesting that a spanning forest may also be used to address different problems due to its interesting properties.

Following a procedure similar to the one used to generate the root map, a distinct label $\lambda(p) \in \{1, \ldots, |\mathcal{S}|\}$ can be assigned to each seed $p \in \mathcal{S}$, which is then propagated to its most closely connected pixels during the execution of the algorithm or via the root map. The latter procedure generates a label map $L : \mathcal{D}_{\mathcal{I}} \rightarrow \{1, \ldots, |\mathcal{S}|\}$ defined by

$$L(q) = \begin{cases} \lambda(q) & \text{if } q \in \mathcal{S} \\ \lambda(R(q)) & \text{otherwise.} \end{cases} \tag{2.9}$$

The IFT is a generalization of Dijkstra's algorithm to multiple sources and more general path-value functions [20]. The IFT employs a *priority queue* $\mathcal{Q} \subseteq D_I$ to compute a predecessor map P rooted in the elements of $\mathcal{S}$ with respect to a connectivity function f_*. At the end of the procedure, a root map R and a label map L are generated, where the latter represents the image segmentation.

Algorithm 1: Image Foresting Transform (**IFT**)

```
Input    : Image $I = (\mathcal{D}_{\mathcal{I}}, \mathbf{I})$, adjacency relation $\mathcal{A}$, connectivity function $f_*$, seed
           set $\mathcal{S}$, and labeling function $\lambda : \mathcal{S} \rightarrow \{1, \ldots, |\mathcal{S}|\}$
Output   : Predecessor map $P$, root map $R$, cost map $C$, and label map $L$
Auxiliar: Priority queue $\mathcal{Q}$ and variable $tmp$

1  $\mathcal{Q} = \emptyset$
2  foreach $q \in \mathcal{D}_{\mathcal{I}}$ do
3      $R(q) \leftarrow q$, $P(s) \leftarrow nil$
4      $C(q) \leftarrow +\infty$, $L(q) \leftarrow 0$
5      if $q \in \mathcal{S}$ then
6          $C(q) \leftarrow 0$, $L(q) \leftarrow \lambda(q)$
7      Insert $q$ into $\mathcal{Q}$
8  while $\mathcal{Q} \neq \emptyset$ do
9      Remove $p$ from $\mathcal{Q}$, such that $p = \text{argmin}_{q \in \mathcal{Q}}\{C(q)\}$
10     foreach $(p, q) \in \mathcal{A} \mid q \in \mathcal{Q}$ do
11         $tmp \leftarrow f_*(\pi_p \cdot \langle p, q \rangle)$
12         if $tmp < C(q)$ then
13             $R(q) \leftarrow R(p), P(q) \leftarrow p$
14             $C(q) \leftarrow tmp$, $L(q) \leftarrow L(p)$
```

Algorithm 1 depicts the pseudocode of the IFT for a connectivity function f_*. Lines 3–

7 initialize the maps for node $q \in D_I$. Next, q is inserted into the priority queue $\mathcal{Q}$ in Line 7. The aforementioned procedure is carried out for all nodes $q \in \mathcal{D_I}$. Lines 8–14 are responsible for the graph partition into a spanning forest. In Lines 13–14, a pixel q is conquered by p (*i.e.*, label propagation from p to q) if the extended path's cost evaluated by $f_*(\pi_p \cdot \langle p, q \rangle)$ is lower than the current optimal cost $C(q)$. Next, the maps' values corresponding to q are updated accordingly. Eventually, all pixels will be conquered by some seed, partitioning the graph into either an optimum-path forest if f_* is a smooth-function or a spanning forest otherwise.

2.3 Iterative Spanning Forest

Vargas-Muñoz *et al.* introduced a framework called *Iterative Spanning Forest* (ISF) [78] for superpixel segmentation. The ISF is based on a sequence of Image Foresting Transforms consisting in the selection of four components: (*i*) a seed sampling strategy; (*ii*) a connectivity function; (*iii*) an adjacency relation; and (*iv*) a seed recomputation procedure. The framework starts with an initial set of seeds obtained by a given strategy. Next, a sequence of IFT executions, each of them followed by a seed recomputation procedure, is carried out for a fixed number of iterations. Each IFT execution partitions the graph into a set of spanning trees (superpixels) rooted in the seed set. Next, the seed recomputation procedure outputs a seed set, based on the previous partition, to generate an improved set of connected superpixels (supervoxels in 3D) per iteration. Such a framework permits creating several solutions by combining different strategies for its individual components, such as seed estimation, seed recomputation, and arc-weight estimation for a given connectivity function. As such, this work can be seen as an extension of the ISF to the feature space. Figure 2.1 illustrates the general flowchart of the ISF framework.

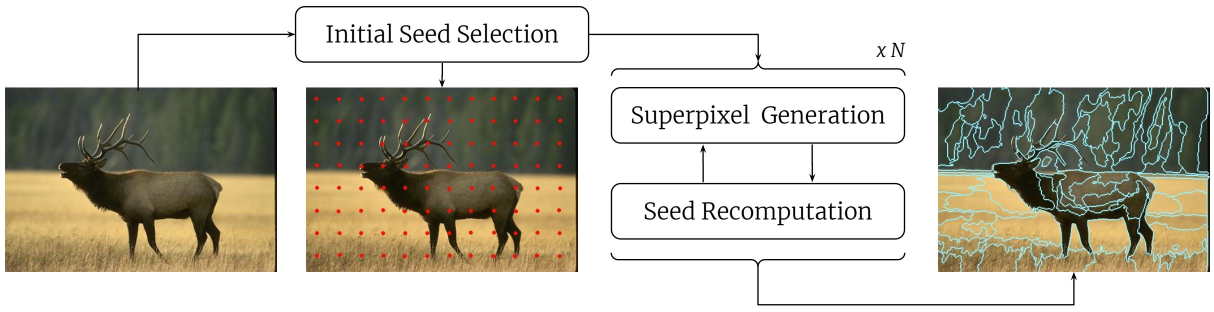


Figure 2.1: Flowchart of the Iterative Spanning Forest framework.

Due to its flexibility and proven effectiveness, the ISF has been extended and used to create solutions for various applications. In [17], Castelo-Fernandez and Falcão extended the ISF to generate superpixels from multiple images of a same class for interest point detection, to build class-specific visual dictionaries subsequently. Martins *et al.* [56] introduced SymmISF, a supervoxel segmentation method based on the ISF, which extracts symmetrical supervoxels from left and right brain hemispheres aiming to identify supervoxels with abnormal asymmetries in MR images of the brain in an unsupervised fashion. Another work by Galvão *et al.* [48], namely *Recursive Iterative Spanning Forest* (RISF), extends the ISF to generate multi-scale superpixel segmentation over region adjacency graphs (RAGs), obtaining results in execution times and boundary adherence superior to existing ISF-based methods, without compromising the benefits of the original framework. Belém *et al.* presented *Object-based ISF* (OISF) [11], an extension of the ISF framework for superpixel segmentation that incorporates object information from an object saliency, improving the boundary adherence when compared to other state-of-the-art methods.

2.3.1 Seed Selection

Superpixel segmentation methods require unsupervised and automatic techniques for sampling the seed set $\mathcal{S}$. The challenge of these strategies lies in the fact that seeds should be selected from regions within the boundary of the objects of interest, leading to precise boundary adherence while avoiding the "leakage" of superpixels near the boundaries. The IFT performs superpixel delineation by seed competition, where each superpixel is represented as an optimum-path forest rooted in its internal seed. Thus, in order to guarantee one seed per object, including the background, the number of seeds should be significantly greater than the number of objects.

2.3.2 Grid Sampling

In *grid sampling* [1] (GRID), seeds are collected at regularly spaced intervals in the image so that superpixels are generated with approximately the same sizes and to guarantee the presence of seeds within the objects of interest, given that no prior information is available about their location. This sampling strategy has been broadly used in several studies [1, 2, 49, 50, 78, 82] due to its intuitiveness and ease of implementation.

In order to sample a given number $k \in \mathbb{N}$ of seeds, the GRID strategy first determines an estimate of the area of superpixels z (in number of pixels) with respect to the total size of image I (total number of pixels in I).

$$z = \frac{|\mathcal{D}_\mathcal{I}|}{k} \tag{2.10}$$

Therefore, to comply with this size criterion, seeds must be spaced $d = \sqrt{z}$ pixels apart to attain a regular seed distribution on the image surface. Such criterion prevents the selection of seeds close to the image boundary, which may compromise the homogeneity of superpixel regions. Thus, every seed s estimated using the criterion mentioned above is collected from regions with low color variance with respect to an adjacency relation $\mathcal{A}^1$

Algorithm 2 depicts the pseudocode for the GRID strategy for a given *gradient function* $\nabla\mathbf{I}$ defined over pixel intensities in $\mathcal{D}_\mathcal{I}$. Line 2 calculates the distance d between seeds to guarantee equidistant spatial grid representation. Lines 4–12 carry out the sampling of the elements of $\mathcal{S}$. In Lines 7–10, for each pixel $p \in \mathcal{D}_\mathcal{I}$ in Line 7, the pixel to be collected as the next seed is the one with the lowest gradient among the pixels in the adjacency set $\mathcal{A}^1(p)$. Figure 2.2 illustrates two examples of grid sampling with different values of k.

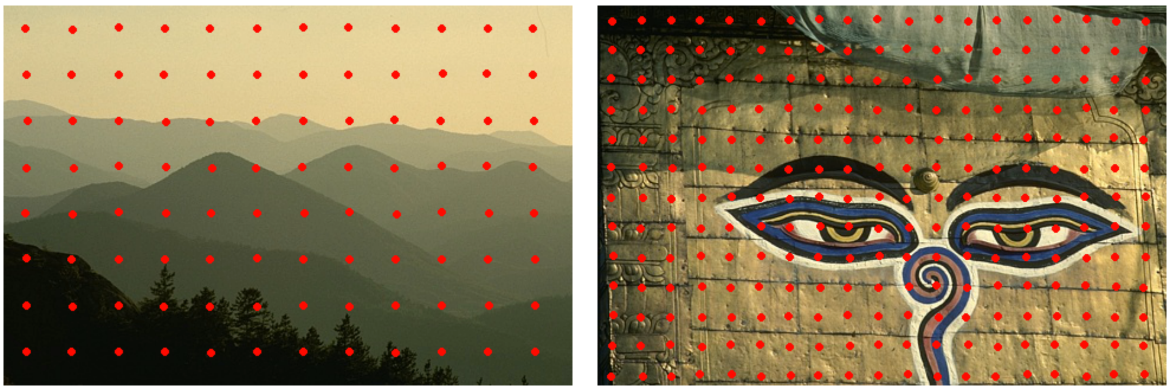

(a) Grid sampling ($k = 100$) (b) Grid sampling ($k = 250$)

Figure 2.2: Grid sampling examples with different values of k. In (a) we use $k = 100$ and in (b) we use $k = 250$.

2.3.3 Mixed Sampling

A natural image is typically composed of objects at many different scales presenting strong correlations within objects while having weaker correlations between objects, leading to regions in the image with high-intensity variance. In these cases, a more robust seed sampling strategy is required. Mixed sampling (MIX) aims to alleviate this problem by sampling more seeds from regions with high entropy while keeping the grid distribution of seeds over the image to guarantee the regularity of superpixel shapes. In some datasets, the use of this strategy led to an improvement in boundary adherence after superpixel segmentation.

A *quadtree* $\mathcal{G}$ is a tree-like data structure built by a recursive decomposition of the image domain into quadrants (*i.e.*, nodes) $Q \subset \mathcal{D}_\mathcal{I}$ where each node either terminates on a leaf or branches into four sub-level quadtrees for each hierarchy level.

Algorithm 2: Grid sampling **GRID**

```
Input  : Image $I = (\mathcal{D}_\mathcal{I}, \mathbf{I})$ with dimensions $n_x \times n_y$, gradient function $\nabla\mathbf{I}$, and
         number of superpixels $k$
Output : Seed set $\mathcal{S}$
1  $\mathcal{S} \leftarrow \emptyset$
2  $z \leftarrow \frac{|\mathcal{D}_\mathcal{I}|}{k}$, $d \leftarrow \sqrt{z}$
3  $x \leftarrow \lfloor \frac{d}{2} \rfloor$
4  while $x < n_x$ do
5      $y \leftarrow \lfloor \frac{d}{2} \rfloor$
6      while $y < n_y$ do
7          $p \leftarrow (x, y)$
8          if $p \in \mathcal{D}_\mathcal{I}$ then
9              $s \leftarrow \text{argmin}_{\forall q \in \mathcal{A}^1(p)} \{\|\nabla\mathbf{I}(q)\|_2\}$
10             $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
11         $y \leftarrow y + \lfloor \frac{d}{2} \rfloor$
12     $x \leftarrow x + \lfloor \frac{d}{2} \rfloor$
```

The MIX algorithm computes the *normalized Shannon's entropy* (NSE) to measure the degree of heterogeneity in a quadrant $Q \in \mathcal{G}$. Equation 2.11 expresses the formula to determine the NSE as a function $\mathbf{E} : \mathcal{G} \to \mathbb{R}$:

$$\mathbf{E}(Q) = -\frac{\sum_{i=1}^{n} \mathbf{p}(i) \log_2(\mathbf{p}(i))}{\log_2(n)} \tag{2.11}$$

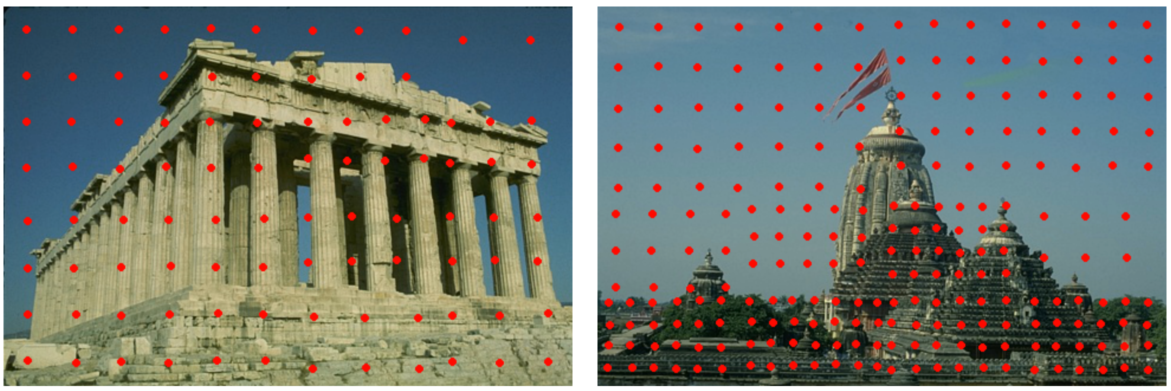
where $\mathbf{p} : \mathbb{R} \to [0, 1]$ is the probability of occurrence of intensity i among the n pixels within quadrant Q. Once $\mathbf{E}$ is computed for all $Q \in \mathcal{G}$, we evaluate the mean $\mu(\mathbf{E})$ and standard deviation $\sigma(\mathbf{E})$ for the four quadrants in each hierarchy level (see Equation 2.12).

$$\begin{aligned} \mu(\mathbf{E}) &= \frac{\sum_{i=1}^{4} \mathbf{E}(Q_i)}{4} \\ \sigma(\mathbf{E}) &= \sqrt{\frac{\sum_{i=1}^{4} (\mathbf{E}(Q_i) - \mu(\mathbf{E}))^2}{3}}. \end{aligned} \tag{2.12}$$

To generate the next hierarchy level, we evaluate the following expression $\|\mathbf{E}(Q) - \mu(\mathbf{E})\|_1 > \sigma(\mathbf{E})$ for quadrant Q. If the inequality holds, then Q is divided into four subquadrants followed by the calculation of their entropies. Once the second hierarchy has been defined, a number k' of seeds, directly proportional to the leaf's entropy, is computed for each leaf in the set of leaves $\mathcal{L}$ of the quadtree $\mathcal{G}$ (see Equation 2.13).

$$k' = k \cdot \frac{\mathbf{E}(Q)}{\sum_{Q' \in \mathcal{L}} \mathbf{E}(Q')} \tag{2.13}$$

Lastly, sets of k' seeds are sampled from each leaf quadrant using the GRID strategy, and the union of these sets comprises the final seed set $\mathcal{S}$. Figure 2.3 illustrates two examples of mixed sampling with different values of k.



(a) Mixed sampling ($k = 100$)   (b) Mixed sampling ($k = 250$)

Figure 2.3: Mixed sampling examples with different values of k. In (a) we use $k = 100$ and in (b) we use $k = 250$.

Algorithm 3 depicts the pseudocode for the MIX strategy. The first level of $\mathcal{G}$ is defined in Line 3 while the entropy for each quadrant is calculate in Lines 4–5. Next, mean and standard deviation for the quadrant's entropy are computed in Lines 6 and 7, respectively. In Lines 8–14, the entropy of each quadrant is evaluated to identify whether a subsequent partition is required to construct the second level of $\mathcal{G}$. Lastly, once $\mathcal{G}$ has been constructed, the computation of the number of seeds (*i.e.*, k') to be collected from each leaf, and their respective sampling through the GRID strategy are carried out in Lines 15–17.

Algorithm 3: Mix sampling (**MIX**)

```
Input  : Image $I = (\mathcal{D}_\mathcal{I}, \mathbf{I})$, gradient function $\nabla\mathbf{I}$, and number of superpixels $k$
Output : Seed set $\mathcal{S}$
1  $\mathcal{S} \leftarrow \emptyset$
2  Partition $\mathcal{D}_\mathcal{I}$ into four quadrants $\mathcal{G} \leftarrow \{Q_1, Q_2, Q_3, Q_4\}$
3  $x \leftarrow \frac{d}{2}$
4  foreach $Q \in \mathcal{G}$ do
5      Calculate $\mathbf{E}(Q) \leftarrow -\frac{\sum_{i=1}^{n} \mathbf{p}(i)\log_2(\mathbf{p}(i))}{\log_2(n)}$ for pixel intensities $i$ from $(Q, \mathbf{I})$.
6  $\mu(\mathbf{E}) \leftarrow \frac{\sum_{j=1}^{4} \mathbf{E}(Q_j)}{4}$
7  $\sigma(\mathbf{E}) \leftarrow \sqrt{\frac{\sum_{j=1}^{4}(\mathbf{E}(Q_j)-\mu(\mathbf{E}))^2}{3}}$
8  foreach $u \in \{1,2,3,4\}$ do
9      if $\|\mathbf{E}(Q_u) - \mu(\mathbf{E})\|_1 > \sigma(\mathbf{E})$ then
10         Partition $Q_u$ into four quadrants $Q_u = \{Q_{u,1}, Q_{u,2}, Q_{u,3}, Q_{u,4}\}$
11         Remove $Q_u$ from $\mathcal{G}$
12         foreach $v \in \{1,2,3,4\}$ do
13             $\mathcal{G} \leftarrow \mathcal{G} \cup Q_{u,v}$
14             $\mathbf{E}(Q_{u,v}) \leftarrow -\frac{\sum_{i=1}^{n} \mathbf{p}(i)\log_2(\mathbf{p}(i))}{\log_2(n)}$ for pixel intensities $i$ from $(Q_{u,v}, \mathbf{I})$.
15 foreach $Q \in \mathcal{L}$ do
16     $k' \leftarrow k \cdot \frac{\mathbf{E}(Q)}{\sum_{Q' \in \mathcal{L}} \mathbf{E}(Q')}$
17     $\mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{GRID}((Q, \mathbf{I}), \nabla\mathbf{I}, k')$
```

2.3.4 Superpixel Segmentation

The original ISF paper [78] considered three connectivity functions, namely f_1, f_2, and f_3 for the computation of the IFT. The first two of them, namely f_1 and f_2, are based on an additive connectivity function with different arc-weight functions to control superpixel regularity and boundary adherence. Even though these functions do not guarantee an optimum-path forest, they can efficiently deal with the problem of intensity inhomogeneity [55]. On the other hand, the third connectivity function (*i.e.*, f_3), based on the maximum connectivity function, guarantees the generation of an optimum-path forest since it meets the conditions of a smooth function. However, ISF-based methods that incorporate this connectivity function presented inferior performance than their counterparts, as is demonstrated in [78].

For a given seed set $\mathcal{S} \subset \mathcal{D}_\mathcal{I}$, the additive connectivity function, f_{sum}, is defined recursively in Equation 2.14 for an arc-weight function w.

$$
\begin{aligned}
f_{\text{sum}}(\langle q\rangle) &= \begin{cases} 0 & \text{if } q \in \mathcal{S} \subset \mathcal{D_I} \\ +\infty & \text{otherwise.} \end{cases} \\
f_{\text{sum}}(\pi_p \cdot \langle p, q\rangle) &= f_{\text{sum}}(\pi_p) + w(p,q),
\end{aligned} \tag{2.14}
$$

where $(p, q) \in \mathcal{A}$. Moreover, let $s^i \in \mathcal{S}$ for iteration $i \geq 1$ during the ISF execution, and $\mu(\mathcal{T})$ the function that computes the mean feature vector of tree $\mathcal{T} = (\mathcal{D_I}^{\mathcal{T}}, \mathcal{A}^{\mathcal{T}})$ defined in Equation 2.15.

$$
\mu(\mathcal{T}) = \frac{\sum_{p \in \mathcal{D_I}^{\mathcal{T}}} \mathbf{I}(p)}{\|\mathcal{D_I}^{\mathcal{T}}\|} \tag{2.15}
$$

Next, we define the function $M : \mathcal{S} \rightarrow \mathbb{R}^m$ that maps a seed s^i to a representative feature vector, which, according to Equation 2.16, can either be the same seed if $i = 1$ or be derived from the tree $\mathcal{T}_{s^{i-1}}$ generated in the previous iteration of the ISF execution.

$$
M(s^i) = \begin{cases} \mathbf{I}(s^i) & \text{if i = 1} \\ \mu(\mathcal{T}_{s^{i-1}}) & \text{otherwise.} \end{cases} \tag{2.16}
$$

Based on these definitions, we define the arc-weight functions w_1 and w_2 in Equation 2.17, where $\alpha \geq 0$ and $\beta \geq 1$ control the compromise between superpixel regularization and boundary adherence, respectively. The value chosen for α is inversely proportional to the regularity of superpixels compromising the boundary adherence. Furthermore, for a suitable value of α, the value for β is chosen directly proportional to the boundary adherence.

$$
\begin{aligned}
w_1(p,q) &= (\alpha\|\mathbf{I}(R(p)) - \mathbf{I}(q)\|_2)^{\beta} + \|p - q\|_2 \\
w_2(p,q) &= (\alpha\|M(R(p)) - \mathbf{I}(q)\|_2)^{\beta} + \|p - q\|_2
\end{aligned} \tag{2.17}
$$

The aforementioned arc-weight functions can be plugged into f_{sum} to obtain f_1 and f_2 as described by Equation 2.18.

$$
\begin{aligned}
f_*(\langle q\rangle) &= \begin{cases} 0 & \text{if } q \in \mathcal{S} \subset \mathcal{D_I} \\ +\infty & \text{otherwise.} \end{cases} \\
f_1(\pi_p \cdot \langle p, q\rangle) &= f_1(\pi_p) + w_1(p,q) \\
f_2(\pi_p \cdot \langle p, q\rangle) &= f_2(\pi_p) + w_2(p,q)
\end{aligned} \tag{2.18}
$$

The third connectivity function, f_3, is based on the $f_{\max}$ connectivity function and defined in Equation 2.19. The connectivity function f_3 substitutes the arc-weight $w(p, q)$ in the original definition of f_{max} (see Equation 2.6) by the value of the gradient image for pixel q represented by $D(q)$, while keeping the same initialization function.

$$
f_3(\pi_p \cdot \langle p, q\rangle) = \max\{f_3(\pi_p), D(q)\}. \tag{2.19}
$$

2.3.5 Seed Recomputation

The recomputation procedure is carried out after the superpixel segmentation by means of a *recomputation function* $r_* : \mathcal{S} \rightarrow \mathcal{D_I}$, which determines a set of seeds $\mathcal{S}' \subseteq \mathcal{D_I}$ that are the seeds for the next iteration. This procedure aims to improve the representativity for each superpixel by selecting the seeds that most likely will provide optimum paths with less cost than the seed set from the previous iteration, therefore, producing more homogeneous superpixels.

An intuitive strategy to address the selection of $\mathcal{S}'$ is through the identification of the pixel closest, in terms of the Euclidean distance, to the centroid of each superpixel. Let $\mathcal{T}_s = (\mathcal{D_I}^{\mathcal{T}_s}, \mathcal{A}^{\mathcal{T}_s})$ be the tree rooted in $s \in \mathcal{S}$ and generated in the previous iteration. Then, a new seed is calculated through the function r_1 defined in Equation 2.20.

$$\begin{aligned} r_1(s) &= \underset{p \in \mathcal{D_I}^{\mathcal{T}_s}}{\text{argmin}}\{\|p - c\|_2\} \\ c &= \frac{1}{|\mathcal{D_I}^{\mathcal{T}_s}|} \sum_{p \in \mathcal{D_I}^{\mathcal{T}_s}} p \end{aligned} \tag{2.20}$$

Despite the intuitive appeal of the strategy mentioned above, r_1 does not consider the distribution of the feature vectors of the pixels within $\mathcal{T}_s$ in the feature space. Following this line of thought, $\mathcal{S}$ can also be obtained using the function r_{med} that selects the pixel whose feature vector is the closest, in terms of the Euclidean distance, to the mean feature vector of the tree (superpixel) to which it belongs. The function r_2 formally implements this strategy in Equation 2.21.

$$\begin{aligned} r_2(s) &= \underset{p \in \mathcal{D_I}^{\mathcal{T}_s}}{\text{argmin}}\{\|\mathbf{I}(p) - \mu(\mathcal{T}_s)\|_2\} \\ \mu(\mathcal{T}_s) &= \frac{1}{|\mathcal{D_I}^{\mathcal{T}_s}|} \sum_{p \in \mathcal{D_I}^{\mathcal{T}_s}} \mathbf{I}(p) \end{aligned} \tag{2.21}$$

Seeds of subsequent iterations may not significantly alter their position, leading to insignificant differences in the delineation of their corresponding superpixels. Therefore, an additional criterion is introduced to determine whether a new seed will be generated for a given superpixel to accelerate the convergence process. Let $m_c, m_s \in \mathbb{R}$ be two thresholds computed using the feature vectors and coordinates of pixels, respectively, defined in Equation 2.22.

$$\begin{aligned} m_c &= \frac{\sum_{p \in \mathcal{D_I}} \|\mathbf{I}(p) - \mathbf{I}(R(p))\|_2}{|\mathcal{D_I}|} \\ m_s &= \frac{\sum_{p \in \mathcal{D_I}} \|p - R(p)\|_2}{|\mathcal{D_I}|} \end{aligned} \tag{2.22}$$

Thus, a seed s will be recomputed for a given superpixel if any of the following conditions presented in Inequalities 2.23 is met; otherwise, the same seed s is kept for the next iteration.

$$\begin{aligned} \sqrt{m_c} &< \|\mathbf{I}(s) - \mathbf{I}(r_*(s))\|_2 \\ \sqrt{m_s} &< \|s - r_*(s)\|_2 \end{aligned} \tag{2.23}$$

The pseudocode for this seed recomputation strategy is depicted in Algorithm 4. Lines 2 and 3 compute the thresholds m_c and m_s, respectively, while Lines 4–8 evaluate whether the new seed $r_*(s)$ will be included in $\mathcal{S}'$ (Line 6) or if s will be kept for the next iteration (Line 8).

Algorithm 4: Seed recomputation (**SEEDRECOMP**)

```
Input  : Image $I = (\mathcal{D}_\mathcal{I}, \mathbf{I})$, seed set $\mathcal{S}$, recomputation function $r_*$, and root
         map $R$
Output : Seed set $\mathcal{S}'$
1 $\mathcal{S}' \leftarrow \emptyset$
2 $m_c \leftarrow \frac{\sum_{p \in \mathcal{D}_\mathcal{I}} \|\mathbf{I}(p) - \mathbf{I}(R(p))\|_2}{|\mathcal{D}_\mathcal{I}|}$
3 $m_s \leftarrow \frac{\sum_{p \in \mathcal{D}_\mathcal{I}} \|p - R(p)\|_2}{|\mathcal{D}_\mathcal{I}|}$
4 foreach $s \in \mathcal{S}$ do
5     if $\sqrt{m_c} < \|\mathbf{I}(s) - \mathbf{I}(r_*(s))\|_2$ or $\sqrt{m_s} < \|s - r_*(s)\|_2$ then
6         $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{r_*(s)\}$
7     else
8         $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{s\}$
```

2.3.6 ISF Algorithm

In the original ISF work [78], Vargas-Muñoz et al. presented five different ISF methods, defined through various combinations of its components. The first two methods, namely ISF-GRID-ROOT and ISF-MIX-ROOT, are based on grid and mixed sampling, respectively, and use both the f_1 connectivity function and the r_2 recomputation function based on the mean feature vector. On the other hand, the third and fourth methods, namely ISF-GRID-MEAN and ISF-MIX-MEAN, are also based on grid and mixed sampling, respectively, and use both the f_2 connectivity function and the r_1 recomputation function based on the geometric center of the superpixel. A fifth ISF method, called ISF-REGMIN, relies on grid sampling and the f_3 connectivity function. ISF-REGMIN uses a single iteration of the IFT with no seed recomputation, thus leading to a faster execution time. In ISF-REGMIN, the seeds are replaced by any pixel located at the closest regional minimum in the gradient magnitude image. The ISF-GRID-ROOT and ISF-MIX-MEAN methods achieved better performance than their counterparts on most datasets.

A single execution of the IFT attains a complexity of $O(|\mathcal{D}_\mathcal{I}| \log(|\mathcal{D}_\mathcal{I}|))$ (linearithmic time) if the priority queue $\mathcal{Q}$ is implemented as a binary heap data structure. Thus, given that the time for seed recomputation is linear, the complexity of the ISF using a binary heap is linearithmic as well, regardless of the number of superpixels. Algorithm 5 depicts the pseudocode of the ISF using GRID sampling.

Algorithm 5: Iterative Spanning Forest (**ISF**)

```
Input    : Image $I = (\mathcal{D}_\mathcal{I}, \mathbf{I})$, adjacency relation $\mathcal{A}$, gradient function $\nabla\mathbf{I}$,
           connectivity function $f_*$, recomputation function $r_*$, number of
           superpixels $k$, and maximum number of iterations $T$
Output   : Predecessor map $P$, root map $R$, cost map $C$, and label map $L$
Auxiliar : Seed set $\mathcal{S}'$

1 $iter \leftarrow 0$
2 $\mathcal{S}' \leftarrow$ GRID$(I, \nabla\mathbf{I}, k)$
3 while $iter < T$ do
4     $\mathcal{S} \leftarrow \mathcal{S}'$
5     $(P, R, C, L) \leftarrow$ IFT$(I, \mathcal{A}, f_*, \mathcal{S})$
6     $\mathcal{S}' \leftarrow$ SEEDRECOMP$(I, \mathcal{S}, r_*, R, P, C)$
7     $iter \leftarrow iter + 1$
```

2.4 Dynamic Trees

In [14], Bragantini *et al.* introduced the *Dynamic IFT* (DynIFT), an extension of the IFT framework for interactive image segmentation that aims to leverage object information for dynamic arc-weight estimation as the trees evolve from the seed set during the IFT algorithm. Thus, the optimum-path trees evolve dynamically from the seed set as arc-weights in the graph are estimated from increasing object knowledge, leading to a more effective object delineation. In [12], Belém *et al.* presented a method called Dynamic Iterative Spanning Forest (DISF) that leverages the DynIFT algorithm to achieve more effective superpixel delineation for smaller numbers of superpixels. The following two sections present these approaches.

2.4.1 Dynamic Trees for Interactive Image Segmentation

Let $\mathcal{S}$ be a seed set comprised by labeled pixels (*e.g.*, labeled scribbles drawn by the user) in each object (including background). The DynIFT aims to partition the image into objects such that pixels enclosed within an object's boundary are more strongly connected to its internal seeds than to any other. An object labeling function λ_O uniquely identifies each seed $s \in \mathcal{S}$ with a belonging label among c objects such that $\lambda_O(s) \in \{0, \ldots, c\}$ with 0 as the background. Since different markers may comprise the same object, a marker labeling function $\lambda_M(s)$ is also implemented, so that it identifies the marker containing the seed s among m markers with $\lambda_M(s) \in \{0, \ldots, m\}$. The marker labeling function may also be used to control the addition and removal of markers during segmentation correction.

Therefore, in addition to the cost map C, predecessor map P, and root map R, the DynIFT algorithm propagates to every node $p \in \mathcal{D}_\mathcal{I}$ the object label map $L(p) = \lambda_O(R(p)) \in \{0, \ldots, c\}$, and the marker label map $M(p) = \lambda_M(R(p))$. The object label map represents the resulting segmentation after the algorithm's execution. The growing optimum-path trees $\mathcal{T}_s$, for each seed $s \in \mathcal{S}$ are referred as *dynamic trees* in the context of this algorithm.

Algorithm 6 depicts the pseudocode for the DynIFT. Lines 1–5 initialize all maps and insert all pixels $p \in D_I$ in a priority queue $\mathcal{Q}$. Next, in Lines 6–13, an optimum-path forest rooted in the seeds in $\mathcal{S}$ is computed. In Line 7, a pixel p with minimum path-cost value is removed from $\mathcal{Q}$ and inserted into the dynamic tree $\mathcal{T}_{R(p)}$ corresponding to the root of p. As the tree $\mathcal{T}_{R(p)}$ rooted in $R(p)$ is evolving, it increasingly contains information about the region within its boundary (including pixel p). The marker map M identifies all trees rooted at seeds with marker label $M(p) \in \{1, \ldots, m\}$ represented by $\bigcup_{\forall r \in \mathcal{S} | M(r)=M(p)} \mathcal{T}_r$, while map L identifies all trees rooted at seeds with object label $L(p) \in \{1, \ldots, c\}$ represented by $\bigcup_{\forall r \in \mathcal{S} | L(r)=L(p)} \mathcal{T}_r$. The dynamic estimation of arc-weights is carried out by leveraging color, texture, and shape about the object or its regions expressed through a dynamic arc-weight function.

```
Algorithm 6: Dynamic IFT (DynIFT) for $f_{\max}$ and $w_2$
  Input    : Image $I = (\mathcal{D}_\mathcal{I}, \mathbf{I})$, adjacency relation $\mathcal{A}$, connectivity function $f_{\max}$,
             arc-weight function $w_1$, and seed set $\mathcal{S}$ with labeling functions $\lambda_O$ and
             $\lambda_M$
  Output   : Object label map $L$
  Auxiliar: Priority queue $\mathcal{Q} = \emptyset$, dynamic sets $\mathcal{T}_r = \emptyset, \forall r \in \mathcal{S}$, maps $C$, $R$, $P$,
             and $M$, and variable $tmp$
1  foreach $p \in \mathcal{D}_\mathcal{I}$ do
2      $C(p) \leftarrow +\infty$, $R(p) \leftarrow p$, and $P(p) \leftarrow nil$
3      if $p \in \mathcal{S}$ then
4          $C(p) \leftarrow 0$, $L(p) \leftarrow \lambda_O(p)$, and $M(p) \leftarrow \lambda_M(p)$
5      Insert $p \in \mathcal{Q}$
6  while $\mathcal{Q} \neq \emptyset$ do
7      Remove $p$ from $\mathcal{Q}$, such that $p = \text{argmin}_{\forall q \in \mathcal{Q}}\{C(q)\}$ and $\mathcal{T}_{R(p)} \leftarrow \mathcal{T}_{R(p)} \cup \{p\}$
8      foreach $(p, q) \in \mathcal{A} \mid q \in \mathcal{Q}$ do
9          Estimate $w_1(p, q)$ as $w_1(p, q) = \min_{\forall r \in \mathcal{S} | L(r)=L(p)} \|\mu_r - \mathbf{I}(q)\|$
10         $tmp \leftarrow \max\{C(p), w_1(p, q))\}$
11         if $tmp < C(q)$ then
12             $C(q) \leftarrow tmp, R(q) \leftarrow R(p)$
13             $L(q) \leftarrow L(p), M(q) \leftarrow M(p)$, and $P(q) \leftarrow p$
```

Thus, the following dynamic arc-weight functions based on the tree's mean feature vector (color) $\mu_{R(p)}$ of the pixels $p \in \mathcal{T}_{R(p)}$ and the object's mean feature vector (color) $\mu_{L(p)}$ of the pixels $p \in \bigcup_{\forall r \in \mathcal{S} | L(r)=L(p)} \mathcal{T}_r$.

$$w_1(p,q) = \|\mu_{R(p)} - \mathbf{I}(q)\|, \tag{2.24}$$
$$w_2(p,q) = \min_{\forall r \in \mathcal{S} | L(r) = L(p)} \|\mu_r - \mathbf{I}(q)\|, \tag{2.25}$$
$$w_3(p,q) = \|\mu_{L(p)} - \mathbf{I}(q)\|, \tag{2.26}$$
$$w_4(p,q) = w_1(p,q) + \|\mathbf{I}(q) - \mathbf{I}(p)\|, \tag{2.27}$$
$$w_5(p,q) = w_2(p,q) + \|\mathbf{I}(q) - \mathbf{I}(p)\|, \tag{2.28}$$
$$w_6(p,q) = w_3(p,q) + \|\mathbf{I}(q) - \mathbf{I}(p)\|. \tag{2.29}$$

The arc-weight function w_1 considers the mean feature vector (color) of $\mathcal{T}_{R(p)}$ as more representative than the feature vector $\mathbf{I}(p)$ since is being constantly updated as the tree grows. Therefore, it substitutes $\mathbf{I}(p)$ in the standard arc-weight formulation. Thus, each seed will conquer pixels whose feature vector is similar to their respective regions' mean feature vector. In contrast, w_2 relaxes this criterion by considering the closest mean feature vector among all dynamic trees rooted at seeds with object label $L(p)$, therefore, favoring the delineation of objects comprised of disjoint regions. On the other hand, w_3 uses the mean feature vector of all pixels comprising all dynamic trees rooted at seeds with object label $L(p)$. The remaining functions add the local arc-weight $\|\mathbf{I}(q) - \mathbf{I}(p)\|$ to the definition of the previous functions to assess the importance of local contrast between regions.

The DynIFT was compared to the *power watershed* algorithm [25], the IFT algorithm with $f_{\max}$ and arc-weight function $w(p,q) = \|\mathbf{I}(q) - \mathbf{I}(p)\|$ [31], and the min-cut/max-flow algorithm [13, 70]. The DynIFT-based methods displayed an improvement in the accuracy of object delineation in comparison to their counterparts. Furthermore, the relaxed versions of w_1, represented by w_2 and w_5 achieved a better performance than the other arc-weight functions, whereas w_3 and w_6 obtained the worst performance among all other arc-weight functions.

2.4.2 Dynamic and Iterative Spanning Forest for Superpixel Segmentation

The Dynamic and Iterative Spanning Forest (DISF) algorithm for superpixel segmentation consists of the following steps. (*i*) a seed set is oversampled from an image graph, such that its size is significantly larger than the number of desired superpixels; (*ii*) the IFT algorithm with dynamic arc-weight estimation is executed to partition the image into connected superpixels; (*iii*) A procedure entailing the deletion of superpixels based on a relevance criterion is carried out. Next, steps (*ii*) and (*iii*) are repeated until the desired number of superpixels is attained.

Step (*i*) aims to increase the likelihood of selecting seeds that lead to superpixels that preserve boundary adherence, while the objective of step (*iii*) is to identify and keep these seeds in subsequent iterations in order to improve boundary delineation. In step (*ii*), superpixel delineation is carried out using the IFT with dynamic arc-weight estimation to leverage object information as the trees grow, aiming to take advantage of the object delineation properties of this formulation to achieve superpixels with better boundary adherence.

The DISF method uses the GRID sampling strategy for seed oversampling since it possesses the following properties: its definition does not take into account any prior object information, it increases the probability of setting a seed within the object of interest, and, last but not least, it generates a large number of seeds, which makes it suitable for this type of application. The GRID strategy selects equally-spaced seeds laid out in a grid pattern on the image as detailed in Section 2.3.2.

The superpixel segmentation is achieved by executing the IFT with dynamic arc-weight estimation using the arc-weight function $w_1(p, q) = \|\mu_{R(p)} - \mathbf{I}(q)\|_2$, previously defined in Equation 2.24, where $\mu_{R(p)}$ is the mean feature vector of the dynamic tree containing pixel p and rooted at $R(p)$, denoted $\mathcal{T}_{R(p)}$ – *i.e.*, $\mu_{R(p)} = \frac{1}{|\mathcal{T}_{R(p)}|}\sum_{\forall q \in \mathcal{T}_{R(p)}} \mathbf{I}(q)$.

The DISF method, in contrast to most recent seed-based superpixel approaches, starts with a number of seeds considerably larger than the number of desired superpixels aiming to include, in an unsupervised manner, seeds that produce superpixels that are effective in preserving boundary adherence. Therefore, the challenge relies on identifying and preserving these seeds for subsequent iterations. DISF introduces a relevance-based strategy to preserve and delete superpixels contingent on their attributed *relevance*. The relevance assigned to each seed is conditioned to the size of the superpixel it produces and the homogeneity of the region to which it belongs. Let $\mathcal{P}$ be the set of optimum-path trees generated by the execution of the IFT algorithm with dynamic arc-weight estimation. Then, a *tree-adjacency relation* $\mathcal{B}$ is defined as $\mathcal{B} = \{(\mathcal{T}_{R(p)}, \mathcal{T}_{R(q)}) \in \mathcal{P} \times \mathcal{P} \mid \exists (p, q) \in \mathcal{A} \text{ and } R(p) \neq R(q)\}$ and the relevance map $V : \mathcal{S} \rightarrow \mathbb{R}$ is defined as $V(s) = \frac{|\mathcal{T}_s|}{|\mathcal{D}_{\mathcal{I}}|} \min_{\forall(\mathcal{T}_s, \mathcal{T}_t) \in \mathcal{B}}\{\|\mu_{R(s)} - \mu_{R(t)}\|_2\}$.

Let N_0 be the number of seeds at iteration $i = 0$ and N_f the number of desired superpixels at the last iteration $i = T-1$. The number of highest priority seeds selected for iteration i is defined by $N(i) = \max\{N_0 \exp(-i), N_f\}$. Once the superpixel segmentation has been obtained, every seed $s \in \mathcal{S}$ from the current iteration i, $i \in \{0, \ldots, T-1\}$, is inserted into an empty priority queue $\mathcal{Q}$ with priority $V(s)$. Then, for the next iteration, $i + 1$, $\mathcal{S}$ is comprised only of the $N(i + 1)$ seeds of highest relevance in $\mathcal{Q}$. Moreover, the positions of the elements of $\mathcal{S}$ in $\mathcal{Q}$ are maintained for iteration $i + 1$ aiming to preserve segmentation consistency and boundary adherence in subsequent iterations.

Algorithm 7 presents a pseudocode for the DISF method. Line 1 initializes variable i, which will act as an iteration counter. Line 2 initializes the seed set $\mathcal{S}$ by seed oversampling using the GRID sampling strategy for $N_0 \gg N_f$ seeds. Lines 3–16 represent the main loop of the algorithm, which stops when the size of $\mathcal{S}$ is equal to N_f (number of desired superpixels) as per if statement in Lines 5–6. The superpixel segmentation occurs in Line 4 with the execution of the DynIFT algorithm where we adapt Algorithm 6 to only use a single labeling function λ for labeling superpixels and to return the tuple of maps (P, R, C, L). In Lines 7–10 a relevance value $V(s)$ is associated to each seed $s \in \mathcal{S}$ based on the size of $\mathcal{T}_s$ and the distance of the centroids of $\mathcal{T}_s$ and $\mathcal{T}_t$, $\forall(\mathcal{T}_s, \mathcal{T}_t) \in \mathcal{B}$, then seed s is inserted into the priority queue $\mathcal{Q}$ with priority $V(s)$. Afterwards, the number of seeds N for the next iteration is computed in Line 11 and, in Lines 12–15, the seed set $\mathcal{S}$ is established for the next iteration by selecting the N seeds with highest relevance. Lastly, the iteration counter i is updated in Line 16.

Algorithm 7: Dynamic Iterative Spanning Forest (**DISF**)

```
Input    : Image $I = (\mathcal{D}_\mathcal{I}, \mathbf{I})$, adjacency relation $\mathcal{A}$, gradient function $\nabla\mathbf{I}$,
           connectivity function $f_{max}$, arc-weight function $w_1$, labeling function $\lambda$,
           number of initial seeds $N_0$, and number of desired superpixels $N_f$
Output   : Predecessor map $P$, root map $R$, cost map $C$, and label map $L$
Auxiliar : Priority queue $\mathcal{Q}$, seed set $\mathcal{S}$, and variables $N$ and $i$

1  $i \leftarrow 0$
2  $\mathcal{S} \leftarrow$ GRID$(I, \nabla\mathbf{I}, N_0)$
3  while $|\mathcal{S}| \geq N_f$ do
4      $(P, R, C, L) \leftarrow$ DynIFT$(I, \mathcal{A}, f_{\max}, w_1, \mathcal{S}, \lambda)$
5      if $|\mathcal{S}| = N_f$ then
6          break
7      $\mathcal{Q} \leftarrow \emptyset$
8      foreach $s \in \mathcal{S}$ do
9          Estimate $V(s)$ as $V(s) = \frac{|\mathcal{T}_s|}{|\mathcal{D}_\mathcal{I}|} \min_{\forall(\mathcal{T}_s,\mathcal{T}_t)\in\mathcal{B}}\{\|\mu_{R(s)} - \mu_{R(t)}\|_2\}$
10         Insert $s$ in $\mathcal{Q}$ with priority $V(s)$
11     $N \leftarrow \max\{N_0 \exp(-(i+1)), N_f\}$
12     $\mathcal{S} \leftarrow \emptyset$
13     while $|\mathcal{S}| < N$ do
14         Remove $s$ from $\mathcal{Q}$, such that $s = \text{argmax}_{\forall t\in\mathcal{Q}}\{V(t)\}$
15         $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
16     $i \leftarrow i + 1$
```

The DISF mehod was compared with other five other state-of-the-art superpixel segmentation algorithms: (*i*) SLIC [1]; (*ii*) SH [81]; (*iii*) LSC [49]; (*iv*) ISF-GRID-ROOT; and (*v*) ISF-MIX-MEAN in the task of object delineation on three image datasets: (*i*) *Birds* [54]; (*ii*) *Liver* [78]; and (*iii*) *BSDS500* [10]. For evaluation purposes, two popular metrics *Boundary Recall* (BR) [1] and *Under-Segmentation Error* (UE) [58] were considered to assess object boundary preservation and to measure to what extend superpixels overlap object boundaries. The experiments were carried out in an interval from 20 to 1000 superpixels and DISF was executed with $N_0 = 8000$ for all datasets. DISF has shown outstanding results, outperforming its counterparts for BR and was among the best methods for UE.

2.5 Optimum-Path Forest

The *Optimum-Path Forest* (OPF) [61] is a framework that can be tailored to implement graph-based pattern recognition techniques to address a variety of problems [15, 62, 52, 68]. Essentially, OPF extends the IFT framework [31] from the image domain to the feature space. The OPF-based techniques reduce a pattern recognition problem to the computation of an optimum-path forest in a graph derived from the samples in the training set. It can provide effective supervised, unsupervised, and semi-supervised solutions, which we further detail in the following sections.

2.5.1 Supervised Learning with OPF

In [63], Papa *et al.* employed the OPF framework to develop a supervised classification method where the training set samples are modeled as the nodes of a complete graph, whose arcs are weighted by the distances between the feature vectors of their extreme nodes. The induced graph is then partitioned into an optimum-path forest comprised of optimum-path trees, each of them representing a single class, rooted at their most representative elements, known as *prototypes*. The prototypes are chosen from the minimum spanning tree of the induced graph by selecting samples belonging to the boundary of classes, in other words, samples of distinct classes that share an arc in this representation. Next, the prototypes carry out a competition procedure so that each sample in the training set ends up connected to its most strongly connected prototype through a minimum-cost path. The OPF-based classifiers present the following advantages over other state-of-the-art supervised classifiers (*e.g.*, *support vector machines* (SVMs) and *artificial neural networks* (ANNs)): (*i*) it is free of parameters; (*ii*) it does not make any prior assumption about the shape/separability of the feature space; and (*iii*) it has a fast training time allowing the development of real-time applications [66].

Let $\mathcal{Z}_1$, $\mathcal{Z}_2$, and $\mathcal{Z}_3$ be the training, evaluation and test sets, respectively, with $|\mathcal{Z}_1|$, $|\mathcal{Z}_2|$ and $|\mathcal{Z}_3|$ samples each. $\mathcal{Z}_1$ is used to create the classifier, while $\mathcal{Z}_3$ is used to determine the prediction accuracy. $\mathcal{Z}_2$ is used to further improve the accuracy of the classifier by randomly exchanging samples in $\mathcal{Z}_1$ with misclassified samples from $\mathcal{Z}_2$. Let $\lambda(s)$ be a labeling function that assigns the correct class label $i \in \{1, \ldots, c\}$ to each sample $s \in \mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$, and v be a feature extractor function that extracts n features from any sample $s \in \mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$, based on sample measurements, returning a vector $v(s) \in \mathbb{R}^n$. A set $\mathcal{S} \subset |\mathcal{Z}_1|$ represents the set of prototypes. The distance $d(s,t) \geq 0$ between the feature vectors of two samples, s and t, is given by any distance function suitable to the problem at hand (*e.g.*, the Euclidean distance $\|v(t) - v(s)\|$). Thus, the pair (v, d), also known as *descriptor*, determines how the feature vectors of the samples are distributed in the feature space.

The problem here entails the creation of a classifier that successfully predicts the ground-truth label given by $\lambda(s)$ for any sample $s \in \mathcal{Z}_3$. The training phase consists of the definition of a discrete optimal partition of $\mathcal{Z}_1$ in the feature space represented by an optimum-path forest rooted at a set of prototypes $\mathcal{S} \in \mathcal{Z}_1$. The classification phase for any sample $s \in \mathcal{Z}_3$ is carried out by evaluating the optimum paths, incrementally, with terminus s and assigning to it the label of its most strongly connected prototype.

Let $G = (\mathcal{Z}_1, \mathcal{A})$ be a complete graph whose nodes are represented by the training samples in $\mathcal{Z}_1$ and $\mathcal{A} \in \mathcal{Z}_1 \times \mathcal{Z}_1$ be the adjacency relation for any pair of distinct samples $s, t \in \mathcal{Z}_1$. A connectivity function f stipulates a path-cost value $f(\pi_t)$ to any sequence of distinct samples (*e.g.*, path) $\pi_t = \langle s_1, s_2, \ldots, t\rangle$ in G. The definitions for a path, detailed in Section 2.1, are also valid in this context by replacing pixels with samples. In [63], the connectivity function $f_{\max}$ (see Equation 2.6) is used, such that the minimization of f_{max} assigns to every sample $t \in \mathcal{Z}_1$ an optimum path rooted at a prototype $s \in \mathcal{S} \subset \mathcal{Z}_1$ leading to the partition of the graph into an optimum-path forest. The aforementioned procedure leads to the propagation of a predecessor map P, cost map C, and label map

L for all samples in $\mathcal{Z}_1$.

Algorithm 8 depicts the pseudocode of an OPF-based classifier executed on a complete graph and using the $f_{\max}$ connectivity function. Lines 1–5 initialize the cost, predecessor, and label maps and insert the prototypes in the priority queue $\mathcal{Q}$. Next, the while loop in Lines 6–14 computes an optimum-path forest rooted at $\mathcal{S}$ with terminus at every sample $s \in \mathcal{Z}_1 \setminus \mathcal{S}$ in a nondecreasing order of minimum cost. At each iteration in Line 6, we remove the nodes with minimum cost from $\mathcal{Q}$, such that a path of minimum cost $C(s)$ with terminus s is obtained. Then, for each sample $t \in \mathcal{Z}_1$, where $s \neq t$ and t has not been previously removed from $\mathcal{Q}$, we evaluate if the path with terminus s extended by the arc (s, t) (*e.g.*, $\pi_s \cdot \langle s, t\rangle$) offers a lower path-cost value than the current path with terminus t and update the position of t in $\mathcal{Q}$, as well as the map values $C(t)$, $L(t)$, and $P(t)$.

Algorithm 8: OPF-based classifier on a complete graph

```
Input    : Graph $G = (\mathcal{Z}_1, \mathcal{A})$, prototype set $\mathcal{S} \subset \mathcal{Z}_1$, labeling function $\lambda$,
           connectivity function $f_{\max}$, and descriptor $(v, d)$ for feature vector and
           distance computations
Output   : Predecessor map $P$, cost map $C$, and label map $L$
Auxiliar: Priority queue $\mathcal{Q}$ and variable $tmp$

1  foreach $s \in \mathcal{Z}_1$ do
2      $C(s) \leftarrow +\infty$
3      if $s \in \mathcal{S}$ then
4          $C(s) \leftarrow 0$, $P(s) \leftarrow nil$, and $L(s) \leftarrow \lambda(s)$
5          Insert $s$ in $\mathcal{Q}$
6  while $\mathcal{Q} \neq \emptyset$ do
7      Remove $s$ from $\mathcal{Q}$, such that $s = \text{argmin}_{\forall t \in \mathcal{Q}}\{C(t)\}$
8      foreach $t \in Z_1 \mid t \neq s$ and $C(t) > C(s)$ do
9          $tmp \leftarrow \max\{C(s), d(s, t)\}$
10         if $tmp < C(t)$ then
11             if $C(t) \neq +\infty$ then
12                 Remove $t$ from $\mathcal{Q}$
13             $C(t) \leftarrow tmp$, $P(t) \leftarrow s$, and $L(t) \leftarrow L(s)$
14             Insert $t$ in $\mathcal{Q}$
```

Once the classifier has been projected, the classification phase occurs in such a way that, for any sample $t \in \mathcal{Z}_3$, we extend all possible paths with terminus $s \in \mathcal{Z}_1$ with the arc (s, t) (*e.g.*, $\pi_s \cdot \langle s, t\rangle$) to determine the optimum path π^*_t rooted at $\mathcal{S}$ that offers the lowest path-cost value, and label t with the same class as its most strongly connected prototype. The path π^*_t can be identified incrementally by $C(t) = \min_{\forall s \in \mathcal{Z}_1}\{\max\{C(s), d(s, t)\}\}$.

The evaluation phase may occur in applications with large datasets, where identifying the most informative samples to project a more effective classifier becomes a daunting task. Therefore, the use of a third evaluation set $\mathcal{Z}_2$ aims to improve the informativeness of samples comprising the training set $\mathcal{Z}_1$ without increasing its size. The projected classifier is evaluated on $\mathcal{Z}_2$, and the misclassified samples are exchanged with randomly selected samples in $\mathcal{Z}_1$. These misclassified samples are considered to contain valuable information and are included in $\mathcal{Z}_1$ to retrain the classifier, aiming to improve its accuracy. The updated sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are used to repeat this procedure for a fixed number of iterations, and the classifier with the highest accuracy among all iterations is selected.

The flexibility of the OPF framework allows the creation of classifiers using different graph topologies, different connectivity functions, and other strategies to estimate prototypes and improve the classifier effectiveness from the evaluation set. In this context, Papa and Falcão [60] propose an OPF-based classifier where the arcs of the graph are established by the k-nearest neighbors of each node in the feature space. The nodes are weighted by their probability density value (pdf), and prototypes are selected from the maxima of the pdf.

2.5.2 Unsupervised Learning with OPF

Rocha *et al.* [68] proposed an unsupervised variation of the optimum-path forest framework for data clustering. This work explores the assumption that natural groups can be viewed as regions with a high density of samples represented by the plateaus of their probability density function (pdf) and exploits optimum connectivity between samples in the feature space. These plateaus or maxima are then determined, and the estimation of its influence zones through the computation of an optimum-path forest defines a group as an optimum-path tree. However, this approach may not lead to the desired number of groups (*i.e.*, optimum-path trees), so a process of cluster reduction is carried to remove "irrelevant" clusters.

Let $G = (\mathcal{Z}, \mathcal{A})$ be a graph whose nodes are represented by the samples in $\mathcal{Z}$, and $\mathcal{A}$ defines the adjacency relation such that $t \in \mathcal{A}(s)$ if t is a k-nearest neighbor of s in the feature space where $k > 0$ is an integer parameter. Let v be a feature extractor function, and $d(s,t)$ (*e.g.*, $d(s,t) = \|v(t) - v(s)\|_2$) be the distance function that determines the distance between samples s and t in the feature space. The graph nodes are weighted by their probability density values $\rho(s)$ estimated via the Parzen-window technique with a Gaussian kernel (see Equation 2.30). The value of σ is set to guarantee the inclusion of the most adjacent samples in the pdf estimation.

$$\begin{aligned} \rho(s) &= \frac{1}{\sqrt{2\pi\sigma^2}|\mathcal{A}(s)|} \sum_{t \in \mathcal{A}(s)} \exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right) \\ \sigma &= \max_{\forall (s,t) \in \mathcal{A}} \left\{\frac{d(s,t)}{3}\right\} \end{aligned} \tag{2.30}$$

The selection of k to define the adjacency relation $\mathcal{A}$ is strongly related to the final graph partition. Therefore, different choices of k may lead to distinct graph partitions (optimum-path forest). Accordingly, the value of k is selected as the one that leads to an optimum-path forest that minimizes a graph-cut measure. The computation of the optimum-path forest leads to the estimation of the influence zones of the pdf's maxima. However, the maximum of the pdf may present adjacent samples with the same density value. Therefore, it is required to guarantee connectivity between any pair of samples in such a region. Thus, the adjacency relation $\mathcal{A}$ is extended to be symmetric in the plateaus of ρ as follows: if $t \in \mathcal{A}(s), s \notin \mathcal{A}(t)$ and $\rho(s) = \rho(t)$, then $\mathcal{A}(t) \leftarrow \mathcal{A}(t) \cup \{s\}$. This approach considers the weights of the nodes instead of the arc-weights towards determining the strength of connectedness for a given path between two samples. Hence, every sample $t \in \mathcal{Z}$ will be assigned to an optimum path π_t^* rooted at its most strongly connected maximum among the pdf's maxima for a smooth connectivity function $f(\pi_t)$ (*e.g.*, $C(t) = \max_{\forall \pi_t \in (\mathcal{Z},\mathcal{A})}\{f(\pi_t)\}$). Since we have no prior knowledge about the maxima of the pdf, a handicap value $f(\langle t \rangle) = h(t) < \rho(t)$, for all $t \in \mathcal{Z}$, is defined to preserve the relevant maxima of the pdf and discard the domes that can be reached by optimum paths rooted at such maxima. Therefore, the connectivity function f, also known as $f_{\min}$, is defined in Equation 2.31.

$$
\begin{aligned}
f_{\min}(\langle t \rangle) &= \begin{cases} \rho(t) & \text{if } t \text{ is a relevant maximum} \\ h(t) & \text{otherwise} \end{cases} \\
f_{\min}(\pi_s \cdot \langle s, t \rangle) &= \min\{f(\pi_s), \rho(t)\},
\end{aligned}
\tag{2.31}
$$

Algorithm 9 presents the pseudocode of OPF-based clustering with connectivity function $f_{\min}$. The relevant maxima are identified and labeled with a consecutive integer l. Then, optimum paths rooted at the maxima are computed for $f_{\min}$ following a nonincreasing order of path-cost values. The maps $C(t)$, $L(t)$, and $P(t)$ store the optimum path-cost, root label, and predecessor for all $t \in \mathcal{Z}$, respectively.

Algorithm 9: Clustering by optimum-path forest for $f_{\min}$

```
Input    : Graph $G = (\mathcal{Z}, \mathcal{A})$ and functions $h$ and $\rho$, $h(t) < \rho(t)$ for all $t \in \mathcal{Z}$
Output   : Label map $L$
Auxiliar : Predecessor map $P$, cost map $C$, priority queue $\mathcal{Q}$, and variables $tmp$
           and $l$

1  $l \leftarrow 1$
2  foreach $s \in \mathcal{Z}$ do
3      $P(s) \leftarrow nil$ and $C(t) \leftarrow h(t)$
4      Insert $s$ in $\mathcal{Q}$
5  while $\mathcal{Q} \neq \emptyset$ do
6      Remove $s$ from $\mathcal{Q}$, such that $s = \text{argmax}_{\forall t \in \mathcal{Q}}\{C(t)\}$
7      if $P(s) = nil$ then
8          $L(s) \leftarrow l$, $l \leftarrow l + 1$, and $C(s) \leftarrow \rho(s)$
9      foreach $t \in \mathcal{A}(s) \mid C(t) < C(s)$ do
10         $tmp \leftarrow \min\{C(s), \rho(t)\}$
11         if $tmp > C(t)$ then
12             $L(t) \leftarrow L(s)$, $P(t) \leftarrow s$, and $C(t) \leftarrow tmp$
13             Update position of $t$ in $\mathcal{Q}$
```

Lines 1–4 initialize variable l, and predecessor and cost maps for each sample $t \in \mathcal{Z}$ followed by the insertion of t into priority queue $\mathcal{Q}$. At a single iteration of the main loop in Lines 5–13, we remove sample s with maximum path-cost $C(s)$ from priority queue $\mathcal{Q}$ such that an optimum path π_s^* is obtained. Next, the test $P(s) = nil$ identifies whether the sample s is a maximum. If the equality holds, s becomes a root of the forest, its label $L(s)$ is set to l, and its path-cost value $C(s)$ is updated to $\rho(s)$, so it can conquer the rest of the samples in such maximum. Lines 9–13 evaluate whether the path with terminus s extended by the arc (s, t) offers a higher path-cost value than the current path with terminus t and update the map values $L(t)$, $P(t)$, and $C(t)$ as well as its position in $\mathcal{Q}$, accordingly.

The final result of the clustering relies on the choice of $\mathcal{A}$ and, therefore, on the value of k, which defines the k-nearest neighbor graph. A graph-cut measure $C(k)$ is employed to determine the most suitable k-nearest neighbor graph. Equation 2.32 defines $C(k)$ where c is the number of generated groups and $k \in [1, |\mathcal{Z}| - 1]$.

$$\begin{aligned} C(k) &= \sum_{i=1}^{c} \frac{W_i'}{W_i + W_i'} \\ W_i &= \sum_{(s,t)\in\mathcal{A}|L(s)=L(t)=i} \frac{1}{d(s,t)} \\ W_i' &= \sum_{(s,t)\in\mathcal{A}|L(s)=i,L(t)\neq i} \frac{1}{d(s,t)} \end{aligned} \tag{2.32}$$

Algorithm 9 is executed for different values of k within a range $[1, k_{\max}]$, for $k_{\max} \ll |\mathcal{Z}|$. Then, k is selected as the value that minimizes the graph-cut measure – *i.e.*, $k = \min_{k\in[1,k_{\max}]} C(k)$. This OPF-based clustering solution has been successfully applied to interactive image segmentation [68], brain tissue MR-image segmentation [15] and active learning [72].

2.5.3 Semi-supervised Learning with OPF

Semi-supervised learning approaches address the problem of training a classifier when only a scarce amount of labeled data is available. *Amorim et al.* [8] introduced a semi-supervised extension of the OPF framework executed on a complete graph and using the maximum arc-weight path-value function (*i.e.*, $f_{\max}$) to project a classifier in cases when labeled data is limited.

Let $\mathcal{Z}_1$, $\mathcal{Z}_2$, and $\mathcal{Z}_3$ be the training, evaluation, and test sets. Let $\mathcal{Z}_1$ be comprised of a set of labeled samples $\mathcal{Z}_1'$ and a set of unlabeled samples $\mathcal{Z}_1''$, such that $\mathcal{Z}_1 = \mathcal{Z}_1' \cup \mathcal{Z}_1''$. A labeling function λ assigns the correct labels to every labeled sample. The set $\mathcal{Z}_1$ is used to project the semi-supervised classifier, while the set $\mathcal{Z}_3$ assesses the classifier's accuracy. The evaluation set $\mathcal{Z}_2$ is assumed to be larger than $\mathcal{Z}_1'$ and is used to improve the classifier's accuracy further and to reduce the propagation error in $\mathcal{Z}_1''$ by randomly replacing samples in $\mathcal{Z}_1'$ with misclassified samples in $\mathcal{Z}_2$. The set of prototypes $\mathcal{S} \subset \mathcal{Z}_1'$ propagates their labels to the samples in $\mathcal{Z}_1''$ during the training process.

Let $G = (\mathcal{Z}_1, \mathcal{A})$ be a complete graph whose nodes are represented by the training samples in $\mathcal{Z}_1 = \mathcal{Z}_1' \cup \mathcal{Z}_1''$ and the adjacency relation $\mathcal{A}$ is defined by each pair of samples $s, t \in \mathcal{Z}_1$, such that $s \neq t$. The training phase starts by defining a set of prototypes $\mathcal{S}$ from the most representative samples of each class. The set $\mathcal{S}$ can be determined from several heuristics, among which is the computation of a minimum-spanning tree on the graph $(\mathcal{Z}'_1, \mathcal{A})$ followed by the selection of samples of distinct classes that share an arc on the minimum-spanning tree representation. Next, the OPF algorithm with connectivity function $f_{\max}$ is executed on G aiming to minimize the path-cost map $C(t) = \min_{\forall \pi_t \in \Pi_t}\{f_{\max}(\pi_t)\}$ for all $t \in \mathcal{Z}_1$. Afterward, an optimum-path forest (*i.e.*, semi-supervised classifier) rooted at $\mathcal{S}$ is generated, such that all samples in $\mathcal{Z}_1$ end up labeled. According to the original work, it is recommended to retrain the OPF-based semi-supervised classifier with the labeled samples in $\mathcal{Z}_1$ in order to improve the prototype set $\mathcal{S}$. The classification phase for a new sample $t \in \mathcal{Z}_3$ (or $\mathcal{Z}_2$) using the projected semi-supervised classifier considers that t is connected to all nodes in training set $\mathcal{Z}_1$. Next, the path with minimum cost π^*_t among all possible paths Π_t (*i.e.*, paths rooted at $\mathcal{S}$ and terminus t) in the training graph is identified, and afterward, t adopts the class label corresponding to the root of path π^*_t.

This semi-supervised approach was compared against the traditional Optimum-Path Forest (OPF) and Transductive Support Vector Machines (TSVM) [22] on four datasets from various domains. The evaluation measures were the mean accuracy, the error rate of label propagation on the training set, and the Friedman's statistical test. The OPF-based semi-supervised approach showed the best performance on all four datasets.

2.6 Concluding Remarks

This chapter reviews the related work

Clustering algorithms Partitional Clustering Algorithms Hierarchical Clustering Algorithms Graph-based Clustering algorithms

K-means

Iterated Watersheds

Clustering by OPF

essential theoretical concepts that set the ground to introduce the Iterative Optimum-Path Forest (IOPF) framework. The Image Foresting Transform (IFT) framework is introduced as a key concept towards presenting the Iterative Spanning Forest (ISF) framework, which is used to design effective superpixel segmentation methods. The ISF framework is described thoroughly, detailing each of its constituent components. Next, the concept of dynamic arc-weight estimation is presented along with applications in interactive image segmentation and superpixel segmentation. Lastly, the Optimum-Path Forest (IOPF) framework, a generalization of the IFT framework from the image domain to the feature space, is introduced with applications to supervised, unsupervised and semi-supervised learning. The next chapter presents the related work and provides basic definitions regarding this master's work.

Chapter 3

Related Work

This chapter reviews the related work concerning clustering algorithms and their categorization based on their solving strategy. Furthermore, we describe several OPF-based clustering algorithms that can be found in the literature. The chapter starts with a brief definition of data clustering and a broad categorization of clustering algorithms in Section 3.1. Next, k-means, a widely popular clustering algorithm, is presented in Section 3.2. Section 3.3 describes the Iterated Watersheds (IW) algorithm, a variation of k-means that exploits optimum connectivity through a sequence of OPF executions. IW can be regarded as a particular implementation of the IOPF framework introduced in this work. Lastly, state-of-the-art OPF-based clustering algorithms are briefly reviewed in Section 3.4.

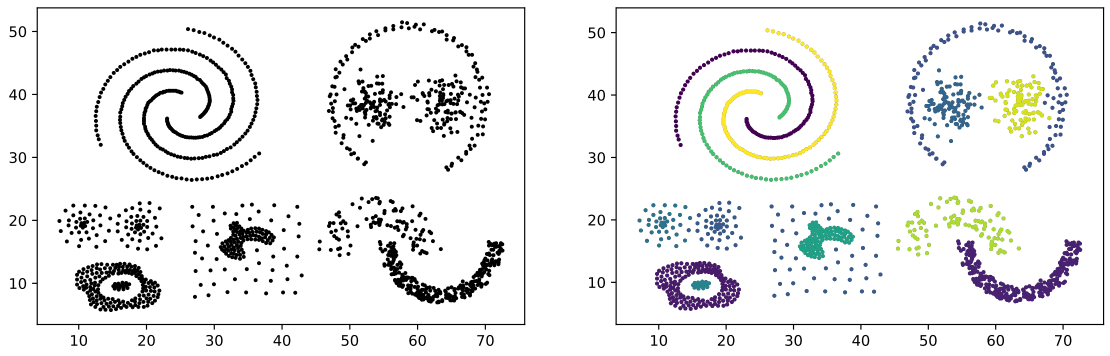
3.1 Clustering algorithms

The goal of data clustering, also known as clustering analysis, is to identify the *natural* grouping of a set of objects aiming to get insights into its underlying structure to perform further analysis and extract knowledge intrinsic to the data.

Clustering can be defined as the process of partitioning a set of n objects, represented by their feature vectors, into k groups, such that objects belonging to the same group displaying a high degree of similarity according to a given similarity measure. Clusters can be found in a variety of shapes, sizes, and densities (see Figure 3.1). Ideally, a cluster should be compact and isolated. However, real-life data usually contains noise, which thwarts the task of cluster identification and separation [43].

A cluster may also be thought of as an abstract entity whose significance and interpretation depend on the observer's domain knowledge. While humans are excellent cluster detectors in two and possibly three dimensions due to their visual capabilities, most problems are better explained in higher dimensions. Therefore, the creation of automatic clustering solutions to address high-dimensional data becomes a necessary endeavor. This challenge led to the proposal of several clustering that tackles this problem using a wide variety of strategies and techniques. In this context, another interesting problem is identifying the number of "natural" clusters for a given set of objects.

Clustering analysis is predominant in disciplines related to the study and analysis of



(a) Input data (b) Desired clustering

Figure 3.1: Diversity of clusters. The clusters in (a) (indicated by distinct colors in (b)) differ in shape, size, and density. Adapted from [43]

multivariate data, such as sequence analysis, image segmentation, and others. There exists an extensive literature on a number of scientific fields and applications encompassing clustering techniques for solving many related problems. Similarly, existing algorithms may also be tailored to address a particular problem, resulting in a variant of the original algorithm. Some problems that might not seem related to clustering analysis can be formulated as a clustering problem, for instance, image segmentation, a critical subarea of computer vision, can be defined as a clustering problem [21]. Clustering analysis can also be used in *document categorization* to generate hierarchies of topics for efficient information access and retrieval; *customer segmentation* to group customers into different categories for efficient marketing; *clustering of genome data* to discover potential relationships among genomic clusters; also *clustering of search engine results* to present the vast number of available web pages in an organized fashion [43].

The main usages of data clustering, based on their applications, can be categorized within three primary purposes according to *Jain* in [43]: *discovering* and *exploring* the underlying structure of the data, *natural classification* – to determine the degree of similarity among the elements of a dataset, and *compression* – to summarize the data by choosing representative elements from the clusters.

Clustering methods can be broadly categorized into different paradigms according to the approach followed to partition the data. Among these paradigms, we find *partitional clustering*, *hierarchical clustering* and *graph-based clustering* which are further described in the following sections. Let $\mathcal{Z}$ be a dataset where $|\mathcal{Z}| = n$ and let v be a function such that it assigns a feature vector $v(s) \in \mathbb{R}^m$ to every sample $s \in \mathcal{Z}$.

3.1.1 Partitional Clustering Algorithms

Partitional clustering attempts to discover a k-partition of $\mathcal{Z}$ given by $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ with $1 \leq k \leq n$, such that

1. $\mathcal{C}_i \neq \emptyset, \forall i \in \{1, \ldots, k\}$
2. $\bigcup_{i=1}^{k} \mathcal{C}_i = \mathcal{Z}$
3. $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset, \forall i, j \in \{1, \ldots, k\}$ and $i \neq j$

Therefore, $\mathcal{Z}$ is partitioned into a prespecified number of non-overlapping groups k with no hierarchical structure, such that each sample $s \in \mathcal{Z}$ is assigned to only a single cluster or subset. In principle, the optimal partition, that is, the partition maximizing both intra-clustering similarity and inter-clustering dissimilarity given by the minimization of an expected loss, can be found by exhaustive enumeration. However, such a brute-force procedure becomes computationally intractable and, therefore, unfeasible for practical purposes. Thus, heuristic solutions have been developed to address this problem and seek for an approximate solution.

A critical component in partitional clustering is the criterion function [40]. The sum-of-squared error (SSE) stand out as one of the most widely used criteria. Let $s_j \in \mathcal{Z}, j \in \{1, \dots, n\}$ be the samples of $\mathcal{Z}$ and let $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_k\}$ the k subsets, in which $\mathcal{Z}$ is partitioned. We define $\Gamma = \{\gamma_{ij}\}$ as a partition matrix such that

$$\gamma_{ij} = \begin{cases} 1 & \text{if } s_j \in C_i \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

with $\sum_{i=1}^{k} \gamma_{ij} = 1, \forall j \in \{1, \dots, n\}$, that is, each sample s_j belongs to only one cluster. Moreover, we define $\mathcal{M} = \{m_i\}, i \in \{1, \dots, k\}$ as a cluster prototype or centroid matrix where $m_i = \frac{1}{n_i} \sum_{j=1}^{n} \gamma_{ij} s_j$ and n_i is the number of elements of cluster $\mathcal{C}_i$ (*i.e.*, $|\mathcal{C}_i| = n_i$). Based upon the previous definitions, we define the SSE in Equation 3.2.

$$\mathcal{J}(\Gamma, \mathcal{M}) = \sum_{i=1}^{k} \sum_{j=1}^{n} \gamma_{ij} \|s_j - m_i\| \tag{3.2}$$

Therefore, the objective of partitional clustering methods is to minimize such function following a certain heuristic. The k-means algorithm is by far the best-known and most widely used squared error-based clustering algorithm [43, 53].

3.1.2 Hierarchical Clustering Algorithms

Hierarchical clustering methods construct a tree-like nested structure partition of $\mathcal{Z}$ represented by an ordered set of hierarchy levels $\mathcal{H} = \{\mathcal{H}_1, \dots, \mathcal{H}_q\}, q \leq n$, where each element $\mathcal{H}_m \in H$ represents a level of the hierarchy, such that if $\mathcal{C}_i \subset \mathcal{H}_m$, $\mathcal{C}_j \subset \mathcal{H}_l$, and $m > l$ imply that $\mathcal{C}_i \subset \mathcal{C}_j$ or $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for all $i, j \neq i$ and $m, l \in \{1, \dots, q\}$. Therefore, $\mathcal{H}_1$ and $\mathcal{H}_q$ constitute the highest and lowest level of the hierarchy, and $\mathcal{C}_i, \mathcal{C}_j$ represents a subset of a partition of $\mathcal{Z}$ in hierarchy levels $\mathcal{H}_m, \mathcal{H}_l$, respectively.

Hierarchical clustering performs the grouping of data samples following a sequence of partitions, either from singleton clusters to a single cluster containing all individuals (*e.g.*, bottom-up) or the other way around (*e.g.*, top-down). The results of hierarchical clustering methods are regularly depicted by a binary tree or dendrogram. The root node of the dendrogram is regarded as the entire data set, whereas each leaf node represents a single data sample. On the other hand, intermediate nodes describe groups of samples based on their proximity to each other. At the same time, while the height of the dendrogram expresses the distance between pairs of objects or clusters, as well as an object and a cluster. A given partition of $\mathcal{Z}$ is obtained by cutting the dendrogram at different levels.

The dendrogram representation provides a visual tool that enables users to describe and unveil potential hierarchical relations in the data.

Hierarchical clustering algorithms are broadly categorized into two categories: *agglomerative* and *divisive* methods. Agglomerative clustering starts with n singleton clusters. Next, a sequence of merge operations is carried out, creating a hierarchical structure, ending up with a single cluster containing all samples in $\mathcal{Z}$. In contrast, divisive clustering follows an inverse procedure, starting with a single cluster containing all samples in the dataset. Then, a divisive procedure is applied successively on the initial cluster, ending up with a set of n singleton clusters. However, this approach is not commonly used since for a cluster of n samples, there are $2^{n-1}-1$ possible pairwise combinations, which turns it computationally prohibitive. The *MONothetic Analysis* (MONA) and *DIvisive ANAlysis* (DIANA), two popular divisive clustering algorithms, are further described in [47].

Let $\mathcal{M} = \{m_{ij}\}$ be a $n \times n$ symmetric matrix, called proximity matrix, such that the element m_{ij}, for $i, j \in \{1, \ldots, n\}$, represents the similarity or dissimilarity measure between the ith and jth elements according to a similarity metric. Thus, the distance between clusters $\mathcal{C}_i, \mathcal{C}_j$ can be denoted as $\delta(\mathcal{C}_i, \mathcal{C}_j) = m_{ij}$ according to the proximity matrix $\mathcal{M}$.

Algorithm 10: General algorithm for agglomerative clustering

```
Input    : Dataset $\mathcal{Z}$
Output   : Hierarchical clustering structure $\mathcal{C}$
Auxiliar: Clusters $\mathcal{C}_i$ and $\mathcal{C}_j$
1  $\mathcal{C} \leftarrow \emptyset$
2  foreach $s \in \mathcal{Z}$ do
3      $\mathcal{C} \leftarrow \{\{s\}\}$
4  Compute the proximity matrix $\mathcal{M}$ of the set $\mathcal{C}$ with $|\mathcal{C}| = n$
5  while $|\mathcal{C}| > 1$ do
6      Search the pair $\mathcal{C}_i, \mathcal{C}_j$ such that $\delta(\mathcal{C}_i, \mathcal{C}_j) = \min_{\forall \mathcal{C}_k, \mathcal{C}_l \in \mathcal{C}, k \neq l} \delta(\mathcal{C}_k, \mathcal{C}_l)$
7      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\mathcal{C}_i \cup \mathcal{C}_j\}$
8      $\mathcal{C} \leftarrow \mathcal{C} \cup \{\{\mathcal{C}_i, \mathcal{C}_j\}\}$
9      Update the proximity matrix by computing the distances between the new
         cluster $\{\mathcal{C}_i, \mathcal{C}_j\}$ and the remaining clusters in $\mathcal{C}$.
10 return $\mathcal{C}$
```

Algorithm 10 depicts a general algorithm for agglomerative clustering. In Lines 1–3, the set $\mathcal{C}$ is initialized with singleton clusters for each sample $s \in \mathcal{Z}$, such that $|\mathcal{C}| = n$. During each iteration of the loop in Lines 5–9, the pair of clusters $\mathcal{C}_i, \mathcal{C}_j \in \mathcal{C}$ with minimal distance $\delta(\mathcal{C}_i, \mathcal{C}_j)$ are identified among all pairs of clusters with $i \neq j$. Next, both clusters $\mathcal{C}_i, \mathcal{C}_j$ are removed from $\mathcal{C}$, then a new cluster $\{\mathcal{C}_i, \mathcal{C}_j\}$ is created and subsequently included in $\mathcal{C}$. Then, the proximity matrix is updated by computing the distances between the recently created cluster and the remaining clusters in $\mathcal{C}$. Finally, the hierarchical structure $\mathcal{C}$ is returned.

Based on the selection of the distance function between two clusters, many agglomerative clustering algorithms can be cited. The two most popular agglomerative techniques are *single linkage* [75] and *complete linkage* [77]. The single linkage strategy determines inter-cluster distance by computing the distance between the two closest elements located in distinct clusters, whereas for the complete linkage strategy, it is determined by the farthest distance between a pair of elements, each located in different clusters. The selection of more complex distance functions led to the proposal of other agglomerative clustering algorithms, namely group average linkage, median linkage, centroid linkage, and Ward's method [57].

Usually, hierarchical clustering solutions present the following drawbacks: (*i*) they are sensitive to noise and outliers; (*ii*) they are not able to correct misclassifications; (*iii*) they display a high time complexity, being at least $O(n^2)$ for most algorithms, and therefore becoming computationally prohibitive for large datasets; and (*iv*) they assume the clusters to be spherical. Recently, several other techniques have been proposed aiming to overcome some of the aforementioned drawbacks with special emphasis on processing large-scale datasets, namely *CURE* [38], *ROCK* [39], *Chameleon* [46], and *BIRCH* [85].

3.1.3 Graph-based Clustering Algorithms

Graph-based clustering algorithms use the concepts and properties of graph theory such that the clustering problem is described by means of graphs. According to Aggarwal [5], graph clustering algorithms may be categorized into node clustering and graph clustering. However, only related works falling in the first category will be reviewed.

The nodes of a weighted graph G represent the samples of the dataset $\mathcal{Z}$ in the feature space, while arcs are built through spatial proximity, reflecting a potential relationship between each pair of samples. A threshold value δ_0 may be selected to define the edges of G in such a way that for any pair of samples $s, t \in \mathcal{Z}$ with $s \neq t$ an edge connecting such nodes is created if $\delta(s,t) < \delta_0$ for a given distance function δ (*e.g.*, Euclidean distance).

Graph-based methods can be applied in both hierarchical and partitional clustering. In hierarchical clustering, for instance, both single linkage and complete linkage clustering can be described as a graph problem equivalent to seeking maximally connected subgraphs (components) and maximally complete subgraphs (cliques) [44], respectively. *Chameleon* [46] is a agglomerative clustering algorithm based on the k-nearest neighbor (k-NN) graph. It starts by partitioning the k-NN graph into a set of subclusters with the minimal edge cut. Next, the characteristics of potential clusters are explored through the computation of the relative interconnectivity and relative closeness. Then, such small subsets are merged, ending up with the final graph partition (clustering).

On the other hand, in the context of partitional clustering, Zhan [84] proposed a graph-based clustering algorithm consisting of identifying and discarding inconsistent edges in the *minimum spanning tree* (MST) from an input weighted graph to partition it into compact subgraphs. Hartuv and Shamir [41] presented *Highly Connected Subgraphs* (HCS) where "highly connected" is equivalent to say that the connectivity (*i.e.*, the minimum number of edges to remove to disconnect the graph) of a subgraph is at least half as great as the number of vertices. HCS determines these subgraphs recursively using the minimum cut procedure. CLICK [74] is another algorithm that can be regarded as an adaption of the HCS algorithm on weighted similarity graphs. The edges of the graph are weighted using a probabilistic approach, and clusters are formed based on the computation of the minimum-weight cut.

Spectral clustering methods constitute another strategy, which may be divided into three steps: (*i*) create a weighted graph, with samples as nodes and arcs between adjacent samples; (*ii*) compute the first k eigenvectors of its Laplacian matrix to define a feature vector for each sample in the $\mathbb{R}^k$ space; and (*iii*) execute the k-means algorithm in the $\mathbb{R}^k$ space to identify and label the groups [19].

A more detailed list of graph-based clustering algorithms may be found in the works by Aggarwal [5] and Schaeffer [73].

3.2 K-means

The k-means algorithm is probably the best-known squared error-based partitional clustering solution [35, 43]. Even though k-means was originally proposed for the first time more than 50 years ago, it remains one of the most commonly used clustering algorithms. Its popularity stems from its ease of implementation, simplicity, efficiency, and empirical success [43].

Let $\mathcal{Z}$ be a set of n m-dimensional objects to be partitioned into a set of k disjoint clusters represented by $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. The pseudocode of the k-means algorithm is depicted in Algorithm 11. It starts by randomly selecting k cluster prototypes $s_i, i \in \{1, \ldots, k\}$ from $\mathcal{Z}$ in Line 1. Then, in Lines 2–3, the set of centroids represented by $\mu_i, i \in \{1, \ldots, k\}$ are initialized to the set of initial cluster prototypes. Next, an iterative procedure is carried out consisting in assigning each object $s \in \mathcal{Z}$ to the nearest cluster (Lines 7–9), subsequently followed by the recomputation of the centroids based on the current partition (Lines 10–11). The main loop in Lines 4–11 iterates until there is no change for each cluster. Lastly, in Line 12, the set of centroids computed during the last iteration is returned.

Algorithm 11: K-means algorithm

```
Input    : Dataset $\mathcal{Z}$ and number of clusters $k$
Output   : Set of centroids $\{\mu_1, \ldots, \mu_k\}$
Auxiliar : Set of clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

1  Select $k$ random seeds from $\mathcal{Z}$: $\{s_1, \ldots, s_k\}$
2  for $i = 1$ to $k$ do
3      $\mu_i \leftarrow s_k$
4  while convergence is not achieved do
5      for $i = 1$ to $k$ do
6          $\mathcal{C}_i \leftarrow \emptyset$
7      foreach $s \in \mathcal{Z}$ do
8          $j \leftarrow \text{argmin}_{1 \leq j' \leq k} \|\mu_{j'} - s\|$
9          $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \{s\}$
10     for $i = 1$ to $k$ do
11         $\mu_i \leftarrow \frac{1}{|\mathcal{C}_i|} \sum_{s \in \mathcal{C}_i} s$
12 return $\{\mu_1, \ldots, \mu_k\}$
```

The k-means algorithm works very well for compact and hyperspherical clusters. It has a time complexity of $O(nkm)$ where the values of k and m are usually much less than n. Therefore, it can be used to partition large datasets effectively. However, there is no universal method for selecting the initial cluster prototypes. Therefore, a general strategy is to run k-means several times and choose the best partition according to an evaluation metric. On the other hand, k-means presents the following drawbacks: (*i*) the number of clusters k must be given in advance; (*ii*) the final partition is heavily dependent on the choice of initial cluster prototypes; (*iii*) The iteratively optimal procedure of k-means cannot guarantee convergence to a global optimum. Thus, k-means computes local optimal partitions; and (*iv*) k-means is sensitive to outliers and noise [71]. Many variants and advances of k-means have been proposed aiming to overcome these issues and to address a wide range of applications. Some of these works can be found in [6].

3.3 Iterated Watersheds

Recently, Soor *et al.* [76] proposed a graph-based variant of k-means, called *Iterated Watersheds* (IW), that exploits optimum connectivity between a set of prototypes and the remaining samples in the dataset. This graph-based clustering solution consists of a sequence of applications of the watershed transform through various executions of the OPF with restricted graph topology from enhanced sets of prototypes. The IW algorithm comprehends three fundamental steps: (*i*) sampling of the initial set of prototypes (seeds) $\mathcal{S}$; (*ii*) graph partitioning by Optimum-Path Forest (OPF); and (*iii*) recomputation of $\mathcal{S}$ based on the graph partitioning of (*ii*). Next, steps (*ii*) and (*iii*) are repeated until a maximum number of iterations is attained or until there is no change for each graph partition (cluster). The steps (*ii*) and (*iii*) can be regarded as equivalent to the expectation-maximization steps of k-means.

Let $G = (\mathcal{Z}, \mathcal{A})$ be a graph where the set of nodes is given by the samples of $\mathcal{Z}$ and the adjacency relation $\mathcal{A}$ is issued by the problem definition. For instance, in the original work, IW is applied in image segmentation and road network analysis. In image segmentation, the pixels represent the nodes, while the edges are given by the 4-neighborhood adjacency between pixels. On the other hand, in road network analysis, the reference points in the cities represent the nodes, and the roads linking such points represent the edges.

For a fixed number of clusters k, let $\mathcal{S} = \{r_1, r_2, \ldots, r_k\}$ be the number of initial cluster prototypes or seeds which are selected randomly from $\mathcal{Z}$ at the outset of the algorithm. The IW algorithm frames the problem of clustering with connectivity constraint as the minimization of a dissimilarity measure between samples and cluster prototypes. In the context of k-means, this dissimilarity measure is given the Euclidean distance between samples and cluster prototypes. Conversely, in IW, it is represented by the value given by a path-cost function (*i.e.*, connectivity function), which measures the connectedness between samples and cluster prototypes through a path in G. The OPF assigns each sample $s \in \mathcal{Z}$ to its most closely connected prototype, that is, the prototype that offers the minimum path-cost among all paths connecting a prototype $r \in \mathcal{S}$ and s. Thus, the objective of IW is to minimize the total path-cost given by

$$\sum_{s \in \mathcal{Z}} \min_{\pi_s \in \Pi_s} f(\pi_s) \tag{3.3}$$

where Π_s represents the set of all paths in G rooted at $\mathcal{S}$ with terminus s and f is a smooth path-cost function. The connectivity function used by IW for all applications was f_{sum} which determines the sum of all arc-weights along a path. Algorithm 12 depicts the IW algorithm to solve the aforementioned optimization problem. The algorithm starts by selecting k seeds $r_i, i \in \{1, \dots, k\}$ from the dataset $\mathcal{Z}$ in Line 1. Then variables *iter* and *converged* are initialized to control the number of iterations and algorithm's convergence, respectively. Next, the main loop in Lines 3–20 carries out steps (*ii*) and (*iii*). In Lines 4–9, priority queue $\mathcal{Q}$, and cost map C and label map L are initialized. The cluster sets $\mathcal{C}_i$ are initialized with seeds $r_i, i \in \{1, \dots, k\}$ followed by the insertion of all samples $s \in \mathcal{S}$ into $\mathcal{Q}$. In Lines 11–16, the graph partitioning occurs assigning each sample to the cluster set containing its most strongly connected seed based on the connectivity function f_{sum}. Next, in Lines 17–19, a seed $r_i \in \mathcal{S}$ is recomputed by selecting the sample closest to the mean feature vector μ_i for each cluster set $\mathcal{C}_i, i \in \{1, \dots, k\}$. Lastly, when either

Algorithm 12: Iterated Watersheds algorithm for f_{sum}

```
Input    : Graph $G = (\mathcal{Z}, \mathcal{A})$, seed set $\mathcal{S}$ with labeling function $\lambda$, descriptor
           function $v$, number of seeds $k \geq 1$, and maximum number $T \geq 1$ of
           iterations
Output   : Label map $L$
Auxiliar : Priority queue $\mathcal{Q}$, cluster sets $\mathcal{C}_i, \forall r_i \in \mathcal{S}$, $i = 1, \dots, k$, maps $C$ and $L$,
           and variables $tmp$ and $converged$
1  Pick randomly $k$ seeds from $\mathcal{Z}$: $\mathcal{S} = \{r_1, \dots, r_k\}$
2  iter ← 1, converged ← false
3  while iter $\leq T$ and converged = false do
4      $\mathcal{Q} = \emptyset$
5      foreach $s \in \mathcal{Z}$ do
6          $C(s) \leftarrow +\infty, L(s) \leftarrow 0$
7          if $s = r_i \in \mathcal{S}$, $i \in \{1, \dots, k\}$ then
8              $C(s) \leftarrow 0$, $L(s) \leftarrow i$
9              $\mathcal{C}_i \leftarrow \emptyset$, $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{r_i\}$
10         Insert $s$ in $\mathcal{Q}$
11     while $\mathcal{Q} \neq \emptyset$ do
12         Remove $s$ from $\mathcal{Q}$, so that $s = \text{argmin}_{t \in Q}\{C(t)\}$ and $\mathcal{C}_{L(s)} \leftarrow \mathcal{C}_{L(s)} \cup \{s\}$
13         foreach $(s, t) \in \mathcal{A} \mid t \in \mathcal{Q}$ do
14             $tmp \leftarrow C(s) + \|v(t) - \mu_{L(s)}\|$
15             if $tmp < C(t)$ then
16                 $C(t) \leftarrow tmp, L(t) \leftarrow L(s)$
17     $\mathcal{S}_{prev} \leftarrow \mathcal{S}$, $\mathcal{S} \leftarrow \emptyset$
18     foreach $i \in \{1, \dots, k\}$ do
19         $r_i \leftarrow \text{argmin}_{s \in \mathscr{C}_i}\{\|v(s) - \mu_i\|\}$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{r_i\}$
20     converged ← ($\mathcal{S} = \mathcal{S}_{prev}$), iter ← iter + 1
21 return $L$
```

the maximum number of iterations or convergence is achieved the label map L generated during the last iteration is returned.

The IW algorithm was applied in image segmentation and road network analysis, obtaining outstanding results. In image segmentation, the IW was compared to k-means, spectral clustering, and isoperimetric partitioning on the Weizmann 1-Object and 2-Object datasets obtaining superior performance based on four popular metrics, namely *adjusted mutual information* (AMI), *adjusted rand index* (ARI), *F-score* (F), and *clustering accuracy* (CA). On the other hand, in road network analysis, the IW displayed the best performance compared to k-means and greedy k-centers on identifying "ideal" points to establish emergency stations on road network maps.

3.4 Clustering by OPF

The graph-based clustering algorithms presented in Section 3.1.3 do not exploit optimum connectivity between samples and seeds for cluster delineation. In this context, several OPF-based clustering algorithms have been introduced to bridge this gap under different approaches. The OPF-based clustering solutions can be broadly categorized into density-based and centroid-based algorithms.

Rocha *et al.* [68] introduced a first clustering method based on optimum connectivity – the maxima of a probability density function (pdf) compete among themselves to conquer the remaining samples of the dataset, and each maximum (dome of the pdf) defines a cluster as an optimum-path tree rooted on it. The pdf is estimated from a k-Nearest Neighbor (kNN) graph, and the choice of k is attained by finding the solution that minimizes a normalized graph-cut measure. This algorithm relies on the pdf estimation for a suitable choice of an interval $[k_{\text{min}}, k_{\text{max}}]$ in which the best value of k must be found; however, the pdf estimation may become computationally prohibitive for large datasets. Moreover, it does not provide user control in determining the number of desired clusters. This algorithm is further described in Section 2.5.2. The selection of the best value of k within an interval $[k_{\text{min}}, k_{\text{max}}]$ is carried out through exhaustive search, which may become computationally intractable for datasets with millions of samples. Costa *et al.* [24] propose nature-inspired optimization techniques to speed up the selection of k for pdf estimation with application to intrusion detection in computer networks.

Cappabianco *et al.* [15] extended the OPF-based clustering approach for large datasets by subsampling training samples, generating candidate solutions, and selecting the most plausible one. The authors demonstrated the advantages of the method for MR-brain tissue segmentation.

Montero and Falcão [27] propose a two-level divide-and-conquer clustering approach based on density-based OPF clustering. This algorithm is well suited to handle large datasets. Firstly, the dataset is divided into a reasonable number of disjoint blocks. Next, OPF-based clustering [68] is used to cluster the samples in each block, such that each block ends up with a set of prototypes that summarize the block's information. These prototypes are then joined together to create a new dataset in which OPF-based clustering is executed once again, obtaining a good approximation of the original dataset's underlying partition. Lastly, label propagation occurs in a cascade manner, from prototypes to the remaining elements in each block.

Chen *et al.* [18] presented an improved OPF-based clustering algorithm for segmentation of remote sensing images based on the principle that cluster centers display high local densities, whereas samples surrounding centers usually exhibit relatively low local densities. In addition, cluster centers are often located far away from samples with higher local densities. Thus, cluster centers are characterized by their densities and the distances between them and samples with higher densities. Following this line of thought, the probability density function of OPF-based clustering is modified for each sample in the dataset to include the distance to samples with higher densities.

Afonso *et al.* [3] introduced a multi-layered OPF-based clustering algorithm inspired by hierarchical clustering. This algorithm, called *Deep Optimum-Path Forest*, builds a model comprised of a fixed number of stacked layers, such that the last layer contains the desired number of clusters. Each layer of the model partitions the input dataset, consisting of the prototypes obtained from the previous layer's clustering, into an optimum-path forest using density-based OPF-based clustering. This procedure is repeated until the last layer is reached. Recently, this algorithm was used in [4] to design visual dictionaries for the automatic identification of Parkinson's disease.

Soor *et al.* [76] propose the *Iterated Watersheds* (IW), a graph-based clustering algorithm based on iterative applications of watershed transforms in a feature space based in a sequence of OPF executions from sets of enhanced cluster prototypes (seeds). This algorithm is a modified version of k-means with connectivity constraints, which turns out to be a particular configuration of the iterative optimum-path forest (IOPF) framework proposed herein. This algorithm is further detailed in Section 3.3.

3.5 Concluding Remarks

This chapter provides the related work concerning this master's work. It starts with a discussion of partitional, hierarchical and graph-based clustering algorithms. Next, the k-means algorithm, a popular partitional clustering algorithm, is described. Afterward, the Iterated Watersheds (IW) algorithm, regarded as a particular implementation of the IOPF framework, is presented. Lastly, a study of OPF-based clustering algorithms is introduced, highlighting the most relevant works. The next chapter comprehensively introduces the Iterative Optimum-Path Forest (IOPF) framework.

Chapter 4

Iterative Optimum-Path Forest

In this chapter[1], we comprehensively introduce *Iterative Optimum-Path Forest*, hereafter referred to as IOPF, a graph-based iterative clustering framework consisting of four components: (*i*) sampling of the initial seed set $\mathcal{S}$, (*ii*) graph partition by OPF from $\mathcal{S}$ in a graph derived from the dataset, (*iii*) recomputation of $\mathcal{S}$ based on the previous graph partition, and after multiple iterations of the last two steps, (*iv*) selection of the forest with the lowest total path cost among all executions.

4.1 Notations and Definitions

Let $\mathcal{Z}$ be a dataset such that for every sample $s \in \mathcal{Z}$ there is a feature vector $v(s) \in \mathbb{R}^m$. For a given adjacency relation $\mathcal{A} \subseteq \mathcal{Z} \times \mathcal{Z}$, the pair $G = (\mathcal{Z}, \mathcal{A})$ defines a graph. The adjacency relation $\mathcal{A}$ can be defined in different ways, based on the definition of the problem. In some cases, the adjacency relation of the graph is given beforehand, whereas in some other cases it must be built from scratch. For instance, if $\mathcal{Z}$ is the set of pixels $s = (x_s, y_s)$ in the bi-dimensional domain of an image, $\mathcal{A}$ may be defined as $\mathcal{A}_r = \{(s,t) \in \mathcal{Z} \times \mathcal{Z} \mid 1 \leq \|(x_t, y_t) - (x_s, y_s)\| \leq r\}$. In this regard, the most notable adjacency relations on this domain are $\mathcal{A}_1$ and $\mathcal{A}_{\sqrt{2}}$, referred to 4- and 8-neighborhood, respectively. As r increases, the local image feature space is explored with less spatial constraint. For arbitrary datasets, we may define $\mathcal{A}$ as follows:

1. $\mathcal{A} = \{(s,t) \in \mathcal{Z} \times \mathcal{Z} \mid s, t \in \mathcal{Z} \wedge s \neq t\}$, so that G represents a complete graph; or
2. $\mathcal{A} = \{(s,t) \in \mathcal{Z} \times \mathcal{Z} \mid v(t)$ is a k-nearest neighbor of $v(s)\}$, for a fixed k

However, in (2) it is important to make sure that all nodes in $\mathcal{Z}$ are reachable from any seed in the seed set; therefore, two conditions should be met: (*i*) if $(s,t) \in \mathcal{A}$, then $(t,s) \in \mathcal{A}$; and (*ii*) G must be a single component.

The objective of an IOPF-based solution is to estimate the graph partition that minimizes the total path-cost given by the sum of path-costs between samples and their most strongly connected seeds in G. The minimization of this objective function is addressed

[1]The text of this chapter is to appear as part of a book chapter entitled "An Iterative Optimum-Path Forest Framework for Clustering" in [34]

following an iterative approach consisting in, given a fixed number of groups k, partitioning the graph G into k optimum-path trees by multiple OPF executions from enhanced sets of seeds. Each OPF execution will output a triplet (L,C,P) consisting of a label map L, a cost map C, and a predecessor map P, leading to the computation of the total path-cost (TPC) given by $\sum_{\forall x \in \mathcal{Z}} C(x)$. The set of enhanced seeds is computed, selecting the samples closest to each optimum-path tree's mean feature vector, and the iterative procedure is repeated until either seed set convergence is achieved or a fixed maximum number of iterations is reached. A search is carried out across all iterations to identify the triplet (L, C, P), providing the lowest total path-cost, which is returned as the final output of the framework. Figure 4.1 depicts the pipeline of the IOPF framework where initial seeds are selected randomly, and seed convergence is achieved at the fourth iteration. In this example, the third iteration minimizes $\sum_{\forall s \in \mathcal{Z}} C(s)$, and hence it is returned as the final clustering.

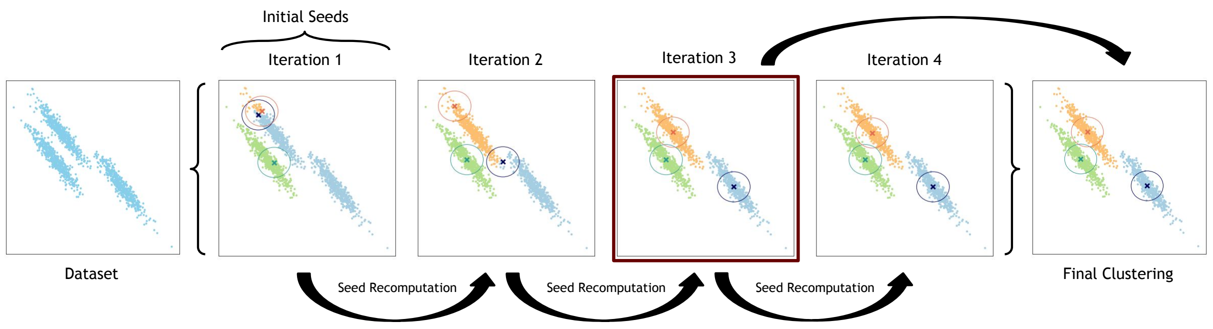


Figure 4.1: IOPF pipeline. Initial seeds are selected randomly and seeds recomputed at the end of each single OPF execution. In the example, seed convergence is attained at the fourth iteration. Lastly, the partition that minimizes $\sum_{\forall s \in \mathcal{Z}} C(s)$ across all iterations is returned as the final clustering.

Based on these definitions, we introduce the individual framework components in the following sections.

4.2 Seed Set Selection

The performance of seed-based algorithms, like k-means, is highly sensitive to the choice of the initial seed set, given that these techniques rely on a local optimization approach. Thus, the initial seed set selection represents a critical element in the framework design. In this work, we present two strategies to address this problem.

For a fixed number of groups, k, the IOPF framework starts off by selecting a set of samples $\mathcal{S} \subset \mathcal{Z}$, such that $|\mathcal{S}| = k$. The first strategy selects randomly the elements of $\mathcal{S}$ This initialization technique provides a different result for each run, therefore allowing to improve results through repetition (restarting) of the algorithm.

As second strategy, we propose an initial seed set sampling that builds $\mathcal{S}$, with one seed per object, by searching the least strongly connected samples (after discarding outliers) by a sequence of OPF executions. Algorithm 13 outlines such iterative procedure.

Given a graph $G = (\mathcal{Z}, \mathcal{A})$, number of seeds $k \geq 1$, an estimated percentage of outliers $0 < h < 1$, and the sum path-cost function (f_{sum}). The algorithm starts in Line 1 with an initial seed set $\mathcal{S}$ consisting of an arbitrary node $s \in \mathcal{Z}$, which is discarded after the first OPF execution to obtain more statistically consistent seeds. Thus, only the seeds acquired at the end of each OPF execution are considered towards establishing the initial seed set $\mathcal{S}$. During each iteration in Lines 2–20, the OPF execution is interrupted after $(1 - h) \times |\mathcal{Z}|$ nodes have been processed (*i.e.*, nodes removed from priority queue $\mathcal{Q}$) in Lines 10–15. Next, in Line 14, the last node s removed from $\mathcal{Q}$ is selected as the next seed and inserted into $\mathcal{S}$. This procedure is repeated with the updated seed set $\mathcal{S}$ until the desired number of seeds is attained and lastly, the seed set $\mathcal{S}$ is returned in Line 21. The nodes are removed from $\mathcal{Q}$ in a non-decreasing order of their path-cost values, hence, all nodes left in $\mathcal{Q}$ have a path-cost value greater than or equal to the path-cost value of the last removed node s. These nodes are "farther" away from the seeds contained in $\mathcal{S}$ than is s and therefore can be regarded as outliers. Figure 4.2 depicts the functioning of the seed selection algorithm where the initial seed is selected randomly and discarded after the first seed was identified by means of an OPF execution. Later, the seed set is progressively built based on a sequence of OPF executions.

Algorithm 13: Seed set selection algorithm for f_{sum}

```
Input    : Graph $G = (\mathcal{Z}, \mathcal{A})$, number of seeds $n \geq 1$, and estimated percentage
           of outliers $0 < h < 1$
Output   : Seed set $\mathcal{S}$
Auxiliar : Priority queue $\mathcal{Q} = \emptyset$, maps $C$ and $P$, and variables $tmp$ and $first\text{-}time$
1  $\mathcal{S} \leftarrow$ random element from $\mathcal{Z}$, $first\text{-}time \leftarrow$ true
2  while $|\mathcal{S}| < k$ do
3      foreach $s \in \mathcal{Z}$ do
4          $C(s) \leftarrow +\infty, P(s) \leftarrow nil$
5          if $s \in \mathcal{S}$ then
6              $C(s) \leftarrow 0$
7          Insert $s$ in $\mathcal{Q}$
8      while $\mathcal{Q} \neq \emptyset$ do
9          Remove $s$ from $\mathcal{Q}$, such that $s = \text{argmin}_{t \in \mathcal{Q}}\{C(t)\}$
10         if $h \times |\mathcal{Z}| \geq |\mathcal{Q}|$ then
11             if first-time then
12                 $\mathcal{S} \leftarrow \{s\}$, first-time $\leftarrow$ false
13             else
14                 $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
15             $\mathcal{Q} \leftarrow \emptyset$ and break
16         foreach $(s, t) \in \mathcal{A} \mid q \in \mathcal{Q}$ do
17             $w(s, t) = \|v(t) - v(s)\|$
18             $tmp \leftarrow C(s) + w(s, t)$
19             if $tmp < C(t)$ then
20                 $C(t) \leftarrow tmp$, $P(t) \leftarrow s$
21 return $\mathcal{S}$
```

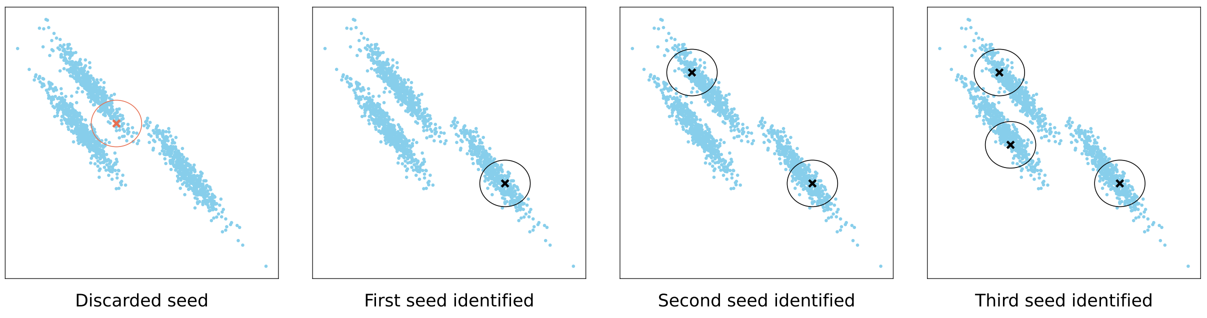


Figure 4.2: Seed selection procedure. It starts by randomly choosing a seed, which is then discarded after the first seed has been identified. Next, the second and third seeds are progressively identified and the resultant seed set is returned.

4.3 Clustering by Optimum-Path Forest

Once the initial seed set $\mathcal{S}$ has been finally established, an iterative procedure based on a sequence of OPF executions is carried out until either convergence or a preset maximum number of iterations $T > 0$ is achieved. In this context, convergence is attained when the seed sets obtained in consecutive iterations are equal (*i.e.*, have the same elements), since this will lead to the same graph partition.

Given a fixed number of groups k, the objective of each single OPF execution is to exploit optimum connectivity by partitioning the graph $G = (\mathcal{Z}, \mathcal{A})$ into k optimum-path trees (clusters) $\mathcal{T}_i, i \in \{1, \ldots, k\}$ which altogether form an optimum-path forest. Each optimum-path tree $\mathcal{T}_i$ is rooted at seed $s_i \in \mathcal{S}, i \in \{1, \ldots, k\}$, such that the nodes constituting $\mathcal{T}_i$ are more strongly connected to s_i, than to any other seed in $\mathcal{S}$.

Let $\mathcal{C}_i \subset \mathcal{Z}$ be the cluster defined by optimum-path tree $\mathcal{T}_i, i \in \{1, \ldots, k\}$, such that $\cup_{i=1}^{k} \mathcal{C}_i = \mathcal{Z}$ and $\cap_{i=1}^{k} \mathcal{C}_i = \emptyset$. By assigning a distinct group label $i \in \{1, \ldots, k\}$ to each seed $s \in \mathcal{S}$, the OPF algorithm propagates the corresponding label $L(s)$ to its most closely connected samples in $\mathcal{Z}$, creating a label map L.

In this work, we address two ways of determining the arc-weights, namely *fixed* and *dynamic* arc-weights. The arc-weight for an edge $(s, t) \in \mathcal{A}$ is represented by $w(s, t)$. For fixed arc-weights, it is determined through the Euclidean distance of the feature vectors of s and t – *i.e.*, $w(s, t) = \|v(t) - v(s)\|$, whereas for dynamic arc-weights we use instead

$$\begin{aligned} w(s,t) &= \|v(t) - \mu_{L(s)}\|, \\ \mu_{L(s)} &= \frac{1}{|\mathcal{C}_{L(s)}|} \sum_{\forall x \in \mathcal{C}_{L(s)}} v(x) \end{aligned} \tag{4.1}$$

where $\mathcal{C}_L(s)$ is the growing cluster (optimum-path tree) that contains s by the time a path $\pi_s \cdot \langle s, t\rangle$ reaches a node $t \in \mathcal{Z} \setminus \cup_{i=1}^{k} \mathcal{C}_i$ under evaluation. The dynamic approach leverages cluster information as the trees grow from the seed set, aiming to obtain more compact clusters since centroids are recomputed as the trees grow, and optimum connectivity is evaluated between such centroids and the remaining samples.

4.4 Seed Recomputation

The seed recomputation stage occurs after each single OPF execution. This procedure aims to obtain enhanced seed sets using clustering information. The seeds are recomputed by selecting the sample whose feature vector is the closest to the mean feature vector of their corresponding cluster. This seed recomputation strategy stems from the idea that seeds that minimize the total path-cost may be located in dense regions of nodes, therefore, an approximation of such regions can be obtained through the centroid estimation for each cluster.

For a fixed number of groups k, let $\mathcal{T}_{ij}, i \in \{1, \ldots, k\}$ be the k optimum-path trees generated by the seed set $\mathcal{S}_j$ after a single OPF execution during iteration j. The nodes comprising the optimum-path tree $\mathcal{T}_{ij}$ define the cluster $\mathcal{C}_{ij}$ for all $i \in \{1, \ldots, k\}$. Therefore, each element $s_{i,j+1}, i \in \{1, \ldots, k\}$ of the seed set for the next iteration $\mathcal{S}_{j+1}$ is calculated as

$$\begin{aligned} s_{i,j+1} &= \underset{s \in \mathcal{C}_{ij}}{\operatorname{argmin}} \{\|v(s) - \mu_{ij}\|\}, \\ \mu_{ij} &= \frac{1}{|\mathcal{C}_{ij}|} \sum_{\forall x \in \mathcal{C}_{ij}} v(x) \end{aligned} \tag{4.2}$$

4.5 Returning the Forest with Lowest Total Path-Cost

Since the objective of the proposed framework is to minimize the objective function given by the total path-cost $\sum_{\forall s \in \mathcal{Z}} C(s)$, the triplet (L, C, P) consisting of label map L, cost map C, and predecessor map P, that leads to the lowest total path-cost must be returned.

This stage is carried out once either seed set convergence has been achieved or the maximum number of iterations has been exhausted. Conversely to IW, the triplet corresponding to the last iteration is not returned since the total path-cost between consecutive iterations is not monotonically decreasing, therefore, a search must be carried out across the triplets of all iterations to identify the one that leads to the lowest total path-cost.

4.6 Algorithm Outline

Algorithm 14 depicts the pseudocode of IOPF with dynamic arc-weight estimation for $f_{\max}$. The algorithm starts off by initializing the cost map C^*, label map L^*, and predecessor map P^* in Lines 1–2. These maps are updated throughout the execution of the algorithm aiming to minimize the total path-cost value derived from C^*. Later, in Line 3, k seeds $r_i \in \mathcal{Z}, i \in \{1, \ldots, k\}$ are picked, such that each is uniquely identified as belonging to one among k clusters.

In Lines 4–24, Algorithm 14 computes k optimum-path trees (clusters) from a seed set $\mathcal{S}$, recomputes the seed set $\mathcal{S}$ in Lines 18–20, and repeats both operations until either the convergence criterion is met or a preset maximum number of iterations T is reached. In Line 5, it sets the dynamic sets $\mathcal{C}_i, i \in \{1, \ldots, k\}$ to empty. In Lines 8–11, it initializes label map L, cost map C, and predecessor map P, and inserts all nodes into a priority queue $\mathcal{Q}$. In Lines 12–17, the algorithm maintains the dynamic sets $\mathcal{C}_i, i \in \{1, \ldots, k\}$, label map L, cost map C, and predecessor map P. At each iteration of this loop, a node s of minimum cost $C(s)$ is removed from $\mathcal{Q}$ and inserted into the corresponding dynamic set $\mathcal{C}_{L(s)}$ in Line 13. At this moment, the current path π_s is optimum (*i.e.*, its cost is minimum among all possible paths with terminus s and rooted at $\mathcal{S}$). In Lines 14–17, node s offers an extended path $\pi_s \cdot \langle s, t\rangle$ to a node $t \in \mathcal{Q}$ (*i.e.*, $t \in \mathcal{Z} \setminus \cup_{i=1}^{k} \mathcal{C}_i$). The path value $f_{\max}(\pi_s \cdot \langle s, t\rangle)$ is computed and stored in *tmp* in Line 15. If *tmp* is less than the cost $C(t)$ of the current path π_t in P, then π_t is replaced by $\pi_s \cdot \langle s, t\rangle$ in Line 17 by updating the values of the predecessor $P(t)$, cost $C(t)$, and label $L(t)$ corresponding to t to s, *tmp*, and $L(s)$, respectively.

In Lines 18–20, the algorithm saves the current seed set $\mathcal{S}$ into $\mathcal{S}'$, resets the seed set $\mathcal{S}$ to empty and then recomputes it by selecting the nodes $r_i \in \mathcal{C}_i$ that are closest to the

Algorithm 14: IOPF for $f_{\max}$ and dynamic arc-weight estimation

Input : Graph $G = (\mathcal{Z}, \mathcal{A})$, seed set $\mathcal{S}$ with labeling function λ, number of seeds $k \geq 1$, and maximum number of iterations $T \geq 1$

Output : Label L^*, cost C^*, and predecessor P^* maps

Auxiliar: Priority queue $\mathcal{Q}$, dynamic sets $\mathcal{C}_i, \forall r_i \in \mathcal{S}$, $i = 1, \ldots, k$, maps C, L and P, and variables *iter*, *converged* and *tmp*

```
1  foreach $s \in \mathcal{Z}$ do
2      $C^*(s) \leftarrow +\infty, L^*(s) \leftarrow 0, P^*(s) \leftarrow nil$
3  Pick $k$ seeds from $\mathcal{Z}$: $\mathcal{S} = \{r_1, \ldots, r_k\}$, iter $\leftarrow 1$, converged $\leftarrow$ false
4  while iter $\leq T$ and converged = false do
5      $\mathcal{C}_i \leftarrow \emptyset, \forall i \in \{1, \ldots, k\}$
6      $\mathcal{Q} = \emptyset$
7      foreach $s \in \mathcal{Z}$ do
8          $C(s) \leftarrow +\infty, L(s) \leftarrow 0, P(s) \leftarrow nil$
9          if $s = r_i \in \mathcal{S}$, $i \in \{1, \ldots, k\}$ then
10             $C(s) \leftarrow 0$, $L(s) \leftarrow i$
11         Insert $s$ in $\mathcal{Q}$
12     while $\mathcal{Q} \neq \emptyset$ do
13         Remove $s$ from $\mathcal{Q}$, so that $s = \text{argmin}_{t \in \mathcal{Q}}\{C(t)\}$ and $\mathcal{C}_{L(s)} \leftarrow \mathcal{C}_{L(s)} \cup \{s\}$
14         foreach $(s, t) \in \mathcal{A} \mid t \in \mathcal{Q}$ do
15             $tmp \leftarrow \max\{C(s), \|v(t) - \mu_{L(s)}\|\}$
16             if $tmp < C(t)$ then
17                 $C(t) \leftarrow tmp, L(t) \leftarrow L(s)$, $P(t) \leftarrow p$
18     $\mathcal{S}_{prev} \leftarrow \mathcal{S}$, $\mathcal{S} \leftarrow \emptyset$
19     foreach $i \in \{1, \ldots, k\}$ do
20         $r_i \leftarrow \text{argmin}_{s \in \mathcal{C}_i}\{\|v(s) - \mu_i\|\}$ and $\mathcal{S} \leftarrow \mathcal{S} \cup \{r_i\}$
21     converged $\leftarrow (\mathcal{S} = \mathcal{S}_{prev})$
22     if $\sum_{\forall s \in \mathcal{Z}} C(s) < \sum_{\forall s \in \mathcal{Z}} C^*(s)$ then
23         $(L^*, C^*, P^*) \leftarrow (L, C, P)$
24     iter $\leftarrow$ iter + 1
25 return $(L^*, C^*, P^*)$
```

mean feature vector of their resulting optimum-path tree $\mathcal{T}_i, i \in \{1, \ldots, k\}$. The mean feature vector is defined as the arithmetic mean of the feature vectors of the elements contained in $\mathcal{C}_i$. Next, in Line 21, $\mathcal{S}'$ is compared to $\mathcal{S}$ to test for convergence and the result of this comparison is saved in *converged.* In Lines 22–23, we test whether the map C provides a lesser total path-cost value than the map C^*, if that is so, then, L^*, C^*, and P^* are updated with the maps L, C, and P, respectively. Lastly, the tuple (L^*, C^*, P^*) with minimum total path-cost value among all iterations is returned in Line 25.

The algorithm for IOPF with fixed arc-weights differs only in Line 15, where the arc-weight between nodes s and t is estimated as $\|v(t) - v(s)\|$. Thus, Line 15 should be modified to $tmp \leftarrow \max\{C(s), \|v(t) - v(s)\|\}$.

In order to reduce the complexity of the algorithm, we can store the mean feature vector of each dynamic set and its size, so that these measures can efficiently be updated during label propagation. Therefore, each time a new element s is added to the dynamic set, the mean feature vector and the dynamic set size are updated as in Equation 4.3, where $v(s)$ represents the feature vector of s, μ_{prev} and μ_{next} are the previous and next mean feature vectors, while n_{prev} and n_{next} represent the size of the dynamic set before and after the update.

$$\begin{aligned} \mu_{next} &= \mu_{prev} + \frac{v(s) - \mu_{prev}}{n_{prev} + 1} \\ n_{next} &= n_{prev} + 1 \end{aligned} \tag{4.3}$$

4.7 Application to Object Delineation

Since IOPF is a generalization of the ISF framework from the image domain to the feature space, its application in the image domain is straightforward. We call the methods for object delineation *Iterative Dynamic Trees* (IDT). A two-dimensional image is a pair $(\mathcal{D}_\mathcal{I}, \mathbf{I})$, such that $\mathbf{I}(p)$ assigns local image features (*e.g.*, color space components) for each pixel $p \in \mathcal{D}_\mathcal{I} \subset \mathbb{Z}^2$. An image can be rendered as a graph $(\mathcal{N}, \mathcal{A})$ under various configurations, depending upon how nodes $\mathcal{N} \subseteq \mathcal{D}_\mathcal{I}$ and adjacency relation $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$ are defined. We define pixels as nodes ($\mathcal{N} = \mathcal{D}_\mathcal{I}$), such that $\mathbf{I}(p)$ represents the CIELab color components of pixel p, and the 8-neighborhood relation defines the arcs.

Given a seed set $\mathcal{S}$, we wish to partition the image into objects such that the pixels enclosed by an object are more closely connected to the seed within the object than to any other seed. A unique object identifier is given to each seed $p \in \mathcal{S}$ by a labeling function $\lambda(p) \in \{1, \ldots, c\}$, where c is the number of objects.

Therefore, this application focuses on object delineation as a superpixel segmentation task that defines each object by a single superpixel. The IDT algorithm consists of four steps: (*i*) initial seed estimation with one seed per object, (*ii*) object delineation as an optimum-path tree, (*iii*) seed set improvement and the loop of steps (*ii*)–(*iii*) for a preset number of iterations or up to seed set convergence. After that, step (*iv*) completes the process by selecting the optimum-path forest from loop (*ii*)–(*iii*) whose total path cost is minimum. The IDT algorithm may be regarded as a new method based on the *Iterative Spanning Forest* (ISF) framework [78], which adds step (*iv*) and drastically reduces the number of superpixels to the number of objects. In ISF, more accurate delineation can be achieved by dynamic arc-weight estimation, as the optimum-path trees grow – a strategy that has been demonstrated for superpixel segmentation [12] and interactive object segmentation [14]. In IDT, we exploit this property in ISF for unsupervised object segmentation for the first time.

All the framework components presented in previous sections are valid for this application. However, given the nature of the problem, some other strategies can be introduced as components for the framework. Following this line of thought, we present a new seed recomputation strategy in the image domain. In Section 4.4, during iteration j, new seeds are selected as the nodes closest to the mean feature vector for each optimum-path tree $\mathcal{T}_{ij}, i \in \{1, \ldots, k\}$. Nevertheless, in the image domain, we may also select the new seeds as the nodes closest to the mean pixel of each optimum-path tree. The mean pixel is defined as the arithmetic mean of pixel coordinates of the elements of clusters $\mathcal{C}_{ij}, i \in \{1, \ldots, k\}$. Thus, each seed $r_{i,j+1}, i = 1, \ldots, n$ for iteration $j+1$ is determined as

$$r_{i,j+1} = \underset{p \in \mathcal{C}_i}{\operatorname{argmin}}\{\|p - \frac{1}{|\mathcal{C}_i|} \times \sum_{\forall q \in \mathcal{C}_i} q\|\} \quad (4.4)$$

Figure 4.3 depicts the application of the IDT algorithm from a randomly selected initial seed set and maximum number of iterations set to 20. The seed set is recomputed at the end of each single iteration and the optimum-path forest with the lowest total cost is returned as the final output.

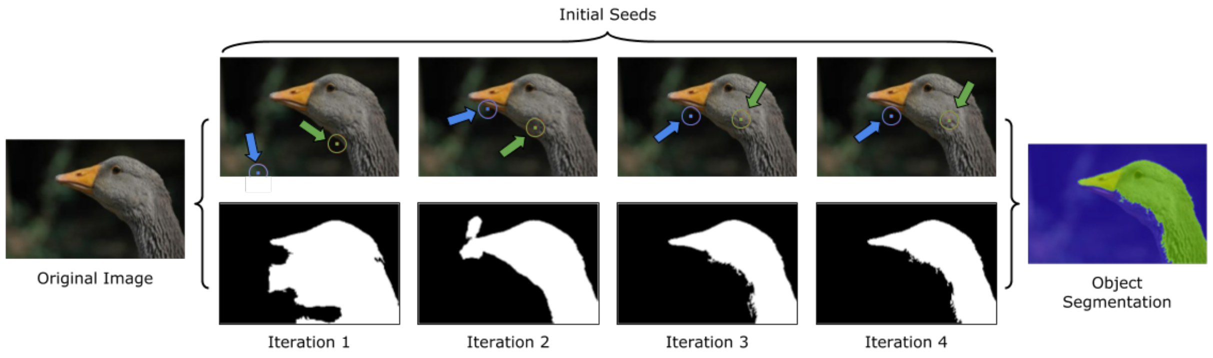


Figure 4.3: Object segmentation using the IDT algorithm.

4.8 Concluding Remarks

In this chapter, we comprehensively present the main contribution of this master's work, the Iterative Optimum-Path Forest (IOPF) framework, describing in detail its constituent components: (*i*) a seed selection strategy, (*ii*) clustering by OPF, (*iii*) a seed recomputation procedure and (*iv*) the return of the forest with lowest total path-cost. Next, an application of the IOPF framework to object delineation is presented, where an alternative seed recomputation method is introduced in the context of images. In the next chapter, we discuss the experiments and results of the IOPF framework applied to three problems that impose different restrictions on the definition of the graph topology.

Chapter 5

Experiments and Results

In this chapter[1], we present three applications to show the robustness and flexiblity of the IOPF framework by designing suitable and effective IOPF-based methods for each problem context. The first application in Section 5.1 addresses the problem of object delineation following an unsupervised approach. The second application in Section 5.2 addresses the analysis of road networks by establishing emergency stations in strategic locations, such that the distance between reference points and emergency stations is minimized. Lastly, the third application in Section 5.3 addresses the problem of clustering synthetic two-dimensional datasets presenting a wide variety of shapes and distributions.

5.1 Object Delineation by Iterative Dynamic Trees

To demonstrate the advantages of step (*iv*) and random seed sampling in step (*i*) in the context of object delineation, we compare four versions of IDT. IDT_1 is the proposed version, as described in Section 4.7 using the connectivity function $f_{\max}$. IDT_2 is IDT_1 without step (iv), it selects the last optimum-path forest after T iterations, as proposed in the original ISF framework and adopted by all previous ISF-based methods, such as DISF [12]. IDT_3 is IDT_1 with grid sampling (seed sampling with uniform distance among seeds) in step (*iii*), as used in some ISF-based approaches, such as DISF [12] and most superpixel segmentation methods. IDT_4 is IDT_1 with seed recomputation based on the mean feature vector instead of the mean pixel, as is detailed in Section 4.4.

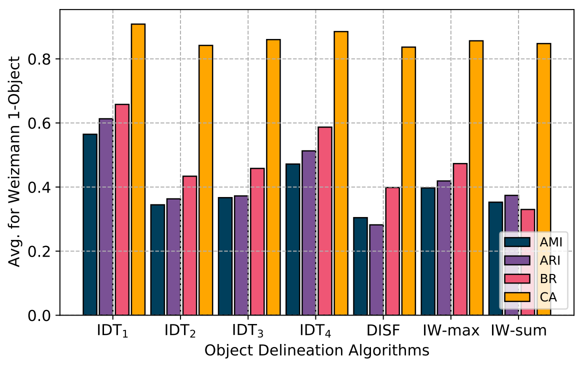
To demonstrate the improvement of IDT for object delineation, we compare it against DISF and IW [76] with two path-cost functions in the IFT algorithm: IW-max computes the cost of a path as the maximum arc weight along it, for fixed arc weights $\|\mathbf{I}(q)-\mathbf{I}(p)\|$, and IW-sum computes the cost of a path as the sum of its arc weights. Like IDT_1 and IDT_2, both IW-based methods start from a random seed set of size equal to the number of desired objects (and background) [76]. DISF starts from a set with 150 seeds selected by grid sampling for all images and reduces the seed set size at every iteration until it reaches the number of desired objects [12]. IW has already been demonstrated to be superior to spectral clustering [80], isoperimetric partitioning [37] and k-means in the task of object

[1]The text of this chapter is to appear as part of a book chapter entitled “An Iterative Optimum-Path Forest Framework for Clustering” in [34]

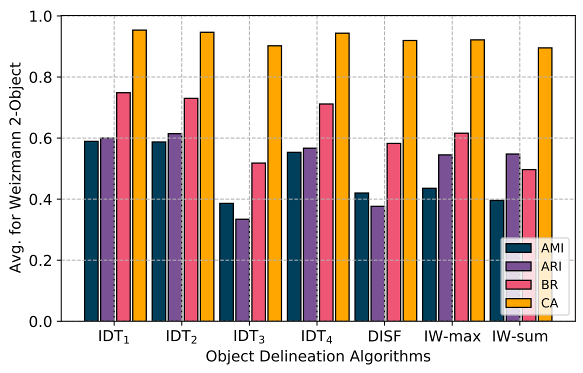
delineation.

Table 5.1: AMI, ARI, Boundary Recall and Cluster Accuracy (Mean +/- Std. Deviation) for Weizmann 1-Object and 2-Object datasets for IDT variants, DISF, IW-max and IW-sum.

	Method	**AMI**	**ARI**	**BR**	**CA**
1-Object	IDT$_1$	**0.564673 $\pm$ 0.283**	**0.613058 $\pm$ 0.317**	**0.657833 $\pm$ 0.241**	**0.908387 $\pm$ 0.091**
	IDT$_2$	0.344623 $\pm$ 0.270	0.363208 $\pm$ 0.323	0.433819 $\pm$ 0.276	0.841895 $\pm$ 0.114
	IDT$_3$	0.366932 $\pm$ 0.307	0.372370 $\pm$ 0.363	0.458131 $\pm$ 0.285	0.860064 $\pm$ 0.107
	IDT$_4$	0.471829 $\pm$ 0.281	0.512906 $\pm$ 0.319	0.586976 $\pm$ 0.243	0.885122 $\pm$ 0.097
	DISF	0.304520 $\pm$ 0.282	0.282088 $\pm$ 0.347	0.398606 $\pm$ 0.296	0.836631 $\pm$ 0.112
	IW-max	0.397320 $\pm$ 0.278	0.419055 $\pm$ 0.318	0.473212 $\pm$ 0.276	0.856288 $\pm$ 0.112
	IW-sum	0.352781 $\pm$ 0.257	0.373990 $\pm$ 0.300	0.330048 $\pm$ 0.243	0.847699 $\pm$ 0.108
2-Object	IDT$_1$	**0.589247 $\pm$ 0.278**	0.600024 $\pm$ 0.345	**0.748527 $\pm$ 0.194**	**0.953605 $\pm$ 0.054**
	IDT$_2$	0.587252 $\pm$ 0.278	**0.614408 $\pm$ 0.333**	0.730065 $\pm$ 0.207	0.946522 $\pm$ 0.064
	IDT$_3$	0.386087 $\pm$ 0.279	0.334149 $\pm$ 0.328	0.518125 $\pm$ 0.263	0.902305 $\pm$ 0.100
	IDT$_4$	0.553327 $\pm$ 0.274	0.566872 $\pm$ 0.324	0.711618 $\pm$ 0.204	0.943516 $\pm$ 0.064
	DISF	0.420036 $\pm$ 0.295	0.376453 $\pm$ 0.352	0.582483 $\pm$ 0.263	0.919615 $\pm$ 0.078
	IW-max	0.435559 $\pm$ 0.330	0.544933 $\pm$ 0.311	0.615948 $\pm$ 0.231	0.921671 $\pm$ 0.086
	IW-sum	0.395757 $\pm$ 0.242	0.347743 $\pm$ 0.299	0.496769 $\pm$ 0.224	0.895421 $\pm$ 0.097



(a) Weizmann 1-Object



(b) Weizmann 2-Object

Figure 5.1: Results obtained in each dataset for AMI, ARI, BR and CA. (a) Weizmann 1-Object dataset, (b) Weizmann 2-Object dataset.

For evaluation of object segmentation, we use the Weizmann 1-Object and 2-Object datasets [7], containing 100 images each, along with ground-truth segmentations. Images in these datasets (available at `http://www.wisdom.weizmann.ac.il/~vision/Seg_Evaluation_DB/`) depict one or two objects in the foreground. For assessment of the methods, we use four popular effectiveness measures: (*i*) *Adjusted Mutual Information* (AMI) [79], which is an adjustment of the Mutual Information (MI) score to account for chance, (*ii*) *Adjusted Rand Index* (ARI) [42], which determines the Rand index (RI) score adjusted for chance, (*iii*) *Boundary Recall* (BR), which measures boundary adherence [1], and (*iv*)

Cluster Accuracy [30], which measures the degree of intersection between predicted and ground-truth segmentation.

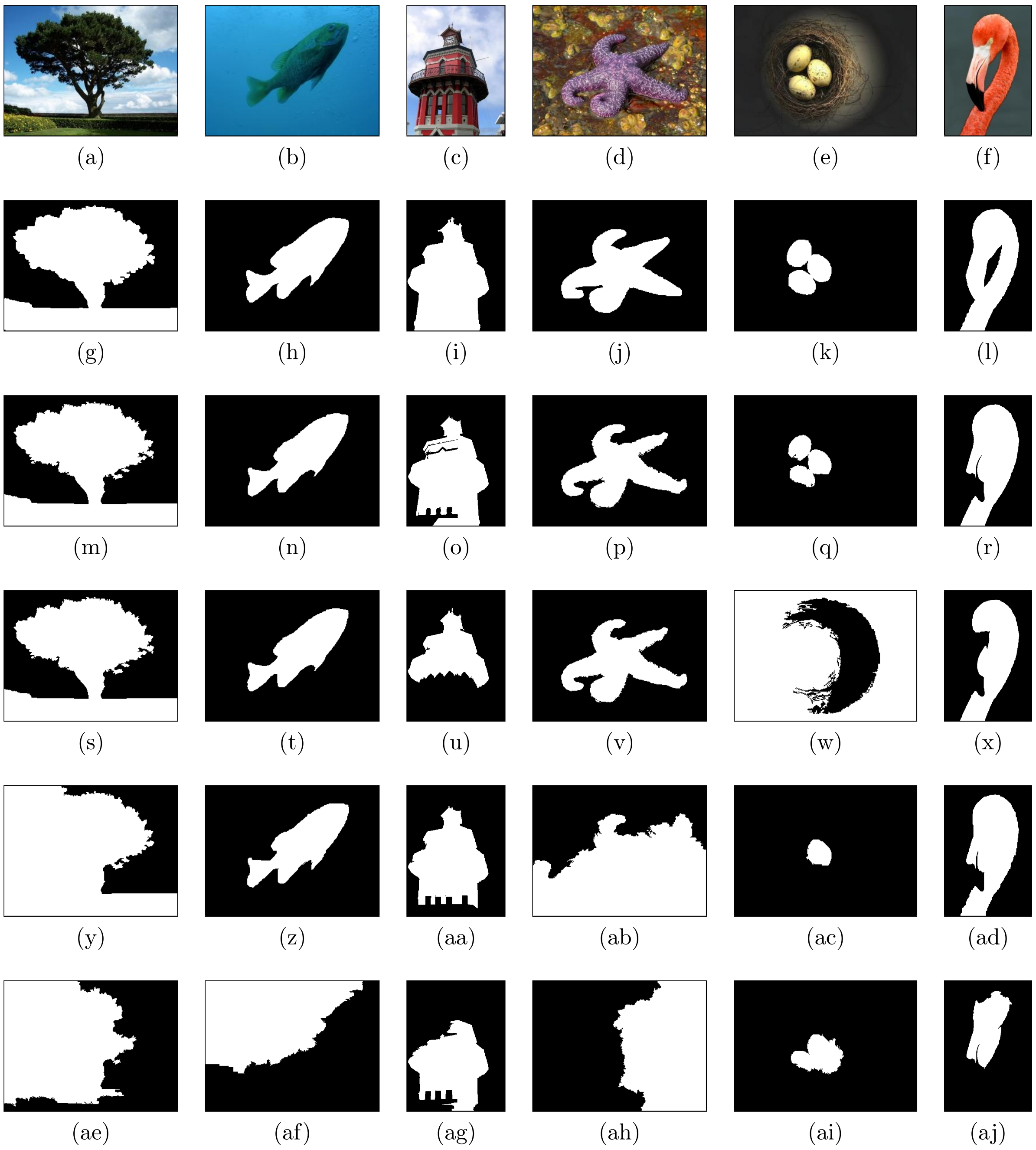


Figure 5.2: Segmentation results for Weizmann 1-Object dataset. (a)–(f) Original images, (g)–(l) Ground-truth, (m)–(r) IDT$_1$, (s)–(x) DISF (N_o = 150), (y)–(ad) IW-max and (ae)–(aj) IW-sum.

The experiments were conducted using the same sets of initial seeds for IDT$_1$, IDT$_2$, IDT$_4$, IW-max and IW-sum. To guarantee the best result from each algorithm, they are executed 20 times for each image, from which the best object segmentation is selected according to the evaluation metrics. Next, mean and standard deviation are computed from these values across all images for each dataset. Table 5.1 shows the effectiveness of object segmentation for all methods according to four different metrics (AMI, ARI,

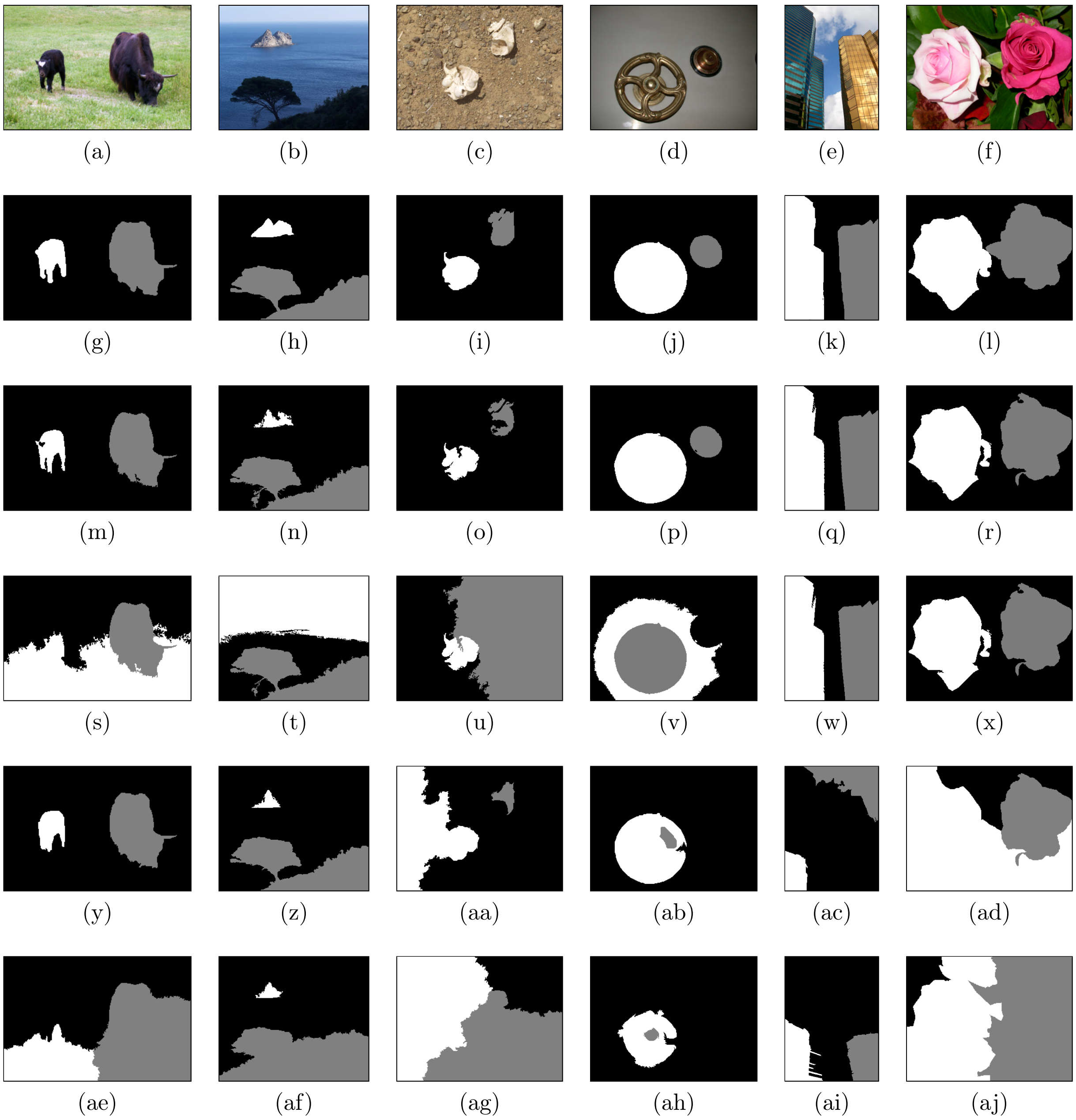


Figure 5.3: Segmentation results for Weizmann 2-Object dataset. (a)–(f) Original images, (g)–(l) Ground-truth, (m)–(r) IDT_1, (s)–(x) DISF ($N_o = 150$), (y)–(ad) IW-max and (ae)–(aj) IW-sum.

BR and CA). IDT_1 is the best approach, being worse than IDT_2 in only a single case, according to ARI for the 2-object segmentation task. IDT_1 exhibits superior results than IDT_4 suggesting that the seed recomputation based on the mean pixel is the best option in the context of image segmentation.

An important finding from the experiments showed that DISF relies heavily on the size of the initial seed set, imparting outstanding results for some images while failing for others. The results also show that random sampling suffices for step (i), and step (iv), added by the proposed approach to the ISF framework, is vital for improved object segmentation. The results raise the question of how good it would be IDT for superpixel

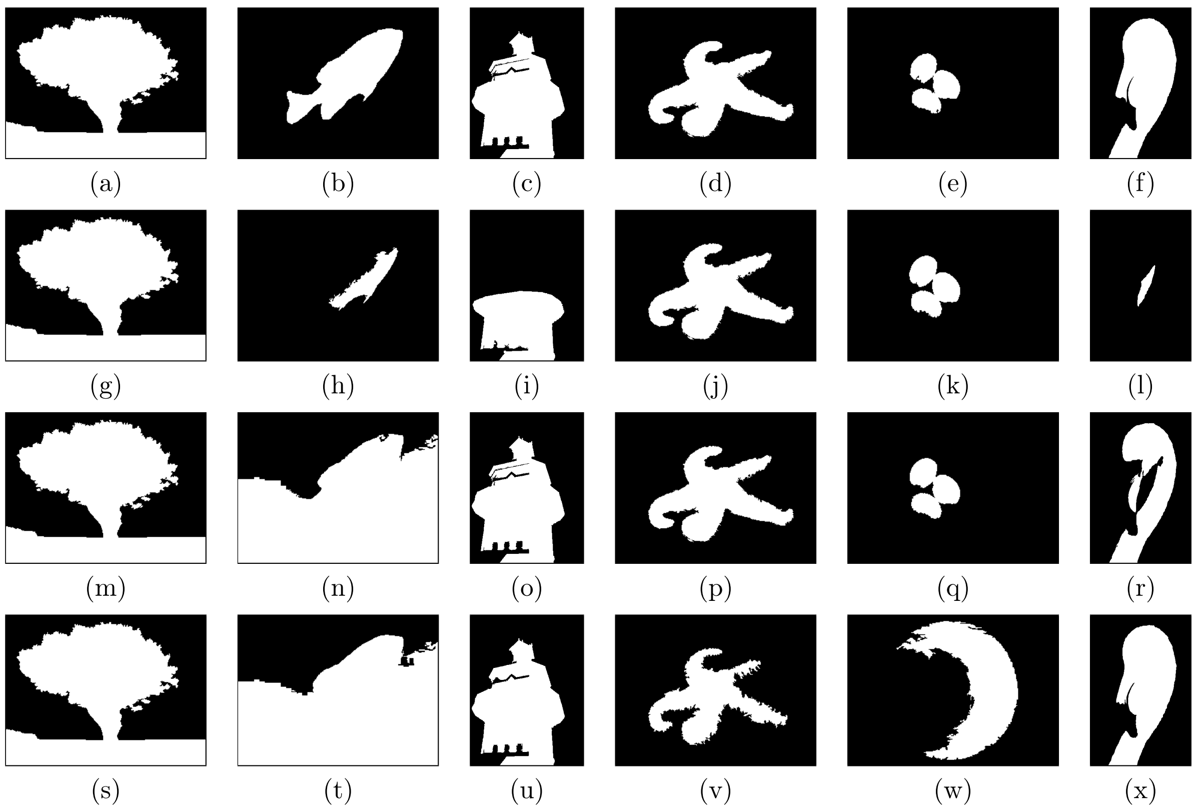


Figure 5.4: Segmentation results for Weizmann 1-Object dataset. (a)–(f) IDT$_1$, (g)–(l) IDT$_2$, (m)–(r) IDT$_3$ and (s)–(x) IDT$_4$.

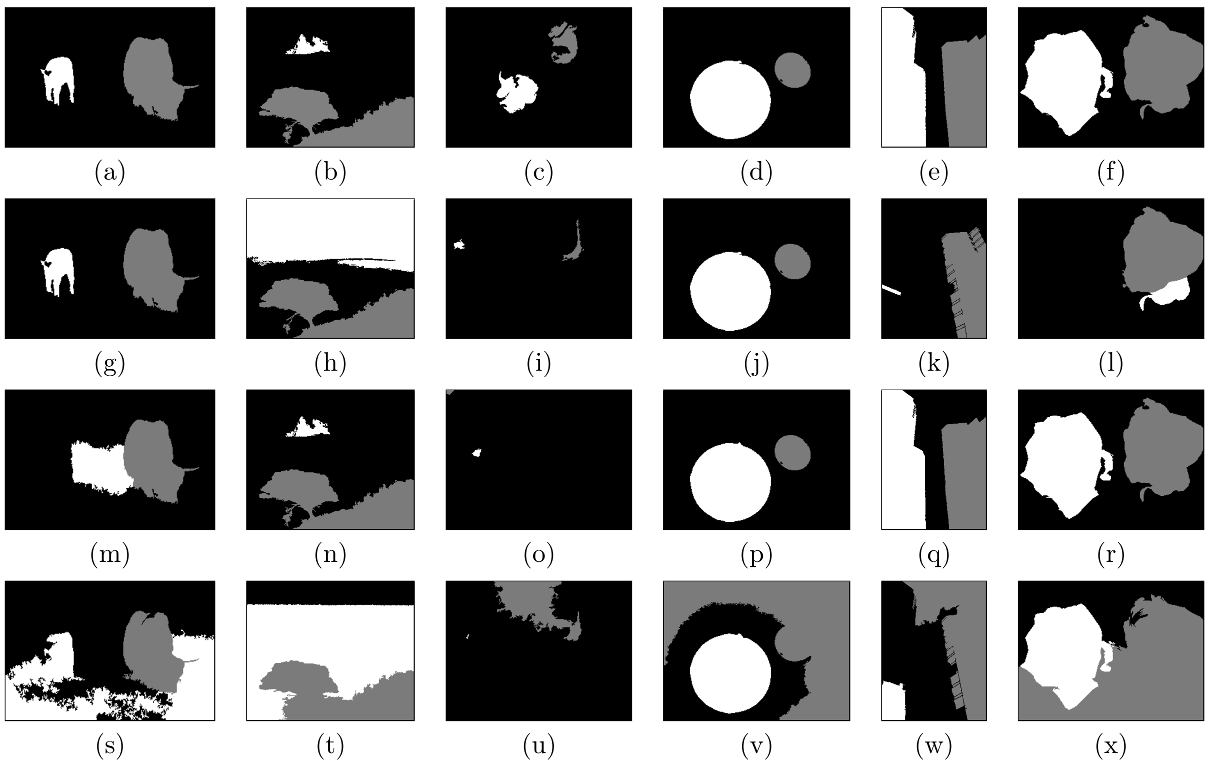
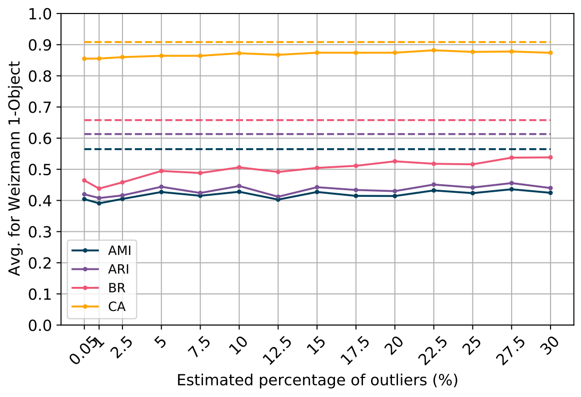
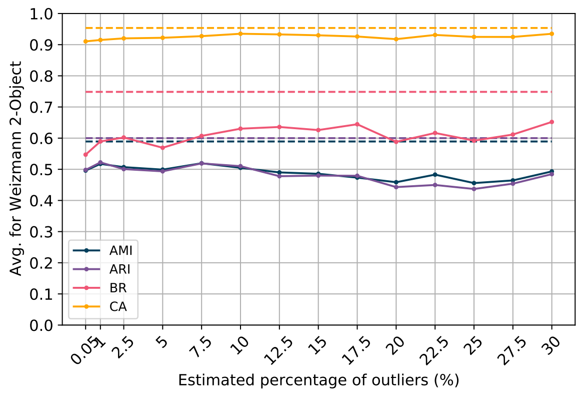


Figure 5.5: Segmentation results for Weizmann 2-Object dataset. (a)–(f) IDT$_1$, (g)–(l) IDT$_2$, (m)–(r) IDT$_3$ and (s)–(x) IDT$_4$.



(a) Weizmann 1-Object



(b) Weizmann 2-Object

Figure 5.6: Results obtained for IDT_1 with Algorithm 13 under different values of estimated seed of outliers (h) in the range $[0.05\%, 30\%]$ in each dataset for AMI, ARI, BR and CA. The score of IDT_1 with random initial seed set (see Table 5.1) is shown in dashed lines for all metrics. (a) Weizmann 1-Object dataset, (b) Weizmann 2-Object dataset.

segmentation (when the number of seeds is higher than the number of desired objects), which we will leave for future work. Figures 5.2 and 5.3 show the segmentation results for IDT_1, DISF, IW-max and IW-sum on some images of Weizmann 1-Object and 2-Object datasets, respectively, where it can be seen that the IDT_1 method achieves the best object segmentation for all image instances in contrast to its counterparts. Next, Figures 5.4 and 5.5 show the segmentation results for IDT_1, IDT_2, IDT_3 and IDT_4 on the same images as in Figures 5.2 and 5.3. From these figures, it can be seen that IDT_1 stands out from its variants as the most effective method for object delineation.

In this context, the seed set selection algorithm was also tested to determine whether this proposed methodology may positively affect the performance of IDT_1. IDT_1 originally starts with a random seed set. However, in this experiment, the algorithm will start with initial seed sets output by Algorithm 13. The seed set selection algorithm was executed with different values of h (estimated percentage of outliers) in the range of 0.05% to 30% with step size of 2.5%. Figure 5.6 exhibit the performance of this methodology expressed in four different metrics for both Weizmann 1-Object and 2-Object datasets. From the figure, we see that as we increase the estimated percentage of outliers, the algorithm's performance, assessed by BR and CA, slightly increases in both datasets. Conversely, the algorithm's performance is barely increasing and sometimes decreasing for some values of h when assessed by AMI and ARI. Nevertheless, these scores are lower than its counterpart IDT_1 with random initial seed set (shown in dashed lines). Therefore, this result suggests that combining the seed selection strategy with IDT_1 does not improve the original formulation where the seed set is selected randomly, and thus we can conclude that in order to attain the best performance in object delineation, IDT_1 with random seed selection is the most suitable configuration.

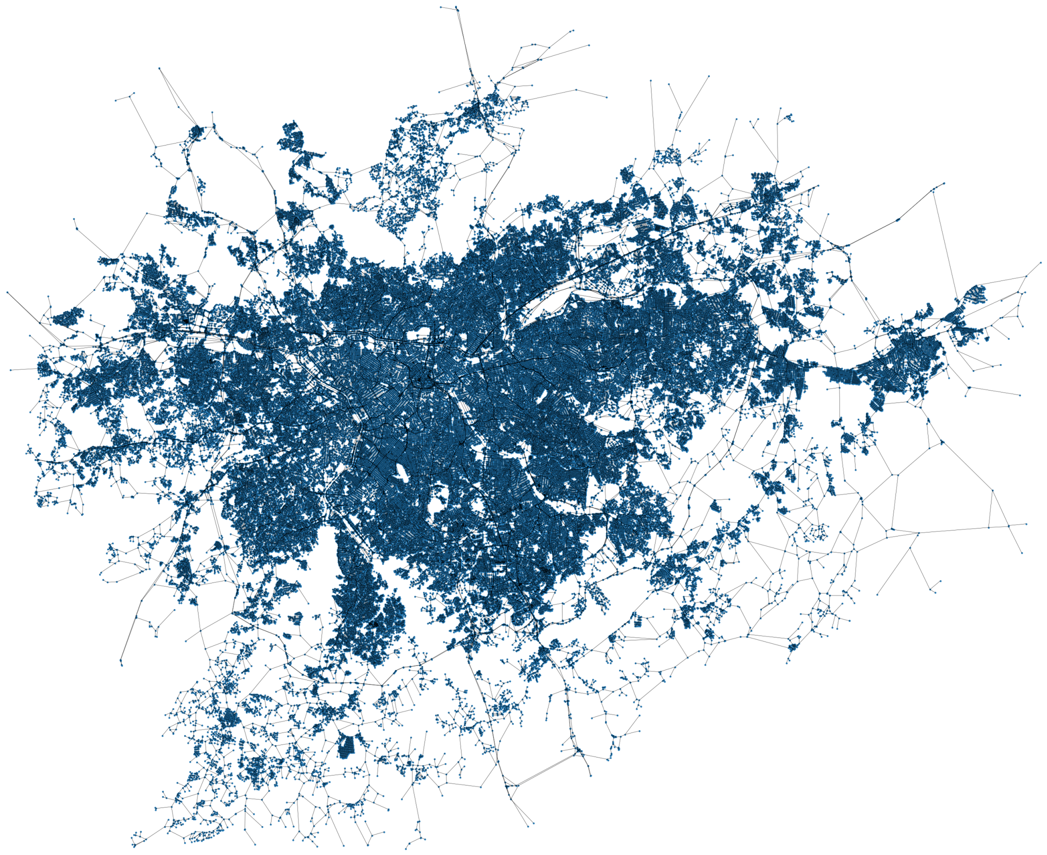


Figure 5.7: Road network of the city of São Paulo, Brazil.

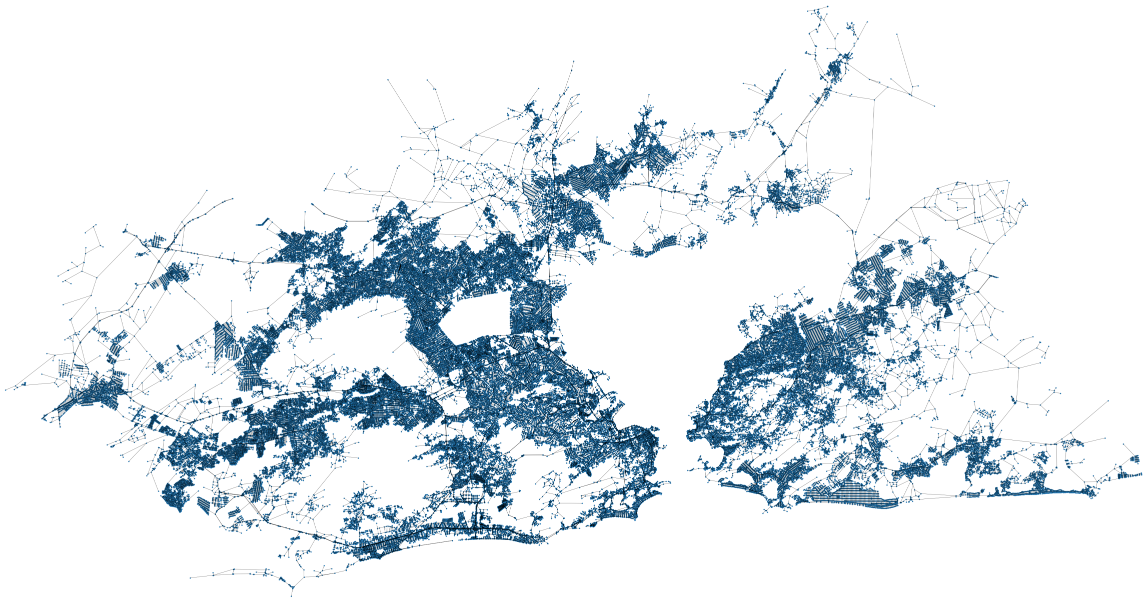
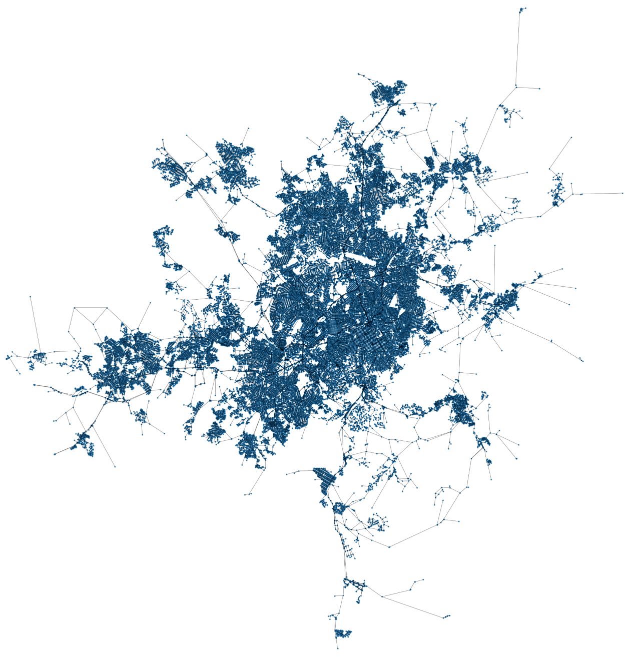


Figure 5.8: Road network of the city of Rio de Janeiro, Brazil.


(a) Belo Horizonte

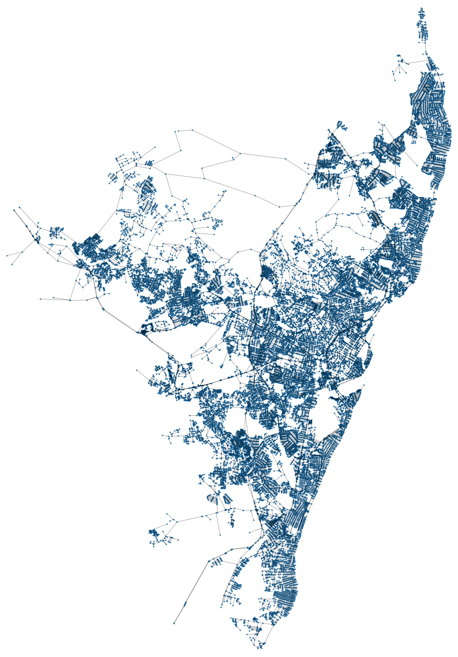
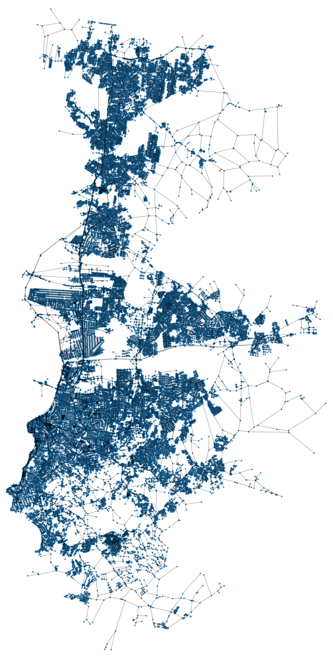

(b) Recife

Figure 5.9: Road networks of the cities of (a) Belo Horizonte and (b) Recife in Brazil.


(a) Porto Alegre

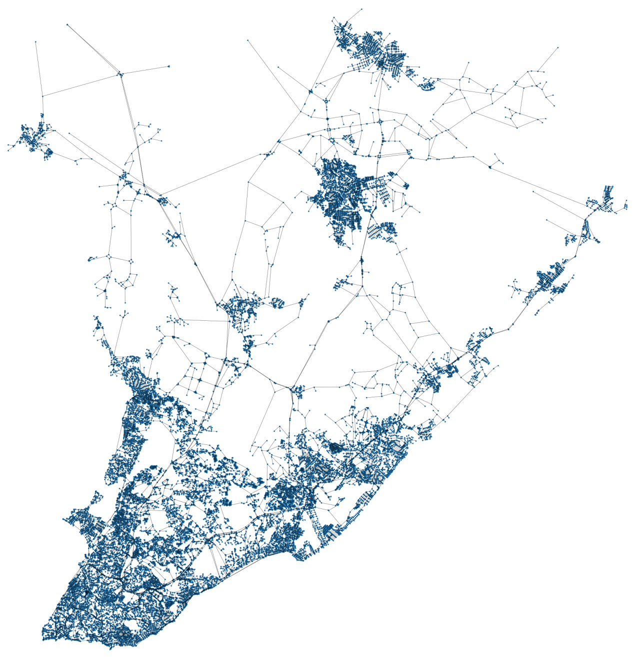
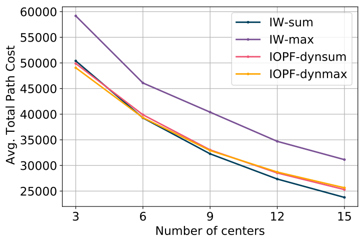

(b) Salvador

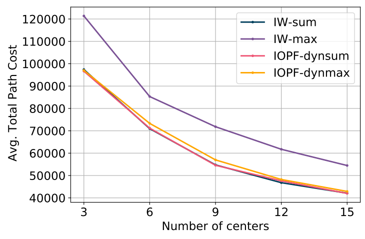
Figure 5.10: Road network of the cities of (a) Porto Alegre and (b) Salvador in Brazil.

5.2 Analysis on Road Networks

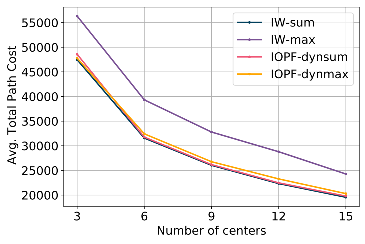
Road networks impose a new constraint on the graph's adjacency relation since it is already predefined by the road network map where edges are defined by the roads connecting two reference points. This experiment addresses the following problem of, given a road network instance, identifying suitable points for placing emergency stations, such that, each emergency station reach the point of incident in the minimum time possible. The following points should be taken into consideration to devise a solution to this problem: (i) The emergency station must be reachable from the point of incident in a short time interval (*i.e.*, the distance between these two points must be minimized) and (ii) The number of emergency stations spread across the map must be as low as possible to minimize the establishment costs.



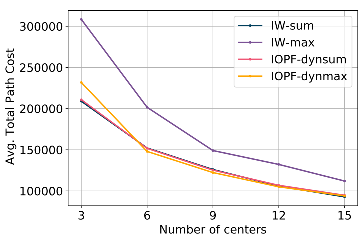
(a) Belo Horizonte



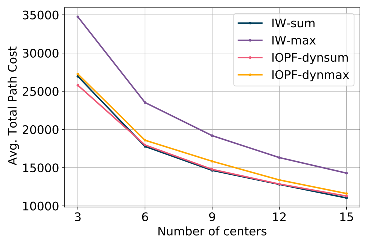
(b) Porto Alegre



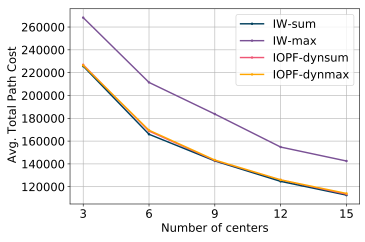
(c) Recife



(d) Rio de Janeiro



(e) Salvador



(f) São Paulo

Figure 5.11: Average total path costs for the Brazilian cities of Belo Horizonte, Porto Alegre, Recife, Rio de Janeiro, Salvador, and São Paulo for the algorithms IW-sum, IW-max, IOPF-dynsum, and IOPF-dynmax with a varying number of centers.

A road network will induce a weighted graph $G = (\mathcal{Z}, \mathcal{A})$, where the nodes are defined by a set of reference points spread across the road map which constitute the dataset $\mathcal{Z}$ and are uniquely identified by a pair of coordinates $x = (x_1, x_2)$. An emergency station must be established at one of such points. The adjacency relation $\mathcal{A}$ that defines the arcs of the graph is given by the set of pairs $(x, y) \in \mathcal{Z} \times \mathcal{Z}$, such that x and y are connected by a road. The arcs are weighted by their corresponding road lengths, which are provided beforehand for this experiment.

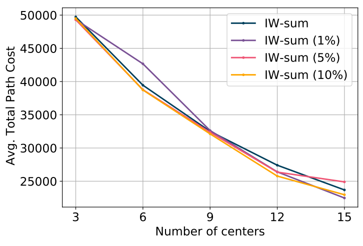
Based on the above definition, we may formulate this problem as the discovering of a set of k emergency points $c_i \in \mathcal{Z}, i \in \{1, \ldots, k\}$, such that sum of path-costs between each reference point $s \in \mathcal{Z}$ and its most closest emergency station – *i.e.*, $\sum_{s\in\mathcal{Z}} f(\pi_s)$), is minimized across all reference points for a given connectivity function f, therefore, the application of the IOPF framework to this problem is straightforward. In this context, the problem described above may be divided into two subsequent stages or subproblems. The

Table 5.2: Average total path costs for the Brazilian cities of Belo Horizonte, Porto Alegre, Recife, Rio de Janeiro, Salvador, and São Paulo for the algorithms IW-sum, IW-max, IOPF-dynsum, and IOPF-dynmax with a varying number of centers.

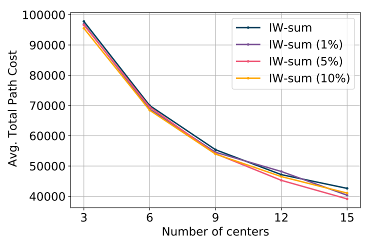
	# Centers	IW-sum	IW-max	IOPF-dynsum	IOPF-dynmax
Belo Horizonte	3	50392.84 ± 1873.05	59173.66 ± 4721.73	49879.25 ± 441.97	**49052.51 ± 1071.37**
	6	**39271.38 ± 1024.23**	46087.54 ± 2813.85	39890.12 ± 1393.49	39326.05 ± 1186.43
	9	**32266.13 ± 1047.32**	40395.69 ± 4305.03	33035.27 ± 1223.22	32909.83 ± 1297.82
	12	**27348.55 ± 1148.55**	34701.26 ± 2364.85	28533.93 ± 1179.58	28714.03 ± 1045.76
	15	**23766.05 ± 703.01**	31133.95 ± 2348.54	25295.12 ± 1083.38	25642.89 ± 957.62
Porto Alegre	3	97407.88 ± 3809.96	121376.97 ± 24728.46	**96640.18 ± 603.42**	97018.90 ± 4720.06
	6	**70961.53 ± 3290.43**	85313.21 ± 7292.24	71178.73 ± 3009.31	73312.43 ± 5177.42
	9	54750.34 ± 4342.36	71823.30 ± 7011.44	**54571.08 ± 2361.63**	56983.83 ± 4804.08
	12	**46793.18 ± 932.39**	61733.62 ± 4060.54	47611.60 ± 1677.85	48224.04 ± 1923.37
	15	42094.06 ± 1369.52	54474.27 ± 6210.77	**41996.50 ± 1673.19**	42888.86 ± 1887.78
Recife	3	**47465.21 ± 1071.42**	56321.18 ± 5700.37	48564.26 ± 2418.34	47793.52 ± 560.23
	6	**31560.96 ± 780.89**	39324.44 ± 4542.26	31759.04 ± 497.83	32420.76 ± 1992.57
	9	**26036.27 ± 971.14**	32790.09 ± 2602.29	26178.34 ± 1167.90	26766.27 ± 1484.19
	12	**22326.89 ± 754.97**	28792.67 ± 2907.76	22475.01 ± 713.84	23276.13 ± 1102.17
	15	**19550.77 ± 430.20**	24273.99 ± 1740.91	19781.37 ± 567.09	20289.22 ± 1009.53
Rio de Janeiro	3	**208968.97 ± 7704.59**	308388.16 ± 83156.19	210828.25 ± 12300.04	231768.06 ± 53873.91
	6	152151.05 ± 8669.53	201568.28 ± 61683.35	151757.59 ± 7249.10	**147985.06 ± 4118.53**
	9	126027.29 ± 6118.03	149069.25 ± 14838.43	125305.51 ± 5534.58	**122329.64 ± 5945.00**
	12	105914.04 ± 3594.79	132210.23 ± 9104.88	106830.88 ± 3795.28	**105052.38 ± 4250.77**
	15	**92811.58 ± 4231.72**	112041.85 ± 8576.53	94756.97 ± 4602.93	94047.29 ± 4831.71
Salvador	3	26953.18 ± 2587.67	34741.45 ± 4769.68	**25786.45 ± 1544.62**	27255.29 ± 3135.52
	6	**17782.28 ± 648.34**	23527.57 ± 4215.90	17970.04 ± 828.34	18591.79 ± 612.93
	9	**14645.86 ± 436.67**	19189.59 ± 3306.66	14777.55 ± 646.55	15838.72 ± 1589.64
	12	**12812.38 ± 811.51**	16319.22 ± 1732.45	12857.59 ± 847.78	13392.92 ± 869.91
	15	**11037.27 ± 764.02**	14287.47 ± 1239.53	11280.82 ± 853.77	11615.92 ± 1083.61
São Paulo	3	**225694.20 ± 2862.17**	268244.34 ± 16638.99	226909.29 ± 6658.03	226372.56 ± 684.42
	6	**165991.84 ± 3441.19**	211433.73 ± 16557.69	168775.45 ± 4092.53	169531.03 ± 5047.84
	9	**142613.33 ± 2214.72**	183700.73 ± 15373.26	143098.41 ± 1907.44	143415.22 ± 3057.78
	12	**124687.04 ± 2041.82**	154775.17 ± 8153.19	126028.38 ± 2688.92	126033.73 ± 3403.75
	15	**112642.07 ± 1868.52**	142541.69 ± 11614.11	113465.35 ± 2149.23	114127.19 ± 2769.51

first stage entails discovering the set of k emergency points through the IOPF framework, which can be carried out under different combinations of connectivity functions and arc-weight estimation strategies. Next, the second stage determines the sum of path-costs between each of the emergency stations discovered during the first stage and its most closest points, which is obtained through a single execution of the OPF algorithm with f_{sum} using the emergency station points as seeds. The ideal number of emergency stations is identified after repeating the experiment with a sequence of increasing values. The ideal number of stations is selected at the point where the reduction of $\sum_{s\in\mathcal{Z}} f(\pi_s))$ does not compensate the placing costs of establishing an additional emergency station.

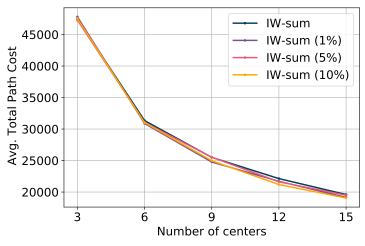
In this experiment, our objective is to determine the IOPF configuration that suits better the problem described above. Therefore, we compare four versions of IOPF, where



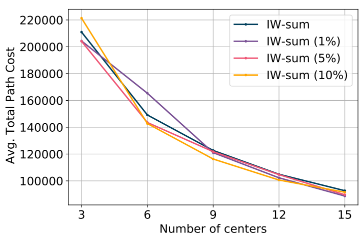
(a) Belo Horizonte



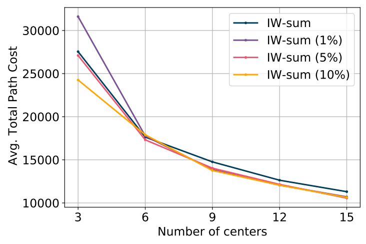
(b) Porto Alegre



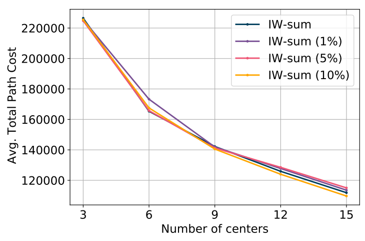
(c) Recife



(d) Rio de Janeiro



(e) Salvador



(f) São Paulo

Figure 5.12: Average total path costs for the Brazilian cities of Belo Horizonte, Porto Alegre, Recife, Rio de Janeiro, Salvador, and São Paulo for IW-sum with random seed selection and IW-sum using the seed selection algorithm with estimated percentage of outliers (h) of 1%, 5%, and 10% with a varying number of centers.

each of them presents a variation in their configurations on either the connectivity function or the arc-weight estimation strategy. IW-sum and IW-max determine the cost of a path using the f_{sum} and f_{max} connectivity functions, respectively, for fixed arc weights $\|x - y\|$. On the other hand, IOPF-dynsum and IOPF-dynmax determine the cost of a path using the f_{sum} and f_{max} connectivity functions for dynamic arc-weight estimation.

The road networks for this experiments are taken from [45] (available at `https://figshare.com/articles/dataset/Urban_Road_Network_Data/2061897`). In [76], a similar experiment was conducted with the road networks corresponding to the Indian cities of Mumbai, Hyderabad, Chennai, Bengaluru, Calcuta, and Delhi. The IW algorithm was compared to k-means and greedy k-center exhibiting a better performance than its counterparts. This experiment uses the road networks corresponding to the Brazilian cities of São Paulo, Rio de Janeiro, Belo Horizonte, Recife, Porto Alegre and Salvador. IOPF-based algorithms assume that the graph on which are executed is connected. Therefore, we need to make sure that the graph induced by a road network is connected. To simplify this task, we select the largest connected component in each road network as the graph induced by the road network. Figures 5.7, 5.8, 5.9, and 5.10 show the graph induced by the road networks of the Brazilian cities where blue dots represent the nodes or reference points, while black lines linking pairs of reference points represent the edges or roads.

The experiments were carried out using the same sets of initial seeds for IW-sum, IW-max, IOPF-dynsum and IOPF-dynmax. For each city's road network, each method is executed thirty times for a varying number of centers, then the sum of path-costs across all nodes $\sum_{s \in \mathcal{Z}} f(\pi_s)$ is averaged across all executions to assess its effectiveness. Table 5.2 shows the comparison of the average values of the total path costs $\sum_{s \in \mathcal{Z}} f(\pi_s)$ with $3, 6, 9, 12$ and 15 centers for each city's road network. IW-sum appears to be the

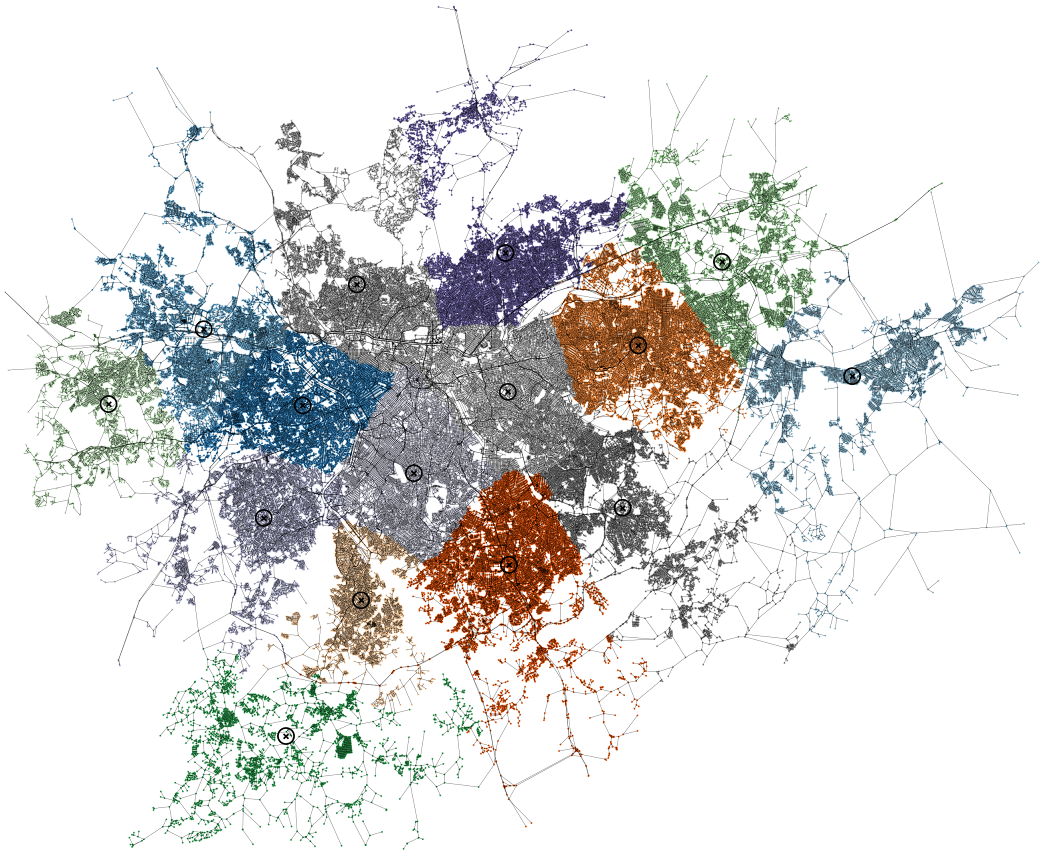


Figure 5.13: Road network of the city of São Paulo, Brazil.

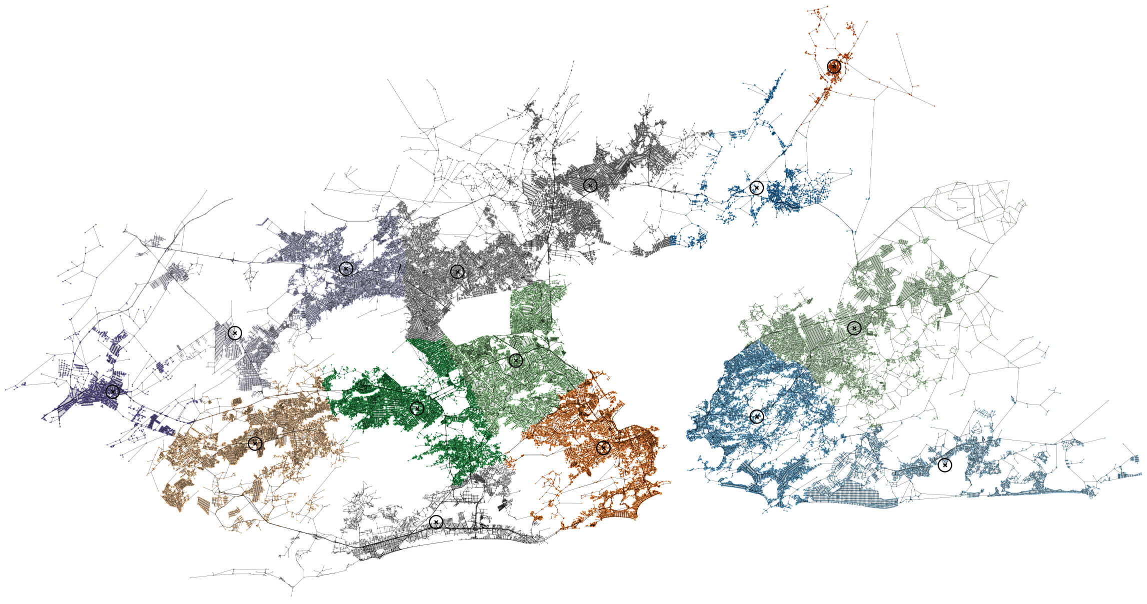
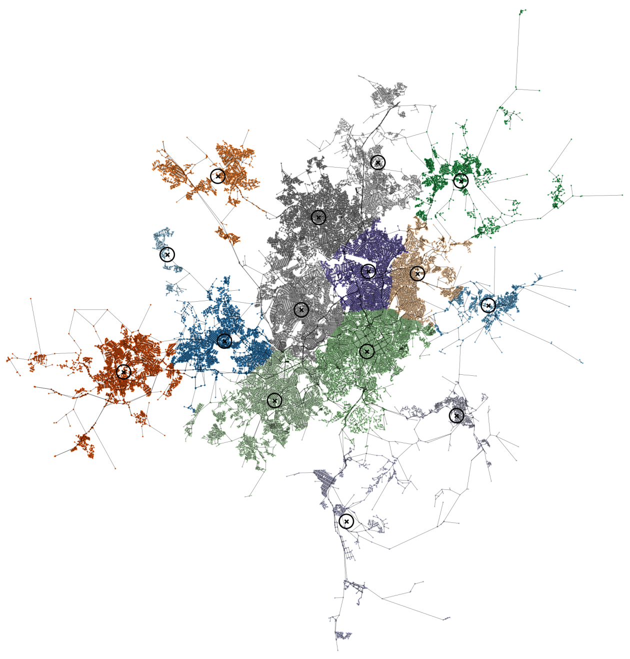


Figure 5.14: Road network of the city of Rio de Janeiro, Brazil.


(a) Belo Horizonte

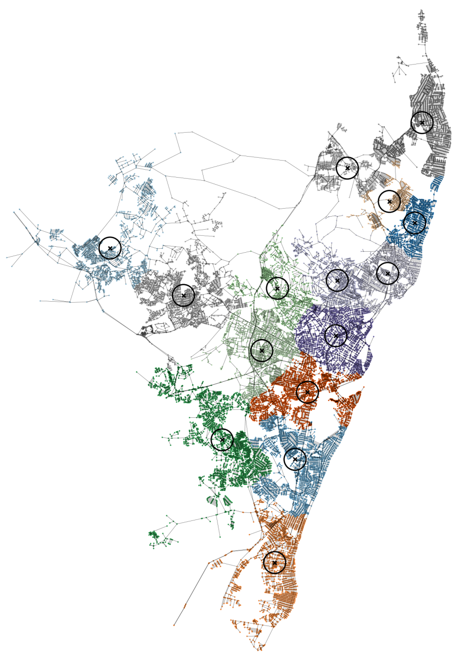
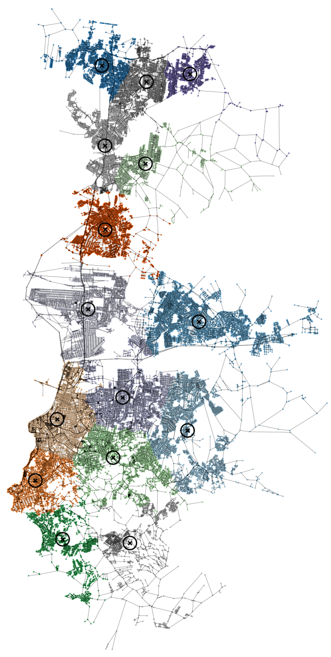

(b) Recife

Figure 5.15: Road networks of the cities of (a) Belo Horizonte and (b) Recife in Brazil.


(a) Porto Alegre

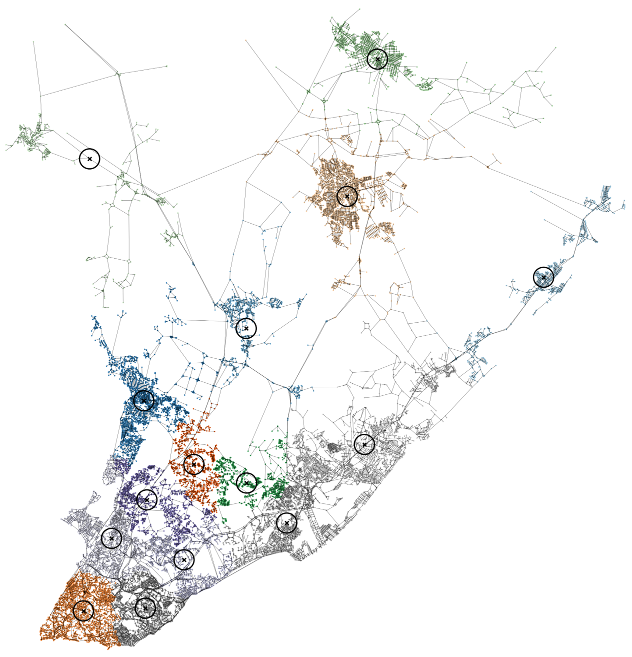

(b) Salvador

Figure 5.16: Road network of the cities of (a) Porto Alegre and (b) Salvador in Brazil.

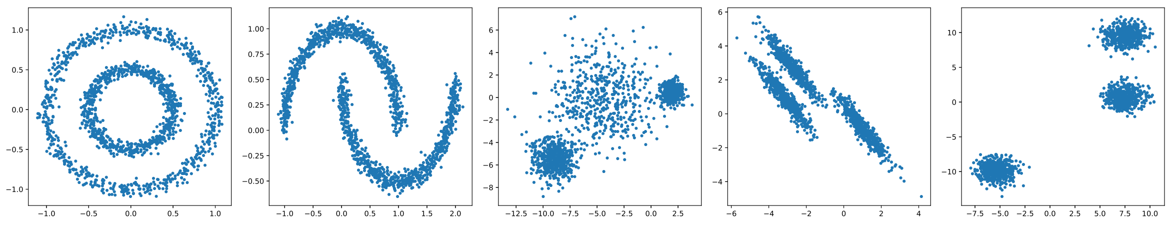
most suitable IOPF-based method in most of the cases achieving a lower value than its counterparts, and being worse than IOPF-dynsum and IOPF-dynmax in only a few cases where the difference in values is not significant. Furthermore, it can also be observed that both IOPF-dynsum and IOPF-dynmax obtain average values which are very similar to those obtained by IW-sum in contrast to IW-max, which obtain the largest average values. Figure 5.11 illustrate this comparison through a line chart for each city's road network. Each line chart shows the average total path cost across an increasing number of centers for each method, where IW-sum stands out as the method that minimizes the total path cost in most of the settings. On the other hand, in order to identify whether the seed selection algorithm may improve the effectiveness of the IW-sum method, we executed the seed selection algorithm for different values of h, namely 5%, 10% and 15%. From Figure 5.12, we can see that some improvement is achieved when coupling the seed selection algorithm with the IW-sum method. This becomes more noticeable as the number of centers increases. We believe that this result can be further improved by incorporating local density information to the seed selection algorithm. This idea stems from the fact that seeds should be chosen from dense regions of points, however, the seed selection algorithm only takes into account distances along a path measured by a given connectivity function. Nevertheless, density computation for large datasets may become computationally expensive, impairing performance. Accordingly, research should be conducted to introduce density information into the seed selection procedure without compromising performance.

Figures 5.13, 5.14, 5.15 and 5.16 show the clustering result of IW-sum with fifteen centers for each city's network, where each cluster is colored with a different color and the centers (*i.e.*, emergency points) are marked with a cross surrounded by a circle.

5.3 Experiments on Synthetic Datasets

In order to ascertain the performance and robustness of the IOPF framework in a broader variety of datasets, we evaluate several IOPF-based methods on synthetic datasets that exhibit a wide spectrum of shapes and distributions. In the first part of this experiment, we use five synthetic datasets generated with *sklearn*, a Python library that implements most of the state-of-the-art machine learning algorithms and techniques. These five datasets are shown in Figure 5.17, where it can be seen that the *noisy circles* and *noisy moons* datasets comprise two groups, while the *varied*, *aniso* and *blobs* datasets comprise three groups. All these datasets consist of 1500 samples each and exhibit a diversity of shapes and distributions, which will allow us to assess the framework's robustness in these case scenarios.

In this kind of datasets, we do not possess enough information about the underlying relationship among the samples to establish a suitable graph topology, conversely to what happens in Sections 5.1 and 5.2. Therefore, a strategy to build suitable graph topologies in the general case is still a topic under investigation. In Chapter 4, we introduced two examples of graph topologies. Let $\mathcal{Z}$ be a dataset such that each sample $s \in \mathcal{Z}$ is represented in the feature space by a feature vector $v(s) \in \mathbb{R}^n$. The adjacency relation $\mathcal{A} \subseteq \mathcal{Z} \times \mathcal{Z}$ may be defined in such a way that the induced graph $G = (\mathcal{Z}, \mathcal{A})$ can be established either as a complete or k-nearest neighbor graph. If G is defined as a complete



(a) Noisy circles (b) Noisy moons (c) Varied (d) Aniso (e) Blobs

Figure 5.17: Synthetic datasets created with sklearn.

graph, IOPF with fixed arc weights and f_{sum} connectivity function (*i.e.*, IW-sum) is expected to behave in a similar way as k-means. In this setting, the edge connecting any seed to any of the remaining samples is selected as the path with the minimum cost over any other path connecting such pair of samples in the graph. This fact can be explained by means of the triangle inequality extended to polygons, and therefore, its behavior matches that of k-means, which considers direct distances between centroids and samples to form the clusters.

We conducted experiments with four configurations of IOPF to ascertain its effectiveness under a complete graph topology. IW-sum and IW-max use fixed arc weights with f_{sum} and f_{max} connectivity functions, respectively. On the other hand, IOPF-dynsum and IOPF-dynmax use dynamic arc weight estimation with f_{sum} and f_{max} connectivity functions, respectively. We compare the aforementioned IOPF-based methods with k-means on the task of clustering the synthetic datasets. Figure 5.18 shows the clustering results for k-means, IW-sum, IW-max, IOPF-dynsum and IOPF-dynmax. For each dataset, the same initial seeds are selected for all methods, which are marked with crosses ($\times$), while the final medoids are marked with triangles (▲). From the results in Figure 5.18, we can see that the final clusterings achieved by k-means, IW-sum, IOPF-dynsum and IOPF-dynmax are similar for all datasets, however, the position of the final medoids differ, and this fact is more evident for the noisy moons dataset in the second row. The k-means, IW-sum, IOPF-dynsum and IOPF-dynmax algorithms fail to separate the clusters for the noisy circles, noisy moons, varied and aniso datasets, which are represented in the first four rows of Figure 5.18. These algorithms only succeed when separating the groups in the blobs dataset, where the clusters are well-defined, well-separated, and exhibit spherical shapes. On the other hand, the IW-max method successfully separates the groups for all synthetic datasets, albeit the final result of the clustering relies heavily on the selection of the initial seeds. IW-max is very sensitive to isolated points or outliers, penalizes spatial gaps in the dataset, and works independently of the clusters' shape. Figure 5.19 shows the clustering results using IW-max, where it fails to separate the groups due to bad initial seeds and spatial gaps in the dataset. Therefore, in an effort to circumvent these cases, we employ the seed selection algorithm (see Algorithm 13) with estimated percentage of outliers $h = 5\%$, as shown in Figure 5.20. It can be seen from the figure that seeds are chosen from dense regions of samples with similar inter-seed distance. In some cases, the algorithm starts with one seed from each cluster (which is the ideal scenario) succeeding in separating the groups. Accordingly, combining IW-max on a complete graph topology with the seed selection algorithm seems a good approach to address the clustering problem in the general case.

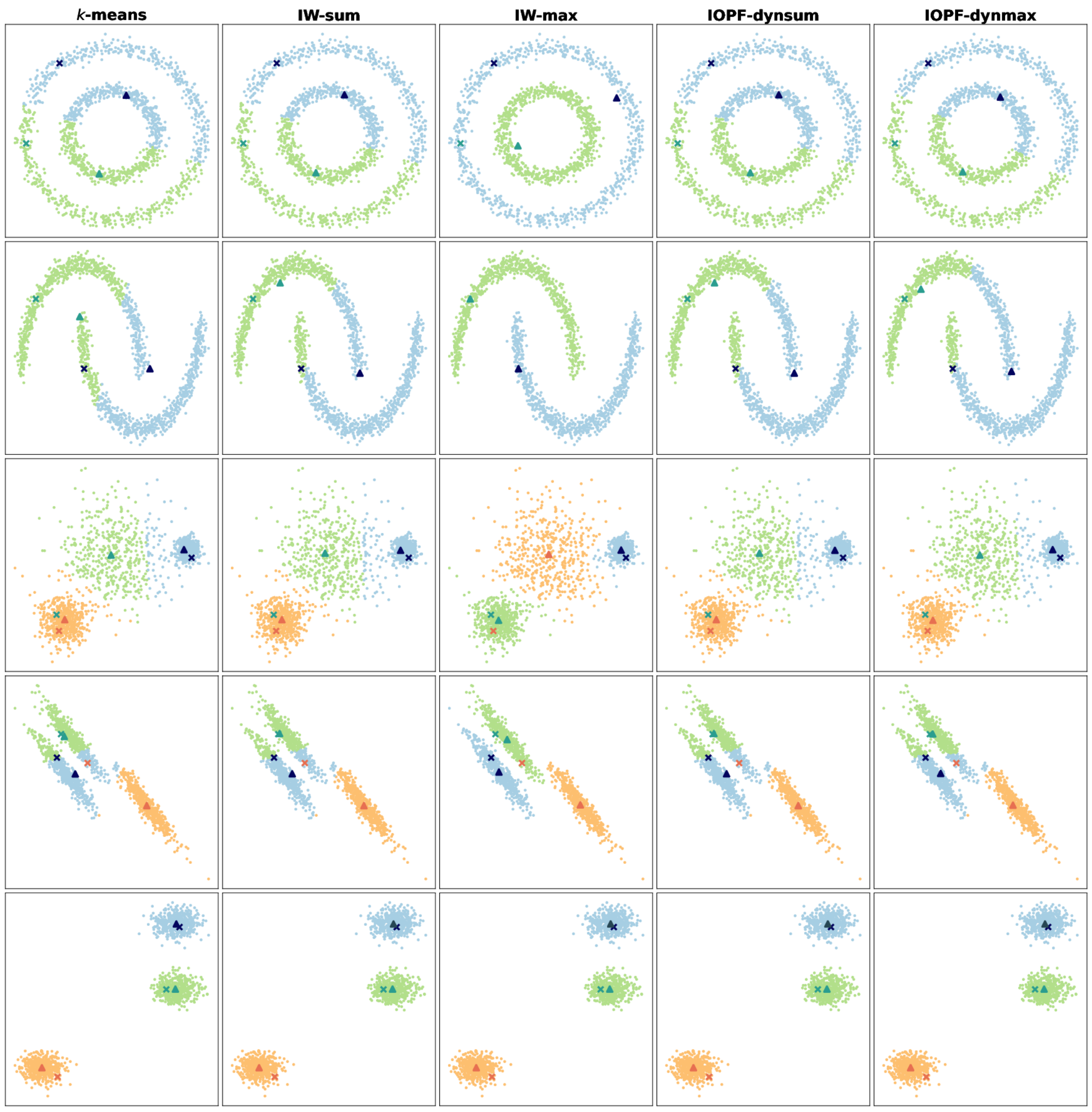


Figure 5.18: Clustering results on the synthetic datasets for k-means, IW-sum, IW-max, IOPF-dynsum and IOPF-dynmax on a complete graph topology. The initial seeds are marked with crosses ($\times$), while the final medoids are marked with triangles ($\blacktriangle$). In cases where the initial seeds are equivalent to the final medoids, only the final medoids are marked.

Alternatively, we may as well define G as a k-nearest neighbor graph, where each sample $s \in \mathcal{Z}$ is linked through an edge to its k closest neighbors for a fixed k. This construction is straightforward, however, we need to make sure that two conditions are met. If $(u, v) \in \mathcal{A}$, then $(v, u) \in \mathcal{A}$, and G must be connected. If G is not connected, the edges of the minimum spanning tree containing all samples in $\mathcal{Z}$ are added to G in increasing order of weight, joining the connected components, until there is a single connected com-

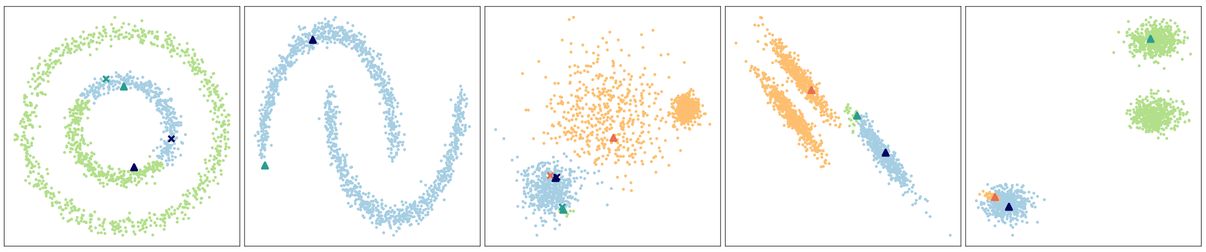


Figure 5.19: Clustering results on the synthetic datasets for IW-max where it fails at separating the groups due to bad initial seeds and spatial gaps in the dataset.

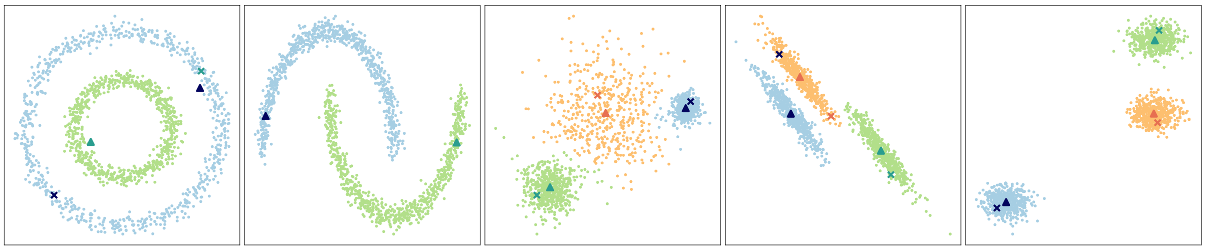


Figure 5.20: Clustering results on the synthetic datasets for IW-max where initial seeds are selected through the seed selection algorithm with estimated percentage of outliers (h) of 5%.

ponent in G. We conducted experiments using a k-nearest neighbor graph topology with $k = 15$ where the steps described above were applied. Figure 5.21 shows the results of running k-means, IW-sum, IW-max, IOPF-dynsum and IOPF-dynmax on the synthetic datasets. From the figure, it can be seen that IW-sum now is able to successfully separate the groups for the noisy circles, noisy moons, aniso and varied datasets. Therefore, imposing restrictions in the graph topology leads to improvements in the clustering capabilities of IW-sum. On the other hand, IOPF-dynsum and IOPF-dynmax still fail at separating the groups for the noisy circles, noisy moons, varied and aniso datasets. IW-max still performs well under this restriction on the graph topology, obtaining a similar result to the case of a complete graph.

To ascertain the framework's effectiveness and robustness against other state-of-the-art clustering algorithms, we compared IW-max using a complete graph topology and the seed selection algorithm where $h = 5\%$ with five popular clustering algorithms: (i) mean shift, (ii) spectral clustering, (iii) DBSCAN, (iv) Gaussian mixture, and (v) agglomerative clustering. Mean shift [23] is a nonparametric iterative mode-seeking clustering algorithm which works by means of maximizing the kernel density estimate to locate the modes in the data. It is widely used in pattern recognition and computer vision. Spectral clustering [80] is a graph-theoretic technique that computes the Laplacian of the graph and exploits its properties to partition the data regardless of the clusters' shapes. DBSCAN [28] is a density-based clustering technique that is capable of discovering arbitrary shape clusters and effectively handling noise and outliers during the clustering process. Gaussian mixture [67] is a parametric probability density function expressed as a weighted sum of a finite number of Gaussian distributions and determine its parameters through expectation maximization (EM). Agglomerative clustering [64] is a hierarchical

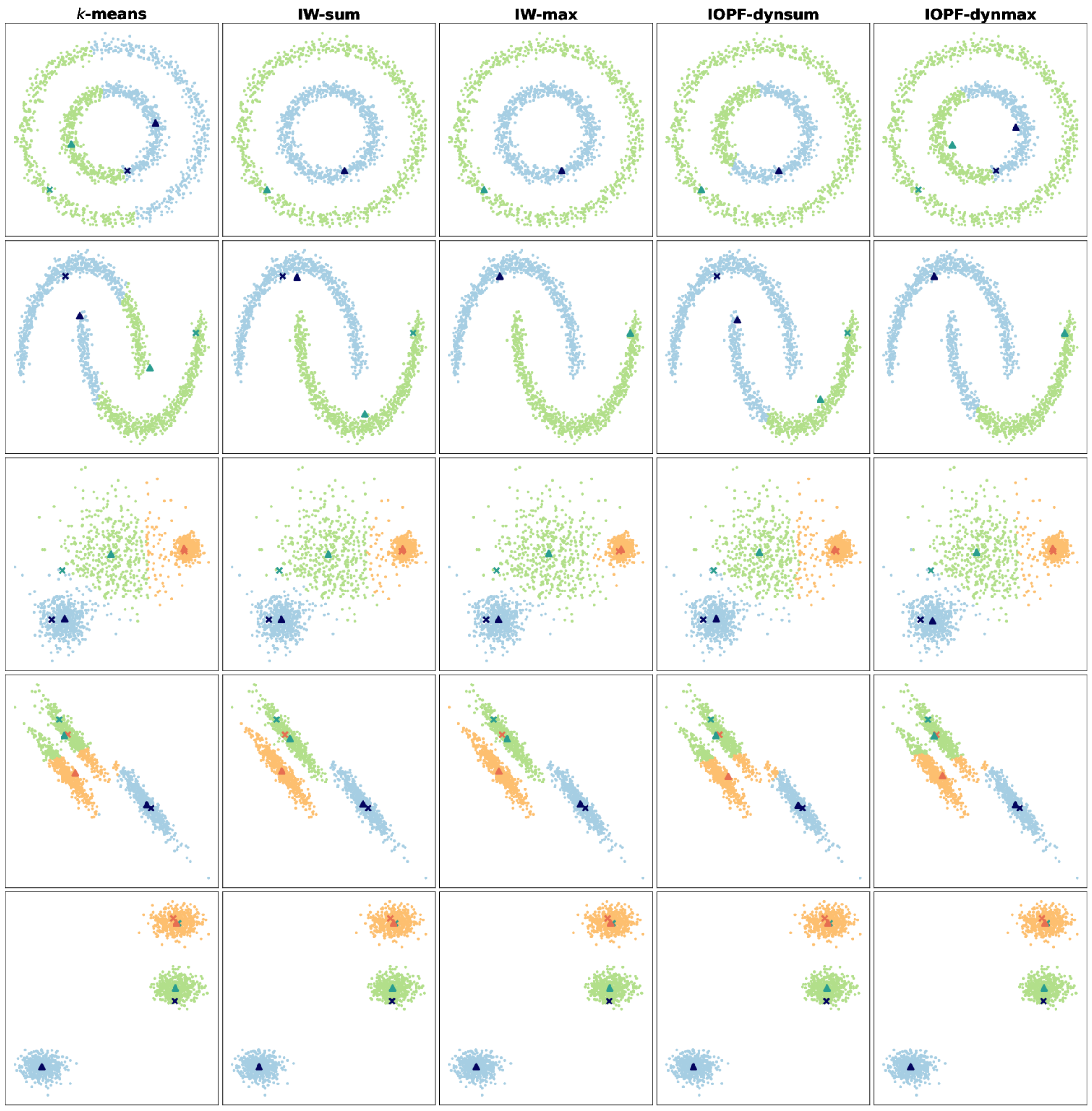


Figure 5.21: Clustering results on the synthetic datasets for k-means, IW-sum, IW-max, IOPF-dynsum and IOPF-dynmax on a k-nearest neighbor graph topology with $k = 15$.

clustering technique that follows a bottom-up approach to progressively partition the data outputting a hierarchical structure.

Figure 5.22 shows the result of this comparison on the synthetic datasets. It can be seen that, in contrast to its counterparts, IW-max successfully separates the groups for all synthetic datasets. The spectral clustering algorithm is also able to separate the groups for most of the datasets, however, it fails for the aniso dataset, not being able to fully separate the blue and green clusters. Similarly, the agglomerative clustering algorithm only fails to identify the correct groups for the aniso dataset, while correctly separating the groups in the rest of the cases. As expected, the Gaussian mixture algorithm correcty identifies elliptical shape clusters, while failing for the noisy circles and noisy moons datasets. DBSCAN works effectively when clusters are dense and well-separated by low density regions, however, it is unsuccessful when clusters' boundaries are too close as in the cases of the varied and aniso datasets. Mean shift, on the other hand, is not able to identify the clusters in the noisy circles, noisy moons and aniso datasets. Finally, all algorithms identify effectively the groups in the blob dataset, where the clusters exhibit spherical shape and are well separated in the feature space.

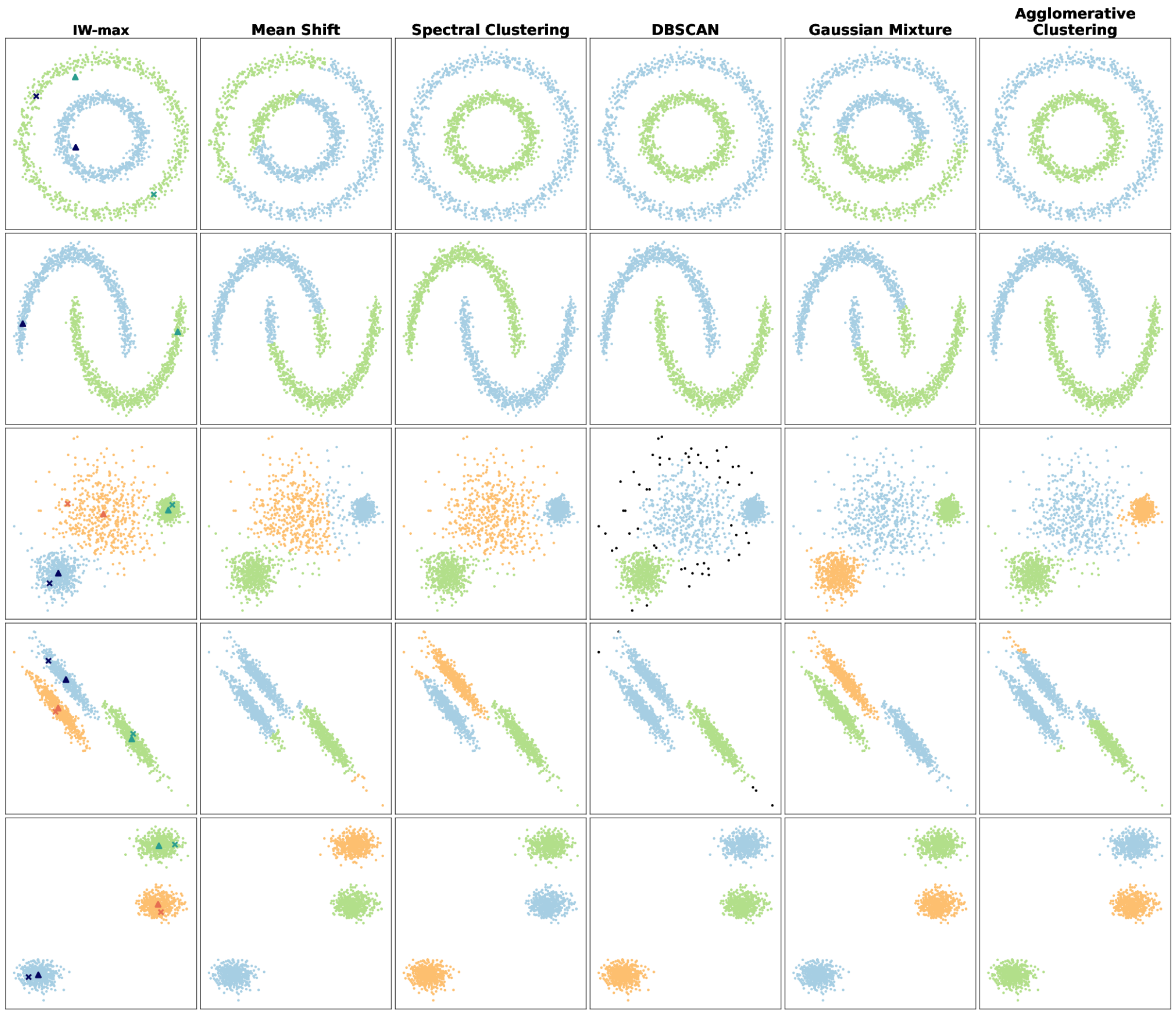
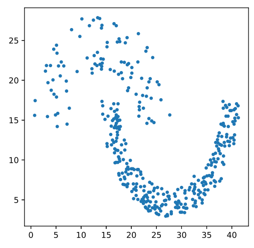


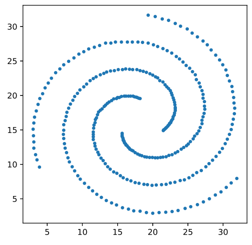
Figure 5.22: Clustering results on the synthetic datasets using IW-sum with a complete graph and the seed selection algorithm, where $h = 5\%$, against mean shift, spectral clustering, DBSCAN, Gaussian mixture and agglomerative clustering.

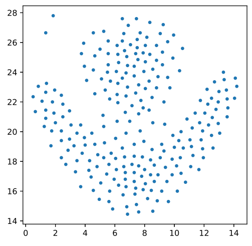
To further assess the framework's performance, we employed seven additional datasets collected from [36]. Each dataset presents some challenges due to their inherent structure and distribution. Figure 5.23 shows the datasets employed in this experiment. The *jain* and *flame* datasets comprise two clusters each, where the clusters' density in the jain dataset is not uniform. The *spiral* datasets comprise three clusters shaped as curves. The *r15*, *s1* and *s2* datasets consist of fifteen clusters each, with different degrees of separation. The *unbalance* dataset consists of eight clusters where the three groups on the left are considerably denser than the five clusters on the right, hence the clusters are unbalanced. Table 5.3 summarizes the datasets' information. We compare IW-max on a complete graph with the seed selection algorithm against k-means, mean shift, spectral clustering, DBSCAN, Gaussian mixture, and agglomerative clustering. Figure 5.24 shows the results of the clustering experiments. Different values of estimated percentage of outliers (h) were used for the seed selection algorithm based on the data distribution and density. For the jain, spiral, flame and r15 datasets, we used $h = 10\%$. For the $s1$ and $s2$ datasets, we used $h = 2\%$ and lastly, for the unbalance dataset, we used $h = 1\%$. From the figure, it can be observed that IW-sum successfully separates the clusters in all datasets. Moroever, through the seed selection algorithm, we are able to locate a seed within each cluster for each dataset, therefore, it serves as a key tool to achieve robust clustering results.

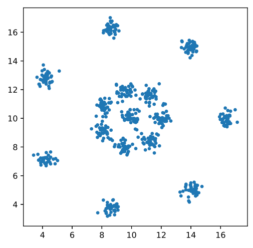
Table 5.3: Number of samples and clusters for each synthetic dataset.

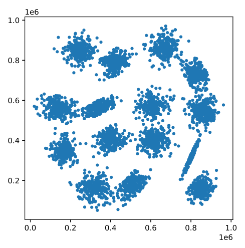
Dataset	**# of Samples**	**# of Clusters**
Jain	373	2
Spiral	312	3
Flame	240	2
R15	600	15
S1	5000	15
S2	5000	15
Unbalance	6500	8

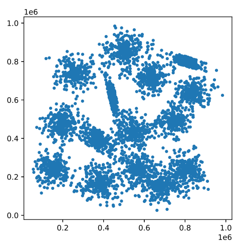

(a) Jain


(b) Spiral


(c) Flame


(d) R15


(e) S1


(f) S2

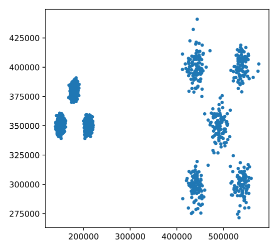

(g) Unbalance

Figure 5.23: Synthetic datasets collected from [36].

The clusters in the jain dataset are only correctly identified by IW-max, while its counterparts are not able to effectively discover the groups. The clusters in the spiral dataset are only correctly detected by IW-max, DBSCAN and agglomerative clustering, while for the flame dataset, the IW-max and DBSCAN are the ones that stand out from the rest. For the r15 dataset, IW-max, k-means, mean shift, and Gaussian mixture are the methods that accurately identify the groups. In the case of the s1 and s2 datasets,

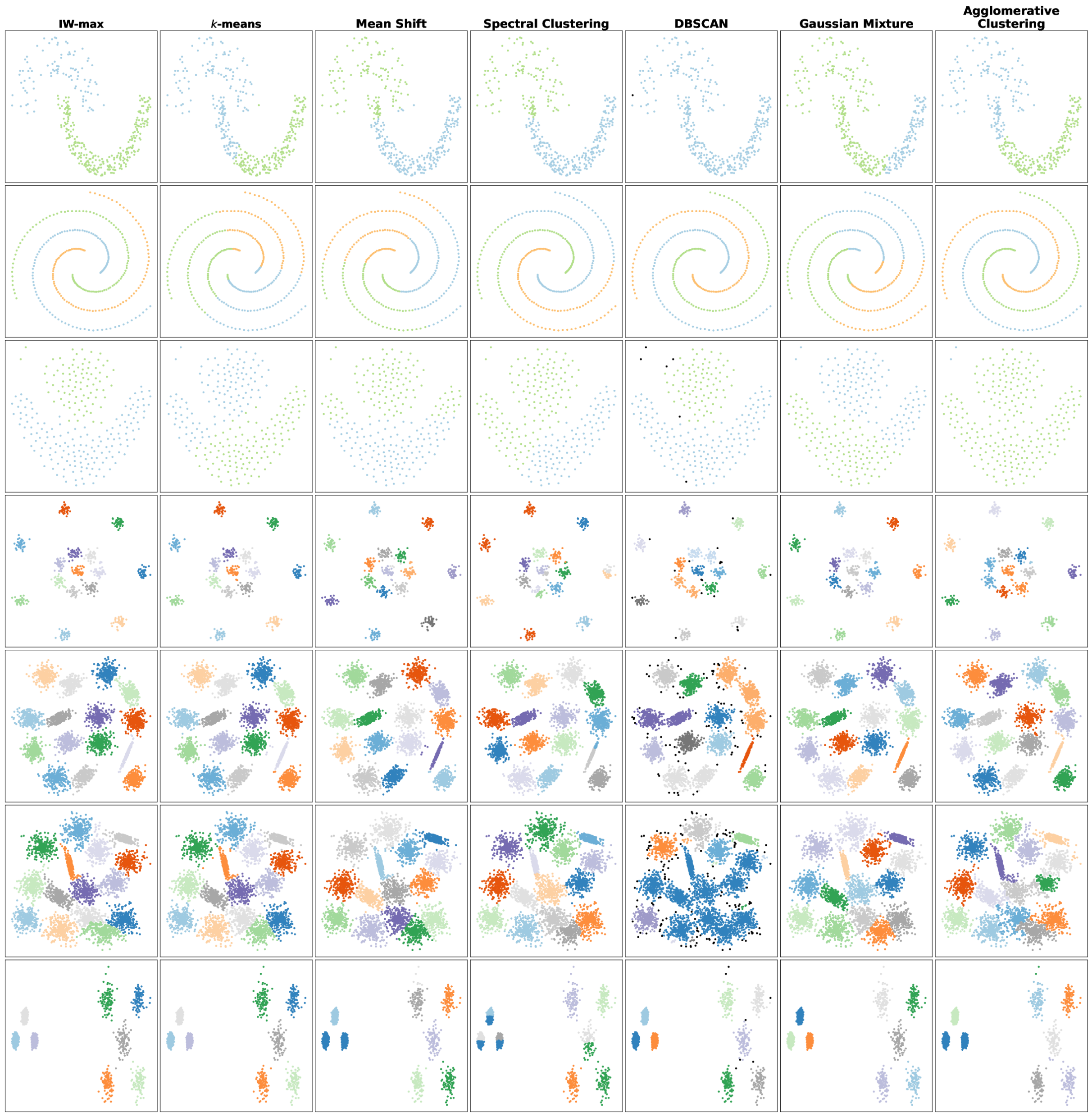


Figure 5.24: Clustering results on the synthetic datasets using IW-max with a complete graph and the seed selection algorithm against k-means, mean shift, spectral clustering, DBSCAN, Gaussian mixture and agglomerative clustering.

the only methods that correctly discover the clusters are the IW-max and Gaussian mixture techniques. Lastly, for the unbalance dataset, the methods that effectively separate the groups are IW-max, k-means, DBSCAN and Gaussian mixture. DBSCAN works effectively in cases where the clusters are well-separated and inter-cluster proximity is significantly larger than the distances between samples within a cluster. However, this is clearly not the case for the jain, s1 and s2 datasets. Gaussian mixture works better for clusters of elliptical shape and when there is no inter-cluster overlapping.

5.4 Concluding Remarks

In this chapter, we discuss the experiments and results of applying the IOPF framework to the following three applications: (*i*) object delineation, (*ii*) analysis on road networks and (*iii*) clustering of synthetic datasets, which impose different constraints on the definition of the graph topology. In this context, we design four IOPF-based solutions, namely IOPF-dynsum, IOPF-dynmax (IDT), IW-sum and IW-max, to address the aforementioned problems. The best IOPF-based methods for each applications are IOPF-dynmax (IDT), IW-sum and IW-max, respectively. Additionally, the inclusion of the seed selection algorithm proposed in Section 4.2 and the imposing of restrictions on the graph topology may improve the results obtained by IW-sum and IW-max. In the next and final chapter, we state the conclusion and future work.

Chapter 6

Conclusion and Future Work

We introduced a flexible and robust graph-based clustering framework, called Iterative Optimum-Path Forest (IOPF), that employs subsequent executions of the Optimum-Path Forest algorithm from reestimated seed sets to partition an input dataset. This framework allows the design of connectivity-based clustering methods by suitable choice of its constituent components. In addition, we introduced an algorithm to select initial seeds for data clustering, improving a previous work [76]. In this context, we presented four IOPF-based clustering methods, IW-sum, IW-max, IOPF-dynsum and IOPF-dynmax. We evaluated them for object delineation, identification of emergency stations in road networks, and clustering of synthetic datasets with various shapes and sizes in a 2D feature space.

We observed that IW-sum improves its effectiveness when the graph topology is constrained to the k-nearest neighbors. On the other hand, IOPF-dynmax, previously called IDT [9], is identified as the best approach for object delineation; IW-sum is the best method for identification of emergency stations in road networks, and IW-max is the winner in the clustering of two-dimensional datasets. Finally, it is worth noting that the proposed versions of IW-sum and IW-max adopt the initial seed selection method introduced in Section 4.4 and the choice of the forest that minimizes the total path-costs, as presented in Section 4.5. Therefore, we can conclude that IW-sum and IW-max are relevant improvements concerning their original versions [76].

The results show that IOPF-dynmax (IDT) can be more effective than DISF [12] for object delineation, leaving as future work a comparative study between them for superpixel segmentation. In the identification of emergency stations in road networks, IW-sum has already shown superior performance than k-means in a previous work [76]. In the present work, we demonstrate the advantages of IW-max over k-means, spectral clustering, DBSCAN, mean shift, Gaussian mixture models, and agglomerative clustering using synthetic datasets. Hence, we may conclude that IOPF is a flexible and robust framework that can be used to design clustering solutions to address a wide variety of problems, being IW-sum, IW-max, and IOPF-dynmax (IDT) its best clustering methods.

As future work, we intend to investigate new techniques for seed recomputation to further improve the effectiveness of IOPF-based methods, to include local density information to identify initial seeds, and to explore new applications for IOPF-based techniques, more specifically, in the context of the clustering of multidimensional datasets. Furthermore, we believe that the development of strategies to design a suitable graph topology, in cases where it can not be induced from the problem definition, seems a plausible research direction as well.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] R. Achanta and S. Susstrunk. Superpixels and polygons using simple non-iterative clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4660, 2017.

[3] L. Afonso, A. Vidal, M. Kuroda, A. X. Falcão, and J. P. Papa. Learning to classify seismic images with deep optimum-path forest. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 401–407, 2016.

[4] L. C. Afonso, C. R. Pereira, S. A. Weber, C. Hook, A. X. Falcão, and J. P. Papa. Hierarchical learning using deep optimum-path forest. *Journal of Visual Communication and Image Representation*, 71:102823, 2020.

[5] C. C. Aggarwal. *Graph Clustering*, pages 459–467. Springer US, Boston, MA, 2010.

[6] M. Ahmed, R. Seraj, and S. M. S. Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8), 2020.

[7] S. Alpert, M. Galun, R. Basri, and A. Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[8] W. P. Amorim, A. X. Falcão, and M. H. d. Carvalho. Semi-supervised pattern classification using optimum-path forest. In *2014 27th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 111–118, 2014.

[9] D. Aparco-Cardenas, P. J. de Rezende, and A. X. Falcão. Object delineation by iterative dynamic trees. In *Iberoamerican Congress on Pattern Recognition*, pages 1–10, 2021.

[10] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[11] F. Belém, S. J. F. Guimarães, and A. X. Falcão. Superpixel segmentation by object-based iterative spanning forest. In *Iberoamerican Congress on Pattern Recognition*, pages 334–341. Springer, 2018.

[12] F. C. Belém, S. J. F. Guimarães, and A. X. Falcão. Superpixel segmentation using dynamic and iterative spanning forest. *IEEE Signal Processing Letters*, 27:1440–1444, 2020.

[13] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[14] J. Bragantini, S. B. Martins, C. Castelo-Fernandez, and A. X. Falcão. Graph-based image segmentation using dynamic trees. In *Iberoamerican Congress on Pattern Recognition*, pages 470–478. Springer, 2018.

[15] F. A. Cappabianco, A. X. Falcão, C. L. Yasuda, and J. K. Udupa. Brain tissue mr-image segmentation via optimum-path forest clustering. *Computer Vision and Image Understanding*, 116(10):1047–1059, 2012.

[16] G. Castellano, R. A. Lotufo, A. X. Falcão, and F. Cendes. Characterization of the human cortex in mr images through the image foresting transform. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 1, pages I–357, 2003.

[17] C. Castelo-Fernández and A. X. Falcão. Learning visual dictionaries from class-specific superpixel segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 171–182. Springer, 2019.

[18] S. Chen, T. Sun, F. Yang, H. Sun, and Y. Guan. An improved optimum-path forest clustering algorithm for remote sensing image segmentation. *Computers & Geosciences*, 112:38–46, 2018.

[19] F. R. Chung and F. C. Graham. *Spectral graph theory.* Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[20] K. C. Ciesielski, A. X. Falcão, and P. A. Miranda. Path-value functions for which dijkstra's algorithm returns optimal mapping. *Journal of Mathematical Imaging and Vision*, 60(7):1025–1036, 2018.

[21] G. B. Coleman and H. C. Andrews. Image segmentation by clustering. *Proceedings of the IEEE*, 67(5):773–785, 1979.

[22] R. Collobert, F. Sinz, J. Weston, L. Bottou, and T. Joachims. Large scale transductive svms. *Journal of Machine Learning Research*, 7(8), 2006.

[23] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[24] K. A. Costa, L. A. Pereira, R. Y. Nakamura, C. R. Pereira, J. P. Papa, and A. X. Falcão. A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks. *Information Sciences*, 294:95–108, 2015.

[25] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1384–1399, 2011.

[26] K. S. Dar, I. Javed, W. Amjad, S. Aslam, and A. Shamim. Survey of clustering applications. *Journal of Network Communications and Emerging Technologies (JNCET)*, 4(3), 2015.

[27] A. Echemendía Montero and A. X. Falcão. A divide-and-conquer clustering approach based on optimum-path forest. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 416–423, 2018.

[28] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[29] R. M. Esteves, T. Hacker, and C. Rong. Competitive k-means, a new accurate and distributed k-means algorithm for large datasets. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 17–24, 2013.

[30] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279, 2014.

[31] A. X. Falcão, J. Stolfi, and R. de Alencar Lotufo. The image foresting transform: theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[32] A. X. Falcão, B. S. Cunha, and R. A. Lotufo. Design of connected operators using the image foresting transform. In M. Sonka and K. M. Hanson, editors, *Medical Imaging 2001: Image Processing*, volume 4322, pages 468 – 479. International Society for Optics and Photonics, SPIE, 2001.

[33] A. Falcão, L. da Fontoura Costa, and B. da Cunha. Multiscale skeletons by image foresting transform and its application to neuromorphometry. *Pattern Recognition*, 35(7):1571–1582, 2002.

[34] A. X. Falcão and J. P. Papa. *Optimum-Path Forest: Theory, Algorithms, and Applications.* Academic Press, 2022.

[35] E. Forgey. Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

[36] P. Fränti and S. Sieranoja. Clustering basic benchmark. `http://cs.joensuu.fi/sipu/datasets/`. Accessed: 2020-09-30.

[37] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(3):469–475, 2006.

[38] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. *ACM Sigmod record*, 27(2):73–84, 1998.

[39] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[40] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical programming*, 79(1):191–215, 1997.

[41] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4):175 – 181, 2000.

[42] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[43] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[44] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[45] A. Karduni, A. Kermanshah, and S. Derrible. A protocol to convert spatial polyline data to network formats and applications to world urban road networks. *Scientific data*, 3(1):1–7, 2016.

[46] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[47] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[48] F. Lemes Galvão, A. X. Falcão, and A. Shankar Chowdhury. Risf: Recursive iterative spanning forest for superpixel segmentation. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 408–415, 2018.

[49] Z. Li and J. Chen. Superpixel segmentation using linear spectral clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1356–1363, 2015.

[50] Y. Liu, M. Yu, B. Li, and Y. He. Intrinsic manifold slic: A simple and efficient method for computing content-sensitive superpixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):653–666, 2018.

[51] R. A. Lotufo, A. X. Falcão, and F. A. Zampirolli. Ift-watershed from gray-scale marker. In *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing*, pages 146–152, 2002.

[52] E. J. d. S. Luz, T. M. Nunes, V. H. C. De Albuquerque, J. P. Papa, and D. Menotti. Ecg arrhythmia classification based on optimum-path forest. *Expert Systems with Applications*, 40(9):3561–3573, 2013.

[53] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[54] L. A. C. Mansilla and P. A. V. Miranda. Oriented image foresting transform segmentation: Connectivity constraints with adjustable width. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 289–296, 2016.

[55] L. A. C. Mansilla, P. A. V. V. Miranda, and F. A. M. Cappabianco. Image segmentation by image foresting transform with non-smooth connectivity functions. In *2013 XXVI Conference on Graphics, Patterns and Images*, pages 147–154, 2013.

[56] S. B. Martins, G. Ruppert, F. Reis, C. L. Yasuda, and A. X. Falcão. A supervoxel-based approach for unsupervised abnormal asymmetry detection in mr images of the brain. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 882–885, 2019.

[57] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, 26(4):354–359, 1983.

[58] P. Neubert and P. Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, volume 6, pages 1–12, 2012.

[59] J. P. Ortega, M. Del, R. B. Rojas, and M. J. Somodevilla. Research issues on k-means algorithm: An experimental trial using matlab. In *CEUR workshop proceedings: semantic web and new technologies*, pages 83–96, 2009.

[60] J. P. Papa and A. X. Falcão. A new variant of the optimum-path forest classifier. In *International Symposium on Visual Computing*, pages 935–944. Springer, 2008.

[61] J. P. Papa and A. X. Falcão. Optimum-path forest: a novel and powerful framework for supervised graph-based pattern recognition techniques. *Institute of Computing - University of Campinas*, 2010.

[62] J. P. Papa, A. X. Falcão, V. H. C. De Albuquerque, and J. M. R. Tavares. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45(1):512–520, 2012.

[63] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[64] S. Patel, S. Sihmar, and A. Jatain. A study of hierarchical clustering algorithms. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 537–541. IEEE, 2015.

[65] D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734, 2000.

[66] C. C. Ramos, A. N. Souza, J. P. Papa, and A. X. Falcão. Learning to identify nontechnical losses with optimum-path forest. In *Proceedings of the 17th International Conference on Systems, Signals and Image Processing (IWSSIP 2010)*, pages 154–157, 2010.

[67] D. A. Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741:659–663, 2009.

[68] L. M. Rocha, F. A. Cappabianco, and A. X. Falcão. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, 19(2):50–68, 2009.

[69] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues. Clustering algorithms: A comparative approach. *PloS one*, 14(1):e0210236, 2019.

[70] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, page 309–314, New York, NY, USA, 2004. Association for Computing Machinery.

[71] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[72] P. T. Saito, P. J. de Rezende, A. X. Falcão, C. T. Suzuki, and J. F. Gomes. A data reduction and organization approach for efficient image annotation. In *Proceedings of the 28th annual ACM symposium on applied computing*, pages 53–57, 2013.

[73] S. E. Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[74] R. Sharan and R. Shamir. Click: a clustering algorithm with applications to gene expression analysis. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, volume 8, page 16, 2000.

[75] P. H. Sneath. The application of computers to taxonomy. *Microbiology*, 17(1):201–226, 1957.

[76] S. Soor, A. Challa, S. Danda, B. Daya Sagar, and L. Najman. Iterated watersheds, a connected variation of k-means for clustering gis data. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2019.

[77] T. A. Sorensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biol. Skar.*, 5:1–34, 1948.

[78] J. E. Vargas-Muñoz, A. S. Chowdhury, E. B. Alexandre, F. L. Galvão, P. A. Vechiatto Miranda, and A. X. Falcão. An iterative spanning forest framework for superpixel segmentation. *IEEE Transactions on Image Processing*, 28(7):3477–3489, 2019.

[79] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, Dec. 2010.

[80] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[81] X. Wei, Q. Yang, Y. Gong, N. Ahuja, and M. Yang. Superpixel hierarchy. *IEEE Transactions on Image Processing*, 27(10):4838–4849, 2018.

[82] X. Xiao, Y. Zhou, and Y. Gong. Content-adaptive superpixel segmentation. *IEEE Transactions on Image Processing*, 27(6):2883–2896, 2018.

[83] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[84] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1):68–86, 1971.

[85] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.