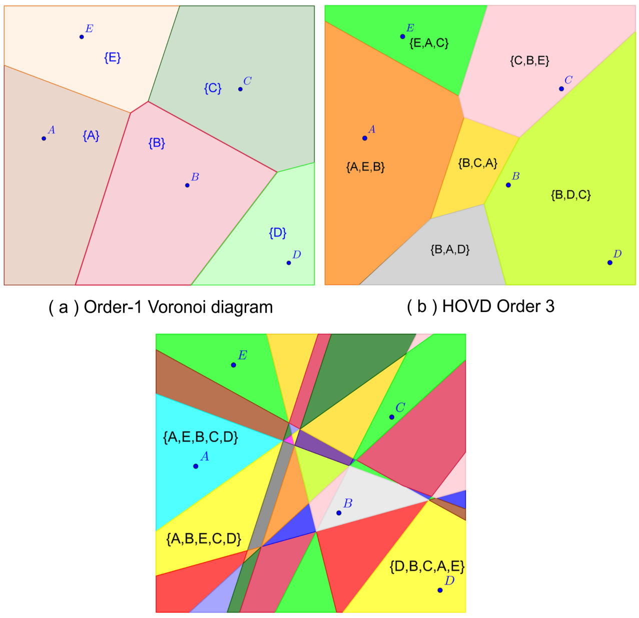
Chapter I

Introduction

1.1 Background

Voronoi diagram is a method that divides the plane into smaller area/region based on the nearest distance to an object [1]. Voronoi diagrams are very useful in computational geometry, particularly for representation or quantization problems, used in the field of robotics for creating a protocol for avoiding detected obstacles [6]. The regions in a Voronoi diagram are called Voronoi cells. When a cell considers a point as the nearest neighbour it is called ordinary Voronoi diagram. As describe in Figure 1.1a, each Voronoi cells contain a single information of the nearest generator point. When a cell considers more than one point as k nearest neighbour, it is called Higher Order Voronoi diagram. The number of nearest generator points information on a Voronoi cell are depend on its k value itself. In Figure 1.1b is higher order Voronoi diagram order-3, it means each Voronoi cell will have 3 nearest generator points information. This reference [4] revealed that the complexity of Voronoi diagram is at most quadratic.

Since ordinary Voronoi diagram and higher order Voronoi diagram doesn't provides a detailed distance sequence to all generator points in each Voronoi cell, then comes a new variation of Voronoi diagram called highest order Voronoi diagram (HSVD). Highest order Voronoi diagram (HSVD) is an extension of Higher Order Voronoi Diagram (HOVD) with wider possible applications. Highest order Voronoi diagram can be used on the field of query processing such as reverse k-nearest neighbour (RKNN), k farthest neighbour (KFN), k nearest neighbour(KNN), etc [1]. Figure 1.1 shows the differences between an ordinary Voronoi diagram, highest order Voronoi diagram order-3 and highest order Voronoi diagram from 5 generator points. From Figure 1.1c, can be seen that HSVD construct more Voronoi cells than other Voronoi diagram variant and also calculate the sequence of Voronoi cells to all generator points. The complexity of HSVD construction is on O(m^4), where m is the number of generator points [1].



(a) Order-1 Voronoi diagram

(b) HOVD Order 3

(c) HSVD

Figure 1.1: An example of Voronoi diagram based on the order.

From related works, there are method called Fast Labelling and Interchange Position (FLIP) and Left with Least-Angle Movement (LAM) used to construct highest order Voronoi diagram [1] [3]. But, both of this method implemented on conventional computing and have limitation on number of points that can be processed and execution time is quite high. Processing on conventional computing caused an inefficiency of reuse a working set of data process which caused the execution time is quite high and limit the number of points that can be processed. Beside that, conventional computing didn't utilize all the available resources.

There are frameworks that can be used to utilize all the available resources to optimize the computing process called distributed computing framework. The popular distributed computing frameworks are Hadoop and Apache Spark. Compared to Hadoop, this paper [14] showed that Apache Spark work well on iterative process that reuse a set of data because of the ability to do in-memory processing. Apache Spark is a distributed computing framework that focussed on reuse a working set of data across multiple parallel operations [14]. Apache Spark usually used on the field of data mining and machine learning that processing a large file size data as input and resulting on much smaller size of data output [14]. But, this implementation do the opposite. Apache Spark started with small file size data (less than 1 MB) as an input, then the input data processed on Apache Spark and generate more large size of data as the output data.

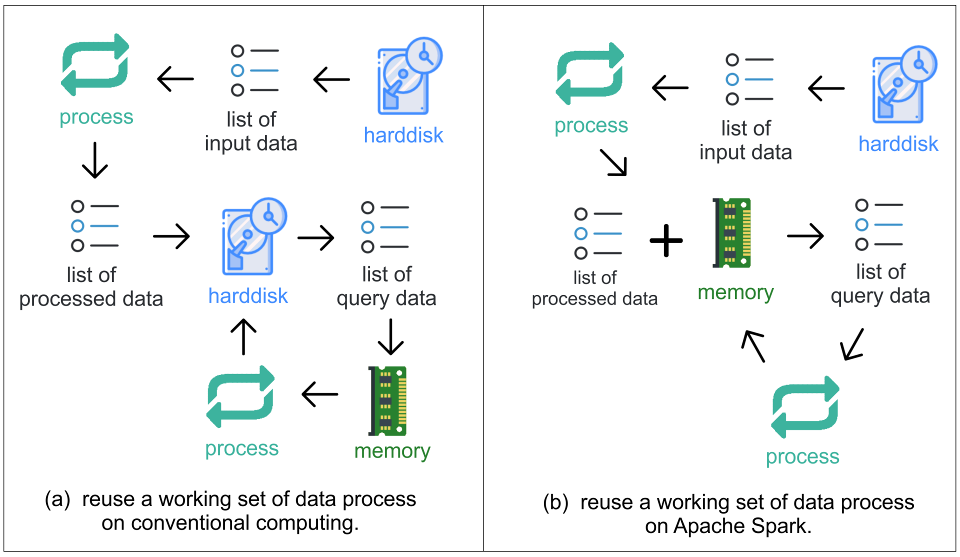


Figure 1.2: Comparison on reuse processing of list of data

Considering Apache Spark will distribute the task to all available resources and put in-memory processing first, the process of reuse a working set of data on highest order Voronoi diagram construction can be improved. As shown in Figure 1.2, when conventional computing have to access disk repeatedly to get list of query data, by keeping the processed data in memory, costly disk accesses can be avoided. In the next process when the data is needed then in-memory processing can be done. To optimize the in-memory processing of highest order Voronoi diagram on Apache Spark, two diferrent labelling method are tried. Those two labelling method are flip labelling and centroid labelling. The comparison of two labelling method will be done and the method with smaller execution time will be choosen.

This minor thesis shows that with the help of Apache Spark framework, the number of point that can be processed increase to 24 with the execution time is 60% faster than LAM implementation in average. This minor thesis also wanted to find out how Apache Spark perform on generating large size of data from small size data input compared to conventional computing by comparing the execution time of this implementation with the previous [3] implementation of highest order Voronoi diagram construction.

1.2 Statement of Problem

Based on the above overview, problems can be formulated as follows:

1. How to reuse a working set of data, so the execution time of highest order Voronoi processing on Apache Spark can be reduced?

2. How does highest order Voronoi diagram construction method on Apache

Spark performance in term of execution time and number of point that can be processed ?

1.3 Objective

The objective that we want to be achieved on this final project are as follows:

1. Adapt the most suitable method on highest order Voronoi diagram construction on Apache Spark so that reuse a working set of data can be done efficiently.

2. Analyze the highest order Voronoi diagram construction on Apache Spark performance.

1.4 Scope

The scopes of this minor thesis is the point dataset used is random uniform distributed.

1.5 Hypothesis

Apache Spark is a distributed computing framework that focussed on reuse a working set of data across multiple parallel operations. In highest order Voronoi diagram construction from previous implementation there are inefficiency of reuse a working set of data process which caused the execution time is quite high and limit the numeber of points that can be processed. Therefore, with the help of Apache Spark framework, we can achieve what we desired, which is increase the number of point that can be processed and able to reduce the execution time of highest order Voronoi diagram construction.