UNIVERSITY OF CALIFORNIA
RIVERSIDE

Scalable Management, Visualization and Regionalization for Spatial Data

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

Ziang Zhao

March 2019

Thesis Committee:

Dr. Amr Magdy, Co-Chairperson
Dr. Ran Wei, Co-Chairperson
Dr. Ahmed Eldawy

The Thesis of Ziang Zhao is approved:

Committee Co-Chairperson

Committee Co-Chairperson

University of California, Riverside

Acknowledgments

To begin with, this paper should give its most of the credit to Dr. Amr Magdy and Dr. Ran Wei. Without their lasting and invaluable support for all various aspects of my research work, I would not have finished this paper and be able to defend my Master degree at UC,Riverside. Dr. Amr has spent much time and effort in advising me with his in-depth insights in spatial data management and visualization, as well as his solid skills in academic writing. While Dr. Ran Wei guided me in spatial regions aggregation and heuristic algorithms to the top of her bent, which has advanced my understandings in spatial data regionalization to a great extent. I was so fortunate to work with these two professors.

It is also my pleasure to have Dr. Ahmed Eldawy as my thesis committee member, his advice and comments on the thesis have made this work sound and sturdy in every aspect in terms of spatial data management and analysis.

Besides, I would like to thank Mr. Yunfan Kang and Mr. Win Cowger for their kind and selfless research efforts for facilitating my work, without them this paper would not have reached here.

In the end, I would like to deliver my heartfelt gratitude to my parents Mrs. Yihua Liu and Mr. Lixun Zhao as well as my Uncle Mingzhi Liu, for their supporting me in pursuing my Master degree abroad, and my girl friend Miss Kangqin Cheng, for her unique and endless encouragement all the time. It is blessed to receive their sincere love from all perspectives. They are my impetus to be fearless towards my objective.

To my family and my beloved girl friend Kangqin Cheng for all the support.

ABSTRACT OF THE THESIS

Scalable Management, Visualization and Regionalization for Spatial Data

by

Ziang Zhao

Master of Science, Graduate Program in Computer Science
University of California, Riverside, March 2019
Dr. Amr Magdy, Co-Chairperson
Dr. Ran Wei, Co-Chairperson

Briefly, this thesis paper consists of two parts, concerning spatial data management and analysis. The first part is targeted at dealing with spatial data management and visualization given some anthropogenic litter data. While the second part aims to leverage some algorithmic techniques to enhance the performance of a proposed baseline algorithm for spatial data regionalization. The combination of both works facilitates the comprehension and real practice from system design to data analysis in terms of spatial data.

In the first part, anthropogenic litter data: data about waste that originates from human activities such as food waste, diapers, construction materials, used motor oil, hypodermic needles, etc, is causing growing problems for the environment and quality of life in modern cities in recent years. Such data has significant importance in the field of environmental sciences due to its important use cases that span saving marine life, reducing the risk from natural hazards, etc. In this paper, we introduce a data-driven approach that enables environmental scientists and organizations to track, manage, and model anthropogenic litter data at a large scale through smart technologies. We make a major

on-going effort to collect and maintain this data worldwide from different sources through a community of environmental scientists and partner organizations. With the increasing volume of collected datasets, existing software packages, such as GIS software, do not scale to process, query, and visualize such data. To overcome this, we provide a scalable data management and visualization framework, called CleanUpOurWorld, that digests datasets from different sources, with different formats, in a scalable backend that cleans, integrates, and unifies them in a structured form. The backend includes four main modules: a data cleaner, a data integrator and loader, a data store, and a query processor. On top of this backend, frontend applications are built to visualize litter data at multiple spatial levels, from continents and oceans to street level, to enable new opportunities for both environmental scientists and organizations to track, model, and clean up litter data. The current CleanUpOurWorld implementation is based on thirty real datasets and provides different interfaces for different kinds of users.

In the second part, spatial areas with multiple attributes are considered to be aggregated into larger regions. These spatial attributes in one area, when taken account of by the researchers can be divided into two categories in our case. One category is based on the similarity measure, such as degree of diversity, income per area, etc. The other can be concluded as the extensive threshold attributes, such as population in one area or other properties that ensure aggregation quality by giving constraints. We proposed a novel algorithm to solve the max-p-regions problem including larger initial region generation in the construction phase and less heterogeneity in the local search phase compared to previous benchmark algorithms. We conducted the experiments on multi-core platforms to ensure

that parallelism is well exploited in a meta-heuristic approach. The novel algorithm provided insights in an empirical and quantitative way that can facilitate future research on this topic.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Anthropogenic Data Management and Visualization

Large-scale environmental problems involve managing and processing large datasets [15]. A prominent example is anthropogenic litter data, which is data about waste that originates from human activities such as food waste, diapers, construction materials, used motor oil, hypodermic needles, etc. This waste is generated in a daily basis worldwide as a part of human daily activities such as eating, working, manufacturing, entertainment, medical treatments, etc. A significant part of this waste ends up in natural dumpsters, such as oceans, which causes several environmental problems [23], [35], [5], e.g., destroying marine life and increasing the impact of natural hazards like floods. This phenomenon is becoming globally more dangerous to the extent that president of the United States (US) signed Save Our Seas Act in October 2018 committing the US to expand efforts to clean up nearly 8

million metric tons of litter polluting the worlds oceans [26]. In addition, this increasing data is affecting the quality of life in large populations urban areas, such as undeveloped neighborhoods and poor countries, through spreading diseases and health problems [28], [9]. Therefore, many environmental scientists and several community and governmental organizations are interested in collecting and tracking waste data as a preliminary step towards addressing these problems. Currently, scientists use smart technologies, such as GIS software, to load and visualize data from raw files in the spatial space. However, with the increasing volume of the collected data worldwide, GIS software does not scale to visualize hundreds of thousands, or even millions, of data points. In addition, GIS software is limited in functionality to solve litter data issues such as cleaning noisy data and large-scale data aggregation. This hinders the current progress in several environmental projects.

This paper introduces CleanUpOurWorld: a scalable research data management framework that enables scientists and organizations to collect, process, query, and visualize human waste data. CleanUpOurWorld stores both raw data points and aggregate data over different spatial levels to enable efficient visualization in large spatial areas. In specific, we currently provide six visualization levels: continent, sub-continent, country, sub-country, city, and street levels. The street level shows individual data points while all other levels show aggregate number of data points classified based on waste type. Users can interactively navigate data through zooming in/out on different spatial levels, panning over different spatial regions, and filtering based on waste type, to visualize and analyze different portions of data with real-time response. In addition, users can add new litter data sources and download subset of existing data.

To enable such scalable visualization and management, CleanUpOurWorld employs a backend that preprocesses different datasets, with different formats, to a common intermediate format. Then, the converted data is fed into a data cleaner that resolves data inconsistency problems such as column names deficiencies, data values irregularities, and inconsistent data formats. The cleaned data is forwarded to a data integrator and loader that integrates data from different sources in a snowflake-like scheme. Common attributes that exists in all datasets are separated and linked to original data records through record identifiers. In addition, data is aggregated at different spatial levels to enable efficient visualization in large areas. Then, all such data are loaded to the data store. The data store is relational PostGIS database that stores each data source in a separate relational table in addition to two tables of integrated and aggregated data. Such database scales to store and query millions of data points, which is enough scale for the first stage of this project. Finally, a query processor is added to the backend to play a mediator role that receives queries from the application level and answer them through accessing the data store with SQL queries. On top of this backend, web-based frontend applications are built to enable users to analyze existing data and input new datasets. CleanUpOurWorld has currently digested thirty different data sources collected by environmental scientists and collaborator organizations. New datasets are being added through collaborations and crowd-sourced data collection.

1.2 Introduction to Spatial Data Regionalization

A homogeneous region is proposed in [11]'s work that is formed by some spatially contiguous areas at lower level, with a relatively low heterogeneity in every region. The problem of aggregating such small areas into regions has been studied in various works [21] [14]. Starting from this initial version of regional aggregation problem, its variants are later introduced and studied, which is adding some constraints when increasing homogeneity. The original intention of introducing constraints is to be more adapted to its more empirical applications in real world.For instance, the precision of the rate estimation for one region is inversely proportional to its size in population. Shapes of regions, manual divided boundaries or some other threshold attributes besides the similarity measure are some typical constraints.

Though many efforts have been taken to employ different strategies based on different formulations, resolving a suitable region number is always controversial when taking the tradeoff of region number and total homogeneity into consideration. According to Duque's [7] new formulation scheme on spatial aggregation model, it tries to extend p-regions problem by imposing a fixed maximal number of regions. Most researchers who have research interest on this topic have profound understandings on the heterogeneity and threshold attribute, while they are not so aware of how to set a proper parameter of the number of regions. However, this scheme helps to resolve this issue by maximizing the initial amount of regions, meaning minimizing the number of aggregation in one area, so as to reduce the loss of similarity observation in one region when the threshold is satisfied. In real world applications, it is not always the case that p-regions can be extended to max-p-regions.

For instance, in political district reform, the researcher might have already set the fixed number of possible regions. Some researcher might have also investigated that when public services(fire service or dmv relocation) need to be assigned to some areas without new stations available, it cannot be referred to as max-p-regions. However, optimal geographic supports for spatially varying statistical estimates might benefit a lot from max-p-regions like disease incidence rates estimation and census tracts [1]. Aggregating areas into regions in a max-p-regions manner would make the succeeding efforts more promising.

The well-formulated max-p-problem in Duque's work [7] predefined the number of regions as endogenous parameter and a two-phase based strategy has been proposed for further possible solutions on both phases.The first phase is straightforward as it corresponds to the max initial region number generation. The algorithm incorporates multiple initial solutions and filter candidates that at least equals the current max region number. Then homogeneity is considered and increased by some local search algorithms(e.g., greedy, simulated annealing, tabu search). This solution thus contained some useful characteristics to facilitate further research, such as self-dictated region shape and area unit minimization(by maximizing region number p). However, it only provided primary solutions that might have considerable space to improve in both phases. To bridge the gap of these primary solutions and the real cases when studying regions, research efforts are still demanded on optimizing this two-stage strategy.

Since the max-p-regions problem have been theoretically studied to have unacceptable time cost when problem size grows large [4] [18], meaning it is not feasible to find a optimal solution in deterministic time. This feature leads to various tentative algorithms,

especially heuristics. One key insight to employ such heuristic in a preferably manner is its property of evading local optima, caused by some uncertain moves from one solution to another in solution search space. An intuitive way to get objective function better optimized is to have larger search space while not losing scalability. As the computing power dramatically increasing, multi-core or massive parallel architectures [19] [17] are commonly used to speed up various algorithms and tasks. To enlarge the search space of such widely used heuristic methods, parallel processing techniques can also be applied in max-p-regions problem from the same initial solution generated by the first phase. Move based restrictions can also be applied to ensure better homogeneity measure output while introducing some synchronization cost. Besides, this paper presents another initial seeding strategy for acquiring a steadily larger max-p than original method.

1.3 Thesis Organization

The first chapter of this thesis gives introductions on both two portions of the thesis, describes the research space and directions of these two problems concerning spatial data, and briefly states the solution that is proposed by this thesis work. The second chapter would be contents regarding management and visualization techniques regarding these anthropogenic litter data, which consists of overview, backend and frontend design and implementation. Then the third chapter outlines the problem of spatial data regionalization, which involves the background, algorithm design and experiments on the problem. The last chapter gives conclusions on both problems and proposes some possible future works on spatial data management and regionalization analysis in terms of the thesis topics.

Chapter 2

Scalable Management and Visualization for Anthropogenic Litter Data

2.1 Overview

This section gives an overview about CleanUpOurWorld components and operations. Figure 2.1 presents CleanUpOurWorld architecture that consists of a scalable backend in addition to frontend applications that build on top of the backend and provide interactions with end users and administrators. The backend takes collected datasets and converts them to a common intermediate data format, comma separated values with UTF8 encoding, to standardize the input format from different data sources. Then, the converted data is fed to a pipeline of data cleaner, data integrator and loader, and data store that preprocesses

the data and store them in a scalable database. Finally, the query processor receives queries from end users through frontend applications and accesses the data through SQL queries.

CleanUpOurWorld currently digested the first batch of real datasets collected from thirty different sources with total of 420K data points. New data batches with hundreds of thousands of points are being collected as described later. The section 2.2 gives backend design for processing and manage the data. The section 2.3 depicts the frontend design for users to access and analyze processed data.

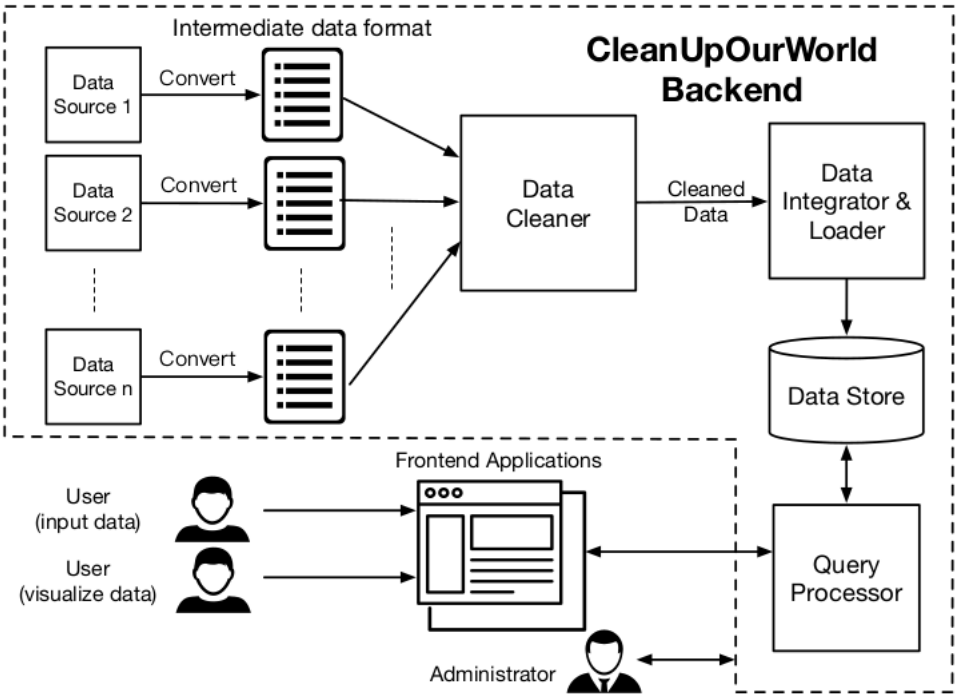


Figure 2.1: CleanUpOurWorld Architecture

2.2 Backend Design and Implementation

CleanUpOurWorld backend consists of a pipeline of modules for data collection, data conversions, data cleaning, data integration and loading, data storage, and query processing. Each is briefly outlined below in the following content.

2.2.1 Data collection, formatting, and conversions

In collaboration with Lets Do it World [6], a nonprofit organization based in Tallinn, Estonia, Gray Lab at the University of California, Riverside started the data collection process by contacting organizations that host the largest datasets such as Ocean Conservancy [5], Litterati [34], Marine Debris Tracker [31], Alice Ferguson Foundation [12], and US National Oceanic and Atmospheric Administration (NOAA) [22]. Some organizations, e.g., Ocean Conservancy and Marine Debris Tracker, provide open access to their data while others, e.g., Alice Ferguson Foundation and NOAA, gave us special permissions to access their data. We have collected thirty datasets so far and we are currently working on adding nine more datasets with collaborator organization and open data sources. In addition, we are running a crowd-sourcing data collection project in Southern California that employs twenty undergraduate environmental sciences students. Datasets are collected in four formats: ESRI Shape files, KML files, XLSX files, and CSV files. Then, all formats are converted into a common CSV format with UTF8 encoding as described earlier.

2.2.2 Data cleaning

The automated data cleaning process includes three operations. The first operation is regulating columns names to avoid empty names, duplicate names, mismatched names, and names that do not comply with SQL standards. The problem of mismatched names arise when a common essential attribute, such as location, appears with multiple names. We detect this for location and date attributes through comparing both data types and names and fill up with a common attribute name that is used in all datasets. Other than the problem of mismatched names, the problems of empty, duplicate, and non-compliant names are fixed with arbitrary names, through regular expressions, prompting the user asynchronously to alter them with new names if needed. The second operation is fixing data irregularities where some data values are not compliant with the data type and in other cases values of the same attribute has different formats. Regular expressions are used to discover non-compliant values and unify the format of the same data type. For example, latitude and longitude polarities are always indicated with +/signs and dates has MMDDYYYY format. For noncompliant values, they are filtered out for further manual processing so that the column data type can be assigned the appropriate data type, e.g., floating point number. The third operation is accommodating different representations for the same attribute, which goes beyond simple data irregularities such as format mismatches. For example, the location attribute is an essential attribute in all our datasets and its common format is a pair of latitude/longitude coordinates. However, some datasets have this attribute as a rough textual description, such as street name, city name, etc. To address this, we employ a Python library GeoPy, with N ominatim geocoder, that converts

this text into precise coordinates. Given the place textual description, GeoPy is searching OpenStreetMaps data to generate synthetic addresses of points. An important parameter that significantly affects the precision of interpretation from addresses to coordinates is adding a country bias as an initial parameter to help disambiguating similar places. This is practical for litter data because the data source organization is usually organizing data in one country, so it is common to have a single value for the country bias parameter.

2.2.3 Data integration, loading, storage, and visualization

After each dataset is cleaned, it is forwarded to a data integrator and loader module. This module goes over three steps. The first step loads each dataset in a separate SQL table, which contains all input data attributes, in a PostGIS database that represents the main data store shown in Figure 2.1. The loading process is performed through a dynamic SQL function, using plpgsql language, that takes as input the dataset name and a CSV file that contains column names in the first row and data records in all subsequent rows. Then, it creates a SQL table and populates the data from the file to this SQL table adding an auto increment primary key field to the columns. The second step loads only essential attributes of each record from the newly loaded dataset in a centralized table, called maintable, that integrates data from all existing datasets in a snowflakelike fashion. This table is created to scale up query processing as we will elaborate later. Briefly, the query processor posts to this table queries that need a few pieces of information, yet they are posted frequently. The maintable have three essential attributes per record: a location as latitude/longitude pair, a date, and a collecting organization, in addition to the record id and the dataset name to link the record back to its original dataset table with full

attribute set. The maintable has one record corresponds to each record in each dataset, so its total number of records equals the summation of all records in all datasets. The third step aggregates data from the maintable in the aggregatetable. In fact, maintable has a large number of raw data points with a street-level granularity. However, when scientists visualize data in large regions, e.g., city, country, continent, or ocean, the large number of points that exist in this large region will limit the existing visualization frontends, such as GoogleMaps and GIS software, to show all points while still being interactive to users. To overcome this problem, CleanUpOurWorld visualizes individual data points only at the street level, while aggregate data is visualized on higher levels such as whole cities, countries, continents, and oceans. To speed up such aggregate visualization, aggregatetable maintains aggregate counts from maintable at five different spatial levels that corresponds to the frontend visualization levels. At each level, the whole world is divided into a set of spatial tiles. Then, the aggregatetable maintains the number of data points in each spatial tile classified by the litter type. When a visualization query comes, this table is queried to retrieve data right away and visualize them to users.

The most intuitive way of storing these aggregate counts is to generate records for all areas at each different level. However, this could be extremely expensive at time and space. For all the aggregated level at level 2, we have 16 million records to generate and save in the spatial data base, it costs almost one month to generate these data points at average throughput rate. Note that the data samples are provided by the trash disposal department, the data sparsity can be exploited to store these records. We then introduce a vacancy label when generating these aggregated levels to indicate that if it is necessary

to do the following queries. In other words, every time we want to generate sub-regions from the last level division, we check this label first to confirm that there are data points in this area. Accordingly, query for aggregated counts can be boosted at time cost because only these areas that have data points will be returned by the spatial query function. The sample structure is depicted in Figure 2.2 This design can help build more efficient queries, consume less database storage and provide more scalability when creating aggregate tables.

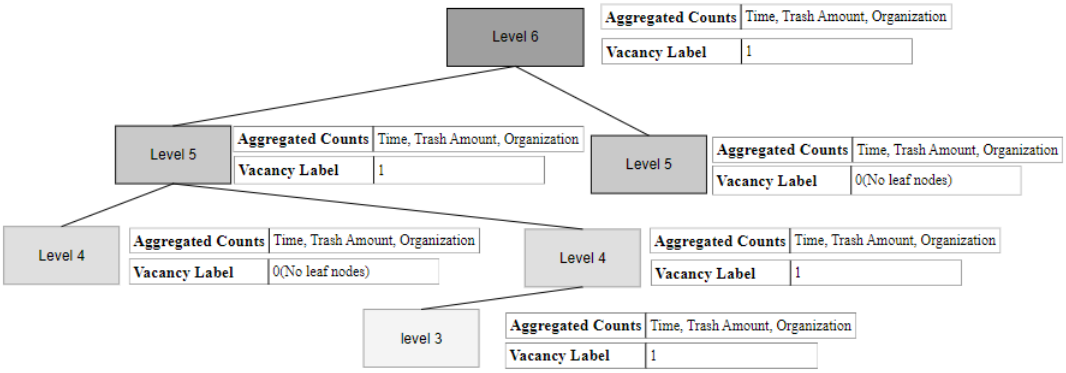


Figure 2.2: Spatial level relation

2.2.4 Query processing

The query processor receives queries from frontend applications and accesses the database to answer them. Incoming queries are three types. First, a spatial query that finds all data points in a certain spatial range. This query is answered from maintable with a single spatial range query. Second, a spatial query that finds aggregate counts in a certain spatial range. This query is answered from aggregatetable with a single spatial range query as well. Third, a query that finds all attributes of a single data record. This

query is answered from the corresponding dataset table with a single query given the record id. Using maintable and aggregatetable enables each incoming query to be translated into a single SQL query, which allow scalable management and visualization of litter data.

2.2.5 Query strategies and empirical results

As aforementioned, we have mainly 3 query types that requires the management system to process.The first primary query for the litter data is to search all points contained in a given area. Spatial query is enabled through spatial queries in *PostGis*. However, a typical query from the original table is always necessary when we only hold some indexing and spatial information in the *maintable* and need more details. This corresponds to the last query that fetches original data. It is evident that the third query is based on the first one, so we have designed three types of query experiments to corroborate that the *maintable* can enormously boost the first and the third query.

- Query 1:Use the spatial function to query in the *maintable*, then accordingly query from each original table with returned table name and record id from the aggregated one, we name this as "spatial-id" strategy.
- Query 2:Use the spatial function to query in the *maintable*, get distinct table names from spatial function and query each original table using the same spatial function, we name this as "saptial-spatial".
- Query 3:Directly query each original table using the spatial function, without exploiting the aggregated one. We name this as "spatial".

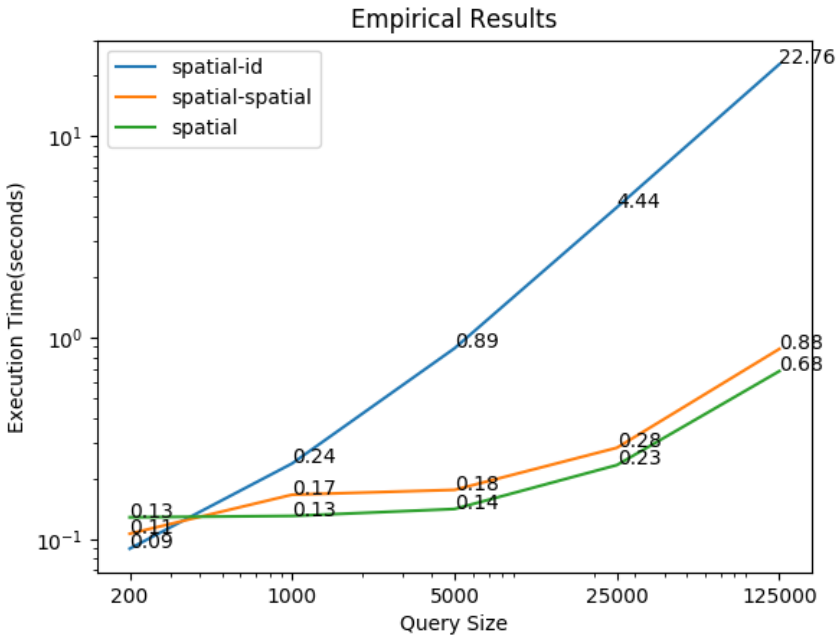


Figure 2.3: Execution time for 3 queries strategies.

Experiment

We use 5 different sizes of query to conduct the experiment in 15 original tables and 1 aggregated table. Python APIs are used for generating test range, connecting to the database, timer recording and figure plotting. The experiment machine is an HP laptop with I7 6500U processor, 8GB Ram, ubuntu 16.04 installed. The python version is 3.5.6.

Result

Figure 2.3 shows execution time of all three strategies in 5 different query size. These are tentative due to some limitations. Firstly, we only experimented on 15 tables, this leads to better performance for the 3rd query. When query tables increase, it should

have longer execution time due to useless queries. In addition, this indicates that when we have small number of tables, the most straightforward query should be the outstanding one. The spatial range should also be more general. In this tentative experiment, we only use one polygon for each size. However, it can be seen that the *maintable* design can help build more efficient queries and thus saves much more time at orders of magnitude than the ordinary case that we do not exploit any optimization with *maintable* on these datasets. However, the "spatial-spatial" could be more efficient when more and more datasets are incorporated, as the "spatial" would produce many useless database communications when data sets are area dependent.

2.3 Frontend Design and Implementation

Building on top of CleanUpOurWorld backend, frontend applications consume the litter data through different interfaces. Interactive visualization of litter data, adding and extracting data, the system administration are respectively demonstrated in the following sub-sections.

2.3.1 Interactive Spatial Visualization of Litter Data

CleanUpOurWorld interactively visualize litter data on multiple spatial levels and for different types of litter. Figure 2.4 and Figure 2.5 depict the main visualization screen of CleanUpOurWorld. Figure 2.4 depicts the highest spatial level of visualization. At this level, the whole world is divided into ten large regions based on continents and their adjacent oceans, eight of them are shown in the Figure 2.4. The Figure 2.4 shows the

spatial boundaries of each region. In addition, each region has a single pie chart that shows percentages of each litter type, out of ten types depicted in the legend. On hovering over the pie chart, the actual number of points of each litter type is displayed. In this visualization screen, a scientist can interactively change the data view by filtering the visualized data based on spatial level, litter type (plastic, glass, metal, wood, etc), temporal period, and collecting organization. The scientist can zoom in and out on the map view to show finer granular data on six different levels: continent, sub-continent, country, subcountry, city, and street level. By zooming in/out on the map view, the data is automatically divided or merged to show data that corresponds to the current level. The continent level divides the whole world into ten regions, the sub-continent level shows data from multiple countries in one spatial tile and divides the whole world into 10x10 tiles, and so on up to the street level. Figure 2.5 depicts the street level visualization in Riverside, California. At this level, only individual data points are shown without any aggregation. The viewed data subset is reflecting the applied filters of litter type, time period, and collecting organizations.

2.3.2 Adding New Data Source

The on-going data collection process and the natural continuity of human litter necessitates adding new datasets to our data store. A simple way is adding new data manually through the administrative tools, e.g., PostGIS admin tools. However, this will limit both number of collaborations and amount of data that contribute to our system. To make it a comprehensive research database that serves as many users worldwide as possible, we are enabling external data entry so that scientists and organizations can contribute their

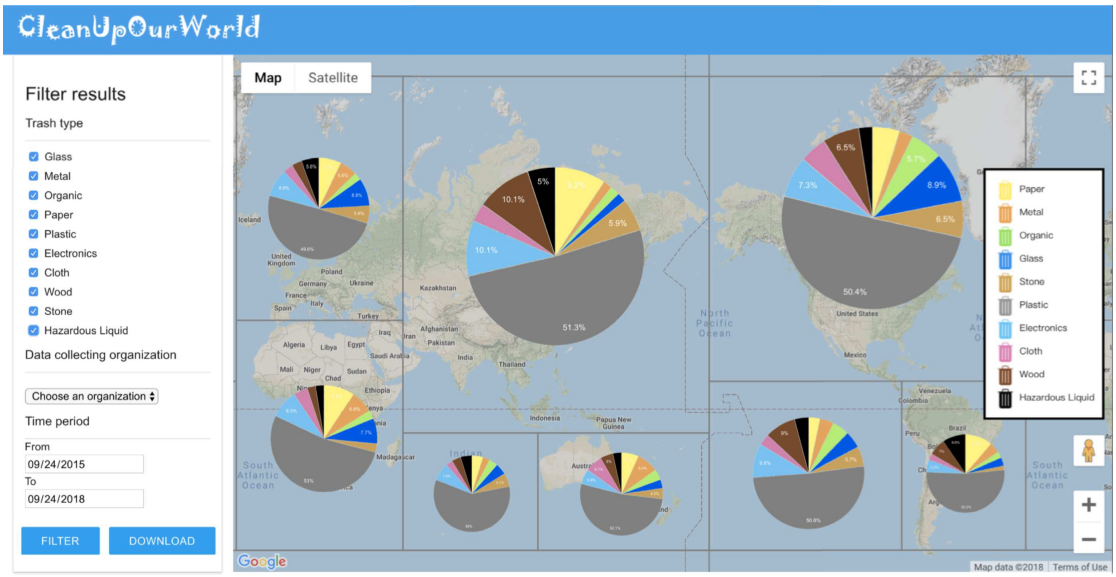


Figure 2.4: Litter Data on Continent Level

data to our repository easily. Figure 2.6 depicts a data entry screen that enables uploading external data to CleanUpOurWorld. The screen allows the user to upload a file of a certain format. Currently supported formats are ESRI Shape files, KML files, XLSX files, and CSV files. By default, the first data row is considered attributes names. After the file is successfully parsed and loaded, the user can rename attribute names and determine their data types. In addition, she can edit any data value in any row so she can possibly add any missing values or correct any incorrect values. Once all corrections are made, the user can submit the uploaded data to our backend database.

The newly uploaded data is not directly integrated with our existing data. Instead, the new data is loaded in a new separate database table and the system administrators are automatically notified with a new dataset addition request. Then, the request is reviewed for data adequacy in terms of completeness of necessary attributes, matching attribute

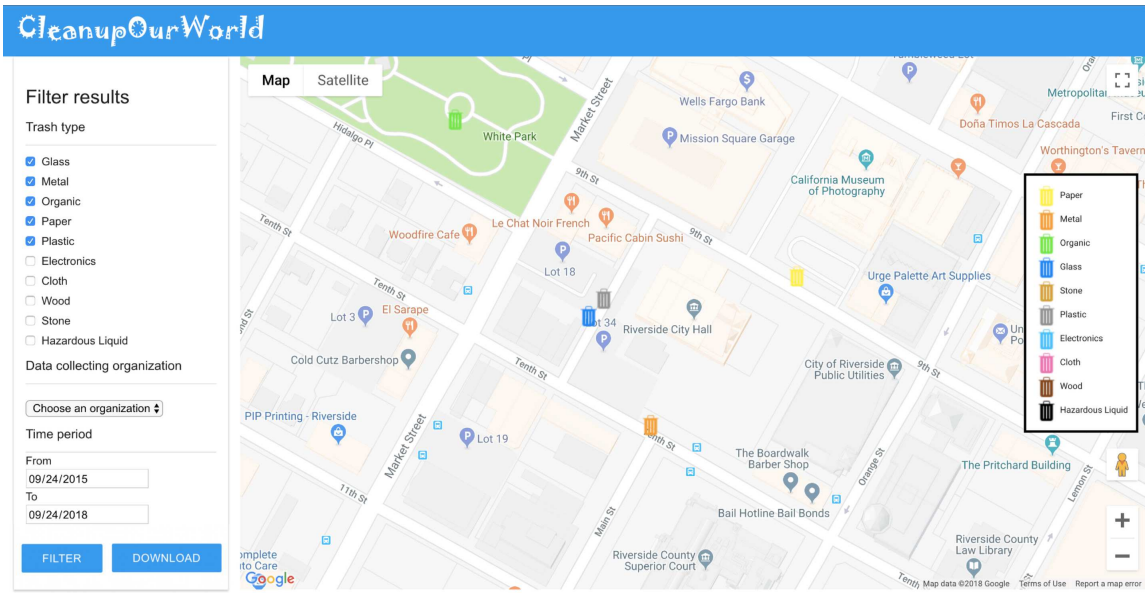
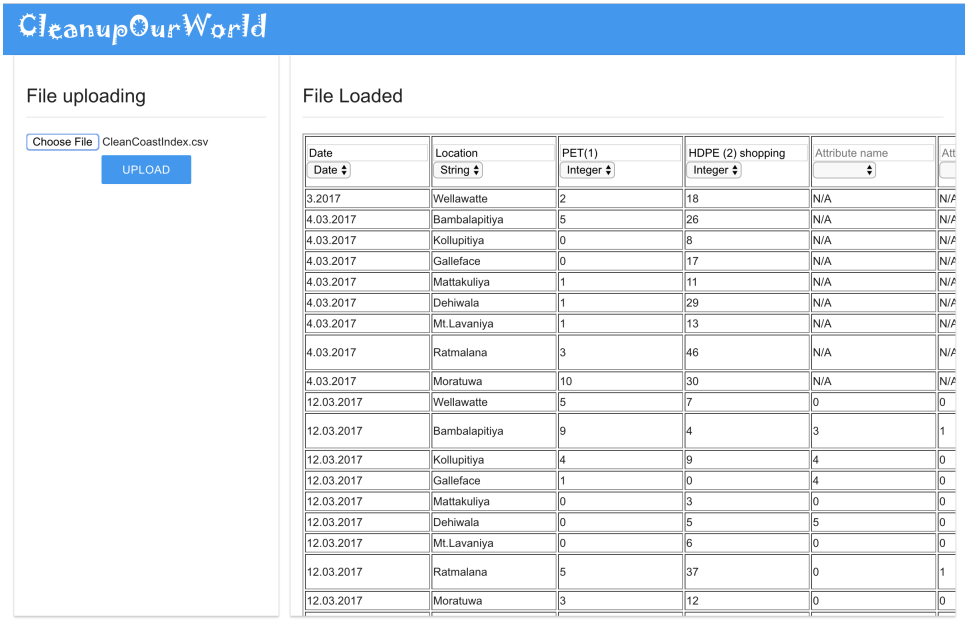


Figure 2.5: Litter Data on Street Level

names, and lack of any data problems such as SQL injection data, severe noise data, etc. In several cases, it is needed to follow up with the data owners to fix data problems before allowing the new data to be integrated with the existing data. Once the uploaded data is put into a good shape to be integrated, the cleaning, integration, and loading process of CleanUpOurWorld is executed so the new data is processed as part of our database. Although this process is lengthy and might include several cycles of interactions between database administrators and data owners, it still allows much faster process than individual collaborations as uploading and refining the data is much faster.

2.3.3 Extracting Data

CleanUpOurWorld will have its prominent value with enabling scientists to use it as data repository where they can add and get data in addition to being a scalable data



Date	Location	PET(1)	HDPE (2) shopping	Attribute name	Att
Date	String	Integer	Integer		
3.2017	Wellawatte	2	18	N/A	N/A
4.03.2017	Bambalapitiya	5	26	N/A	N/A
4.03.2017	Kollupitiya	0	8	N/A	N/A
4.03.2017	Galleface	0	17	N/A	N/A
4.03.2017	Mattakuliya	1	11	N/A	N/A
4.03.2017	Dehiwala	1	29	N/A	N/A
4.03.2017	Mt.Lavaniya	1	13	N/A	N/A
4.03.2017	Ratmalana	3	46	N/A	N/A
4.03.2017	Moratuwa	10	30	N/A	N/A
12.03.2017	Wellawatte	5	7	0	0
12.03.2017	Bambalapitiya	9	4	3	1
12.03.2017	Kollupitiya	4	9	4	0
12.03.2017	Galleface	1	0	4	0
12.03.2017	Mattakuliya	0	3	0	0
12.03.2017	Dehiwala	0	5	5	0
12.03.2017	Mt.Lavaniya	0	6	0	0
12.03.2017	Ratmalana	5	37	0	1
12.03.2017	Moratuwa	3	12	0	0

Figure 2.6: Adding litter data to the management system

manager and visualizer that track litter data. For this, we will enable users to download subsets of data based on terms and conditions of each dataset. Figure 2.4 shows a download button that enables users to extract the subset of data that is currently visualized on the map view with all the applied filters. The extracted data will include any aggregate data that is shown on the map view, in addition to any individual data points, contributing to these aggregates, that are permitted by the data owners to be downloaded for public. Additional custom data extraction options are being added to the system. This includes targeting certain datasets, regions, or types of litter.

2.3.4 Administrating Data

Despite the automated tasks provided by CleanUpOurWorld in maintaining litter data, database administrators may still need to manually fix data issues, e.g. clean unknown

formatted attributes or tweak any dataset-specific problem. Thus, system administrators have direct access to the backend database via the administration console through either command line or GUI interfaces. This includes executing SQL queries of differ- ent types. Examples are selection queries to view existing data records, update statements to modify certain data values, alter table statements to modify data scheme properties (changing attributes names, data types, adding attributes, etc). In addition, the system administrators develop dynamic SQL functions that are used in several operations such as data loading, part of the cleaning process, etc.

Chapter 3

Spatial Data Regionalization

3.1 Background

For more comprehensive understanding of the problem and the mechanism of the parallel computing that might be exploited in the context, we present the background of the the problem definition, the literature review and the parallelism architecture that could facilitate the local search phase.

3.1.1 Problem Definition

The mathematical formulation has been defined in Duque's work [7] precisely, in this section we do not give the same mixed integer programming model as we will not conduct the experiments based on some conventional methods for solving integer programming(such as CPLEX), the heuristics is the option that is much less computationally expensive for general datasets.The following notation and description gives problem definition with more readability.

Key Notations

- Areas. The basic unit before aggregation process, initially we have a set $\boldsymbol{A}$ of areas with a total number of $\boldsymbol{n}$.

$$\boldsymbol{A} = \{A_1, A_2, ..., A_n\}, |\boldsymbol{A}| = \boldsymbol{n}$$

- Attribute. In max-p-regions problem, we have two major attributes to be considered, one is the dissimilarity measure of the every area, denoted by $\boldsymbol{D}$, and the other is the threshold attribute, denoted by $\boldsymbol{T}$. $\boldsymbol{D}_i$ means the attribute of i-th area, the same rule applies to $\boldsymbol{T}_i$.

- Spatial contiguity. As all the partitions have to be spatially contiguous in one region, we use graph $\boldsymbol{W} = (\boldsymbol{V}, \boldsymbol{E})$ to represent the adjacent relations of all the areas.Edge v_i, v_j exists if and only A_i and A_i are adjacent spatially. The vertices correspond to the areas.

- Partition. We use R_p to denote one aggregated region made up of some areas that are connected while p represents the index of the region. A feasible partition is a set

$$\boldsymbol{P} = \{R_1, R_2, ..., R_p\}, |\boldsymbol{P}| = p, 1 \leq p \leq n$$

  while

$$\bigcup_{i=1}^{p} R_i = \boldsymbol{A}, |R_i| > 0$$

Constraints

- Contiguity. In one feasible partition R_p,$W(R_p)$ must be a connected graph, meaning that any two areas must share a common border.

- Threshold constraint. In one feasible partition, for every region R, given a reasonable threshold value s(can not exceed the total threshold attribute of all areas)

$$\sum_{A_i \in R_k} T_i \geq s, k = 1, 2, ..., p$$

Optimization Criteria

The local search phase is responsible for optimizing the homogeneity, we use $H(\boldsymbol{P})$ to represent the heterogeneity of a feasible solution and $h(\boldsymbol{R})$ to represent the heterogeneity of one region, we use $d_{ij} = |\boldsymbol{T}_i - \boldsymbol{T}_j|$ to represent a pairwise dissimilarity between two areas. Then

$$h(\boldsymbol{R}) = \sum_{ij:A_i,A_j \in R} d_{ij}, \boldsymbol{R} \in \boldsymbol{P}$$

and

$$H(\boldsymbol{P}) = \sum_{\boldsymbol{R} \in \boldsymbol{P}} h(\boldsymbol{R}),$$

Then using Π as the set of all feasible solutions, the max-p-regions problem can be viewed

$$p = max\{p|p = |\boldsymbol{P}|, \boldsymbol{P} \in \Pi\}$$

and after maximizing p

$$H(\boldsymbol{P}) = min\{H(\boldsymbol{P})|\boldsymbol{P} \in \Pi\}$$

3.1.2 Literature Review

Regionalization

While looking back all the research efforts that have been made to explore this topic, many solution strategies have been proposed on traditional aggregating homogeneous regions. In [24] [25], they apply a conventional clustering algorithm to get clusters of these areas, regardless of the spatial contiguity. While clusters have formed, the algorithm finds possible areas that have already been assigned to the same cluster, thus forming regions. The major limitation of this method is the clusters might have produce regions that do not hold highly adjacent areas inside, which requires further optimization. While in [32] [20] [33], as the well-developed clustering algorithms have promising performance, two additional coordinates are included as additional attribute for the clustering algorithm. This method has taken account the spatial contiguity but in a weighted manner. However, how to resolve the weight tradeoff between the original attributes and spatial attributes would be another problem.

In Duque's work [7], the contiguity condition is no longer treated as posteriori. Multiple strategies for ensuring spatial contiguity are investigated in this paper, in which the seeded region strategy is selected to be adapted to the problem . In our work, the scheme is inherited in the same manner, while in specific solution the seeding strategy is modified to get larger number of initial regions.

While the spatial contiguous property has been guaranteed by endogenizing the number of regions to maximal, solving such a combinatorial problem of aggregating areas into regions can not be treated as the conventional mixed integer programming as its

computational cost is extremely high when the problem size grows larger . It has been demonstrated [7] that the constraints and the variables are $3n + (n-1)n^2 + n\frac{n^2-n}{2}$ and $(n-1)n^2 + \frac{n^2-n}{2}$ respectively. This means when areas number grows, it is nearly infeasible to get the solution in a very short time using a common integer programming solver. To seek for better solution quality at a reasonable time cost, heuristics are designed and employed to evade local optima, though global optima is not guaranteed in such non-convex optimization. For this problem, two main heuristics have been tentatively tested, which are simulated annealing and tabu search. Simulated annealing is a process that simulates the annealing progress of the metallurgy, which gives a dynamic possibility not to choose the new generated solution in the following search step, while this possibility is decreasing gradually. Compared to the simulated annealing, tabu search needs more time to converge as it forbids the solution to go back to the previous moves to evade local optima, unless this move can generate a best-so-far solution. In our problem, tabu search is capable of locating solution of high quality while converge at a acceptable time cost. This paper mainly focuses on modification of tabu search.

Parallel Heuristics

While heuristics have been studied fairly well, parallelization of them did not draw people's attention until recent years. As multi-core architecture and massive parallel platforms being widely applied in the recent decade, many algorithms [19] have evolved into more advanced variants. Massive data processing systems like Hadoop, Spark and HPCC have apply the concept of parallelism to large scale clusters, leading to huge benefits in computational cost and problem resolution. Fiechter's work [10] has used a parallel tabu

search for addressing TSP problems at scale, the problem itself tries to separate the problem into several sub problems, which means the parallelism is realized by executing tabu moves simultaneously while limiting the move possibilities between different solutions. Another tabu search parallelism reflected on [3], their work was designed in a solution-exchange manner, while every process is a sequential version of the algorithm and every single pipeline is parameterized differently for receiving and transmitting for solution quality enhancement. Vehicle routing has also benefited from parallel structures [13] [2]. However, most of them were either executing in one processor sequentially or conducting less communications(Only in the beginning and at the end) between these processors to avoid computational cost, none of them has proposed a typical strategy to enlarge the search space while improving the solution quality. Our paper proposed a novel tabu search algorithm that can be either implemented in multi-core architectures or massive parallel clusters with faster processing speed and larger data throughput at scale.

3.1.3 Parallelism Architecture

According to the state-of-the-art parallel processing architecture or systems, parallel computing mechanism can be classified from multi-core and multi-processors within a single machine to cluters and grids utilizing multiple computers executing the same task.

In a single personal computer, parallelism can be achieved by multi-threads or multi-process. The first one is light weight, all threads in one thread can share the resource of a process that has been allocated by the operating system. In fact, threads are just sequences of executions of process. As setting multi-thread parameters, multiple threads can be assigned to multiple cores or processing units to perform tasks. The communication

cost is relatively light compared to process level parallelism, as all tasks happen in one process. Besides, it is less robust as the resources are shared among threads. In contrast, process is the basic unit for resource allocation, meaning that when executing parallel tasks at multi-process level, sound isolation is guaranteed among these processes, which also leads to heavy communication cost when synchronization is required. This can be verified observably especially in Windows operating system. However, the parallelism at this level holds more robustness in contrast to thread level.

In massive computing clusters, multiple computers are connected through switches and routers, providing both physical and virtual parallelism compared to single personal computer. Customized massive data processing systems like Hadoop [13], Spark [36] and BigQuery [30] are installed on these machines to deal with typical tasks. In this paper, all experiments are done in single PC to make the parallelism exploited at process level. .

3.2 Algorithm Design and Implementation

In this section we will demonstrate the design and implementation of our two-phased based algorithm. The first phase design considerably increases the initial number of regions since it provides denser solutions then the original seeding strategy. The second part of this section gives specific algorithmic description of the our parallel tabu search strategy and implementation details. Though the general construction phase and local search phase are inherited from (Duque 2012)'s work, we give a algorithmic description in Algorithm 1. Both phases mentioned in this algorithm are leveraged to demonstrate considerable experimental outcomes, which will be depicted in the experimental results

section. The algorithm first takes the original data as the input for region construction. Then, the solutions generated by the first phase is fed into the second phase as inputs for local search algorithms. In Algorithm 1, the parameter *maxitr* should be pre-defined before running the algorithm in order to get considerable solutions though dissimilarity measure is not considered yet. Besides, *maxp* is also confirmed in the first phase, this satisfies the concept of formatting p-regions problem into max-p-regions problem by endogenizing the number of regions before solution quality enhancement.

3.2.1 Initial solution construction

In this section, we detail the algorithm of the first phase, which is to generate feasible solutions regardless of the dissimilarity. Compared to the baseline algorithm in the paper that defines max-p-regions, this part has considerable improvement in a more dense manner that generates more regions initially, which does not depend on the data as well. In the original solution, random seeds are always generated from all available areas, this randomness is not necessary in terms of solving our problem. In in Algorithm 2, only one random seed is generated , the area index is re-organized so as to be traversed not randomly but by the original index order, from the start of the random seed area. Then every seed is followed by the index as long as it is not assigned to a region. For instance, in a area set that forms a square, this would grow regions from the top-left corner in a left-to-right, top-down manner, if the most top left corner is selected as the seed. For irregular areas, this would also work as long as the original index of the areas are maintained in some certain order. The following two subsections will give details of growing clusters given one seed

Algorithm 1 Parallel Tabu Search For Max-P-problems

procedure MXP($\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{D}, \boldsymbol{W}$)

$maxp = 0, , minh = +\infty, \Pi = \emptyset$ ▷ Π represents the solution set

while $itr \neq maxitr$ **do** ▷ Construction Phase

$\boldsymbol{P} = \boldsymbol{RegionConstruction}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{D}, \boldsymbol{W}, threshold)$

if $maxp < |\boldsymbol{P}|$ **then**

$maxp = |\boldsymbol{P}|$

$\Pi = \boldsymbol{P}$

else

if $maxp = |\boldsymbol{P}|$ **then**

$\Pi = \boldsymbol{P} \cup \Pi$

for $\boldsymbol{P}_{maxp} \in \Pi$ **do** ▷ Local Search Phase

$\boldsymbol{P}^{new} = \boldsymbol{ParallelTabuSearch}(P_{maxp}, procNum, tabulength)$

if $H(\boldsymbol{P}^{new}) < minh$ **then**

$minh = H(\boldsymbol{P}^{new})$

$\boldsymbol{P}^{optimal} = \boldsymbol{P}^{new}$

return $\boldsymbol{P}^{optimal}$

area and assigning enclaves given partially completed regions.

Grow Regions

This section will introduce when a seed area is given, how can we generate a region, which can be viewed as a function defined in Region Construction. Given a seed area, we use the graph $\boldsymbol{W}$ to determine all the neighbors not assigned to any other regions. If the area itself already exceeds the threshold, then it is regarded as one region with only one area. If the threshold of assigned areas cannot exceed the required threshold, the neighbors of the seed's neighbors are added to the neighbor list, this also ensures the density from the inside out. The algorithm is demonstrated in Algorithm 3.

Assign Enclaves

An enclave is created after one area has no other neighbors to be assigned to itself while its neighbors have all formed regions, or some regions cannot satisfy the threshold. Both algorithms for constructing regions in our paper and the baseline requires all areas regionalized, which means no enclaves can be left unattended. However, due to the seeding strategy, enclaves are relevantly less in our case. When assigning these enclaves to neighbor regions, the dissimilarity measure is considered in one region-based manner, which is a greedy strategy oriented scheme to form a complete solution, though it cannot be proved optimal. Since we focus on ensuring the spatial contiguity during this phase, the solution quality should be considered more in the second phase.

Algorithm 2 Region Construction

procedure RegionConstruction($\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{D}, \boldsymbol{W}, threshold$)

$labels = 0, |labels| = |\boldsymbol{A}|, enclave = \emptyset$

$rseed = random(0, |\boldsymbol{A}| - 1)$ ▷ select random seed from the index of areas

$array = concatenate(rseed : |\boldsymbol{A}| - 1, 0 : rseed - 1)$ ▷ Generate new index array

for $index \in array$ **do**

if $labels[index] \neq 0$ **then**

continue

if $W.neighbors[index] \leq 0$ **then**

labels[index] = -1

else

$labels, \boldsymbol{R_{index}} = \boldsymbol{GrowClustersForOneArea}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{W}, threshold, index)$

if $h(R_{index}) < threshold$ **then**

enclave.append($areas \in R_{index}$) ▷ Regions can not exceed the threshold

else

$\boldsymbol{P}^{feasible} \cup R_{index}$

for $label_{index} \in |labels|$ **do**

if $labels[label_{index}] == -1$ **then**

enclave.append($\boldsymbol{A}[label_index]$) ▷ Add islands to enclaves

for $area \in enclave$ **do**

assign area to neighbor regions according to $argminH(P^{feasible)})$

return $\boldsymbol{P}^{feasible}$

Algorithm 3 Grow Cluster For One Area

procedure GROWCLUSTERSFORONEAREA($\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{W}, threshold, index$)

$Neighbors = \boldsymbol{W}(index).neighbors$

$thres_{index} = \boldsymbol{T}(index)$

$labels[index]$ =new assigned region Id

$\boldsymbol{R}_{index} = \boldsymbol{A}(index)$

for $neighbor \in Neighbors$ **do**

if $thres_{index} \geq threshold$ **then**

break

if $labels[neighbor] == 0$ **then**

$labels[neighbor] = labels[index]$

$\boldsymbol{R}_{index} = \boldsymbol{R}_{index} \cup A(neighbor)$

if $h(\boldsymbol{R}_{index}) < threshold$ **then**

$Neighbors = Neighbors + \boldsymbol{W}(neighbor).neighbors$

return $labels, \boldsymbol{R}_{index}$

3.2.2 Parallelized Tabu search

In this section, a novel local search algorithm based on tabu search is proposed. While improving the dissimilarity measure, such non-convex optimization problems fall into local optima when performing some non-heuristic methods, this gives high convergence speed as well as relatively poor solution quality. However, there is no guarantee that heuristics might produce better solutions if run the algorithm in any specific settings. How to explore feasible solution space as large as possible be taken account of in this case. In the next two subsections we introduce tabu search algorithm and how we parallelize it.

Tabu Search

Tabu search is the main part of our local search algorithm as it gives considerable solution quality improvement than other heuristics. The algorithmic description is proposed in the Algorithm 4. First of all, tabu list is a FIFO(first in first out) queue, with a user defined length, given a initial solution. We give one area of regions to its neighbor to generate more feasible solutions rather than the first one. This could yield more options to explore in the search space, in contrast to exchanging areas between neighbors or other methods. When the best option of all feasible solutions is located, the move from the current solution to this candidate is checked in the tabu list, if found, then check it with the best so far solution to see if there is any improvement, this is the only way that a move in tabu list can be made, which is also called aspiration criterion.If it does not exceed the best-so-far solution, then this move is forbidden. If not found in the tabu list, make the move and update current solution and best-so-far solution when necessary. When the

best-so-far solution has not improved in some user defined number of iterations, the tabu search will stop executing.

Parallelism Exploitation

As the tabu search tries to evade the local optimal based on forbidding moves in its tabu list, a shared tabu list can be used to ensure multiple execution program running in such a manner. As the parallel computing solutions are advanced, conducting local search parallelly can also be exploited to enlarge the search space. There are 3 possible parallelism that can be leveraged in our tabu search phase. The first one can be intuitive, while we give the same amount of solutions to the search algorithm as the number of tasks that can be executed concurrently, these generated solutions do not have any relations prior to local search, which might result in unstable results, since some solutions at first are not promising. The second one is to build relations between these initial solution that are going to be fed into the local search algorithm. We start from one first initial solution, and then generate corresponding amount of solutions that can exploit the parallelism to the extent of the computational limit. These solution have strong relations with slight difference made by the local search algorithm, the way to ensure that they do not make moves in a same manner is to make them have shared tabu list. This list will store all the moves that have been made to perform local search tasks. Meanwhile, the tabu moves they will make in the next step is also sent to some shared space to ensure that the moves would not overlap.To ensure the solution quality compared to the sequential one, there could be one solution with priority to select moves, meaning that when it has contradictory moves with others, it decides with priority. The third parallelism is quite similar to the second one, except that the possible

moves are compared globally, thus generating next possible solutions globally, note that the second type of parallelism only considers solution quality in the current executing thread or process when selecting the next possible move. Both the second and the third requires synchronization, however the communication cost for the third one is much higher than the second one while not generating promising results in a guaranteed manner. In our case, the second scheme is preferable.

The algorithmic description is in Algorithm 5. The initial solutions are equal to the number of currently parallelizable threads or processes, while they are generated by the same starting solution by running tabu search $procNum$ times(The $procNum$ is empirically set to 5 in our following experiment). This generation scheme could build relations between relations, it to some extent guarantees the improvement of solution quality from one start point compared to feeding the parallel pipeline with 5 irrelevant solutions and no tabulist is shared(first type of parallelism). While the computing power on the experimental machine should be exclusively reserved for processing the tabu search phase parallelly, the usual total work (number of operations) is set less than the available cores at maximum for flexible parallelism. We use Figure 3.1 to demonstrate the working flow and the parallel architecture of our algorithm. The shared tabu list is responsible for checking tabu moves and their commitment when move choices have been made. Besides this, another variable that requires synchronization is the global best solution, which is used to determine the stop criteria. In traditional tabu search, this is also used to check for aspiration criteria, however, this does not apply in the parallelized version. Each process has its own version of best so far solution, so as to determine the this key characteristic in tabu search. The

autonomous copy is always less than or equal to the global copy, while this gives search flexibility to those solutions that are not promising initially. The algorithm will be stopped once there is no improvement after some iterations of moves among all processes.

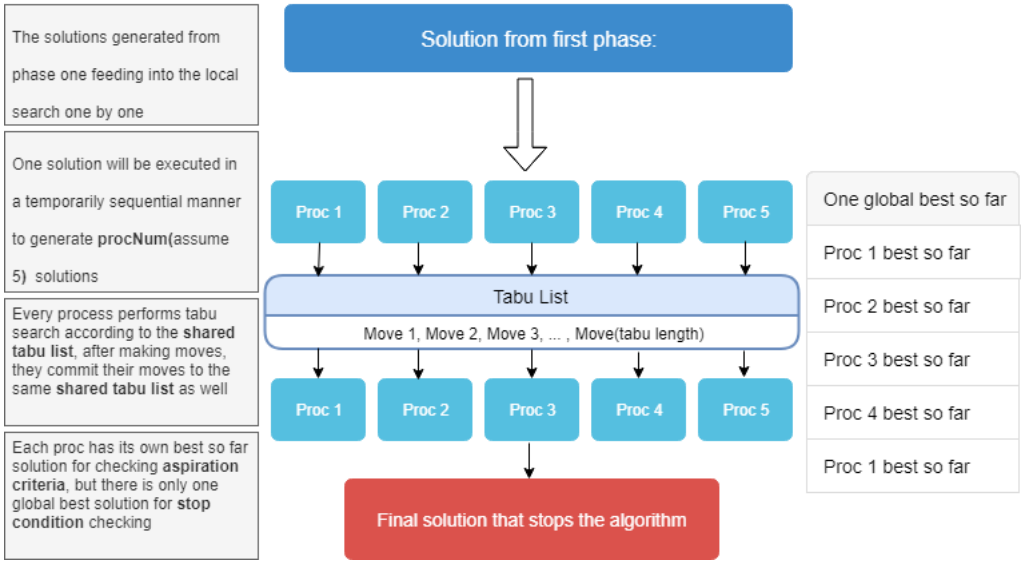


Figure 3.1: Parallel Working Flow Architecture

Algorithm 4 Tabu Search

procedure TABUSEARCH($\boldsymbol{P}, tabulength, tabulist, stopitr$)

$\boldsymbol{P}_{best} = \boldsymbol{P}_{current} = \boldsymbol{P}$

$maxitr = 0$ ▷ Max iterations that objective is not improved

while $maxitr < stopitr$ **do**

$\Pi_{Neighbors} = \boldsymbol{P}_{current}.NeighborSolution$

if $|\Pi_{Neighbors}| = 0$ **then**

break

Sort $\Pi_{Neighbors}$ according to $H(\boldsymbol{P}), \boldsymbol{P} \in \Pi_{Neighbors}$ in ascending order

for $\boldsymbol{P}_{new} \in \Pi_{Neighbors}$ **do**

if *move* (from $\boldsymbol{P}_{current}$ to $\boldsymbol{P}_{new}$) $\in tabulist$ **then**

if $H(\boldsymbol{P}_{new}) < H(\boldsymbol{P}_{best})$ **then** ▷ Aspiration Criterion

$\boldsymbol{P}_{best} = \boldsymbol{P}_{new}$

$\boldsymbol{P}_{current} = \boldsymbol{P}_{new}$

maxitr = 0

tabulist.add(*move*)

break

else

$\boldsymbol{P}_{current} = \boldsymbol{P}_{new}$

maxitr = maxitr + 1

if $H(\boldsymbol{P}_{new}) < H(\boldsymbol{P}_{best})$ **then**

$\boldsymbol{P}_{best} = \boldsymbol{P}_{new}$

maxitr = 0

tabulist.add(*move*)

break

return P_{best}

Algorithm 5 Parallel Tabu Search

procedure PARALLELTABUSEARCH($\boldsymbol{P}, procNum, tabulength$)

Shared tabulist = [] ▷ Tabu list is shared by all sub-processes

$\Pi_{initial} = \boldsymbol{TabuSearch}(\boldsymbol{P}, tabulength, tabulist, stopitr)$ ▷ Make procNum moves

for Every sub-process in a total number of $procNum$ **do**

Execute $\boldsymbol{TabuSearch}(\boldsymbol{P}_{sub}, tabulength, tabulist, stopitr), \boldsymbol{P}_{sub} \in \Pi_{initial}$

Commit to tabulist

Break when converged ▷ Tabu list is shared here

return P_{best}

3.3 Experiment

3.3.1 Experimental Setup

The experiment is conducted in a linux based environment, the version number is *Ubuntu*18.04, the hardware includes Intel I7 6700HQ CPU with 4 cores and 8 threads and 16GB memory. The parallel processes(*procNum* is empirically set to 5) Table 3.1 outlines details of data utilized in our experiments. Our experimental datasets incorporates both machine generated and real-world data based on some social statistics in the U.S.

Table 3.1: Dataset information

Dataset Name	Dataset Size	Dataset source	Dataset shape
n100	100 areas	Machine simulated	Square
n529	529 areas	Machine simulated	Square
n1024	1024 areas	Machine simulated	Square
n2025	2025 areas	Machine simulated	Square
CA_Polygons	58 areas	Real-world Data	Irregular

3.3.2 Computational Results

The following tables gives experimental results of our algorithms and the baseline from the aforementioned paper [7]. Table 3.2 gives comparison between the baseline method(Base-construct) and our method(Dense-construct) after 10 times of execution. The threshold for the machine simulated data is set to 100, while the real-world data has the threshold of population, which is set to 1 million for the data in 2002 in California. It can be seen that from every dataset, the dense construct has considerable improvement in terms of region number in the construction phase. In "CA_Polygons", there are some areas that are irregular. For example, in San Francisco Bay Area and Great Los Angeles Area, the population density stays extremely high compared to other counties. So initial region generation is quite limited by these big blocks. In other datasets, the elevation of the initial region number has been depicted as conspicuous increase, while making sound preparations for local search quality enhancement.

Table 3.2: Region generation(TH = 100 or 1,000,000)

Algorithm,Dataset	Execution Count									
	1	2	3	4	5	6	7	8	9	10
Base,n100	8	9	8	10	9	9	9	8	8	9
Dense,n100	11	10	11	9	11	11	10	11	11	10
Base,n529	50	49	49	50	49	50	51	50	52	51
Dense,n529	58	59	59	60	59	61	59	58	60	58
Base,n1024	98	99	96	97	97	99	96	97	96	96
Dense,n1024	114	116	114	116	114	113	115	118	113	115
Base,n2025	195	196	198	195	200	201	200	197	195	198
Dense,n2025	230	230	234	229	232	231	230	230	229	231
Base,CA_Polygons	14	15	14	14	15	14	15	15	15	15
Dense,CA_Polygons	13	16	14	16	13	14	16	15	14	13

The following tables and figures give quantitative results in terms of execution time , number of solutions from first phase and heterogeneity after local search.For instance, Table 3.3 depicts the number of first phase solutions being fed to the local search phase, while in some cases the total running time might be much longer when making comparison, it could result from different number of solutions that are fed to the second phase, thus making it perform the local search algorithm in times directly proportional to the number of initial solutions. As such, Table 3.4 and the following tables present the execution time at different scale while in proportion to the number of initial solutions. In each table, we name these two algorithms as "Base-Tabu" and "Dense-Parallel" for the baseline algorithm and our algorithm respectively. The number after "tabu" indicates the length of tabu list. Three types of threshold value are set for the experiment, while we also empirically choose 3 kinds of threshold for dataset "CA_POLYGONS", Table 3.9 and Table 3.10 show the data for this real-world dataset.

Table 3.3: Number of First Phase Solutions(TH = 100)

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
n100	4	5	1	1	3	2
n529	1	1	1	5	1	4
n1024	2	1	1	1	3	1
n2025	1	1	1	3	1	1

Table 3.4: Average Execution time(seconds) per solution(TH = 100)

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
n100	5.32	5.17	6.45	4.76	5.68	8.87
n529	80.54	95.03	82.81	146.54	132.21	144.99
n1024	200.16	202.42	218.09	484.07	459.32	459.29
n2025	721.12	765.72	788.46	1793.01	1715.55	1761.03

Table 3.5: Number of First Phase Solutions(TH=300)

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
n100	6	7	6	1	3	2
n529	2	2	1	3	3	2
n1024	1	3	3	3	1	2
n2025	2	1	1	2	2	1

Table 3.6: Average Execution time(seconds) per solution (TH=300)

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
n100	4.28	3.66	3.90	3.51	5.20	4.64
n529	90.79	89.70	101.39	110.95	111.96	128.48
n1024	421.17	254.21	236.98	481.88	651.77	455.42
n2025	934.54	1056.96	984.90	1840.79	1925.21	1941.08

Table 3.7: Number of First Phase Solutions(TH=500)

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
n100	1	3	2	10	10	10
n529	1	3	1	7	2	1
n1024	4	2	1	5	6	1
n2025	1	1	1	1	4	2

Table 3.8: Average Execution time(seconds) per solution(TH=500)

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
n100	18.86	9.21	11.61	3.25	3.41	3.93
n529	160.36	113.64	208.05	111.62	102.70	95.14
n1024	284.27	315.20	380.71	416.78	415.18	423.74
n2025	1117.22	1133.97	1208.60	1657.75	1741.65	1828.98

Table 3.9: Number of First Phase Solutions for CA_POLYGONS

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
TH = 500,000	4	1	4	5	6	1
TH = 1,000,000	1	1	8	1	2	4
TH = 2,000,000	3	3	4	7	5	7

Table 3.10: Average Execution time(seconds) per solution for CA_POLYGONS

Dataset Name	Base-Tabu			Dense-Parallel		
	tabu-10	tabu-24	tabu-85	tabu-10	tabu-24	tabu-85
TH = 500,000	3.97	2.49	6.11	4.10	5.36	3.32
TH = 1,000,000	4.29	3.95	5.75	3.92	4.22	4.56
TH = 2,000,000	7.86	8.79	5.61	6.23	6.95	5.62

Heterogeneity reduction is the objective of the second phase, we then present all the computational results from Figure 3.2 to Figure 3.6, the baseline algorithm is represented by "Base" followed by a tabu list length, which also applies to our case in terms of tabu length, however, the name is abbreviated as "Pa".

3.3.3 Quantitative analysis

The experimental result show considerable improvement in heterogeneity while maintaining acceptable time cost. As the initial solution generation incorporates some randomness from the seeding strategy, it is more reasonable to combine the first and the

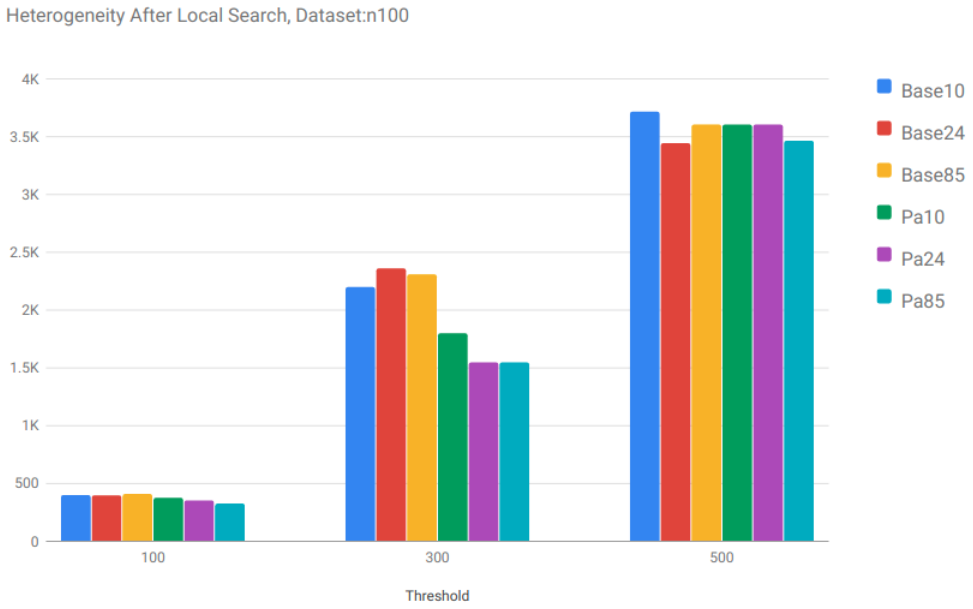


Figure 3.2: Heterogeneity after local search, dataset:n100

second phase every time the algorithm is executed. While looking at average time per solution costs, it is evident that the time cost for our algorithm stays at the same order of the magnitude, while in some cases the average time for running one solution is even much less than the baseline algorithms(for instance,dataset:n100,TH=500). It can be concluded that in most cases the time cost of our algorithm is at the same order of magnitude compared to the baseline, ranging from nearly identical to less than 2 times of baseline . Note that it is unfair to compare the time that digests different amount of first phase solutions as every solution will be executed by the second local search phase. It is also noteworthy that when the dataset is very small(n100 and CA_POLYGONS), the improvement on both phases are not promising as others, which results from the fact that the actual search space is quite limited, in this case even some greedy or simpler search strategies can outperform the meta-heuristic approaches. There is another characteristic in the experiment that can

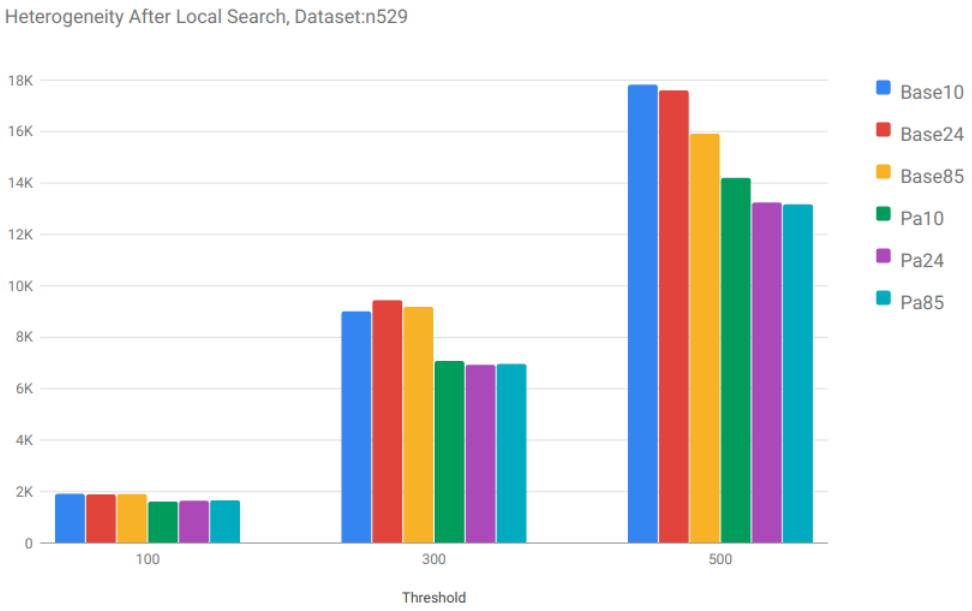


Figure 3.3: Heterogeneity after local search, dataset:n529

arouse research interest, the pairwise dissimilarity generated by our algorithm is usually a little bit higher before the local search phase, while we know from simple inference that in the end larger number of regions leads to less heterogeneity, this can cost the algorithm more time to improve the solution at the second phase.

In the heterogeneity histogram, there is a steady reduction when comparing the baseline to our algorithm, when the tabu list is identically set. Even we neglect the tabu list length, the heterogeneity reduction is still evident in these regular datasets including "n529", "n1024" and "n2025". The other two datasets has much less areas, which leads to search space limitation as aforementioned. Besides, it is also noticeable that if the threshold value is set improperly, there could be two cases that have effect on the algorithm performance. In one way, if the threshold is too small, then many areas will form one region, which makes the aggregation process meaningless. In the other way, suppose the threshold value is set

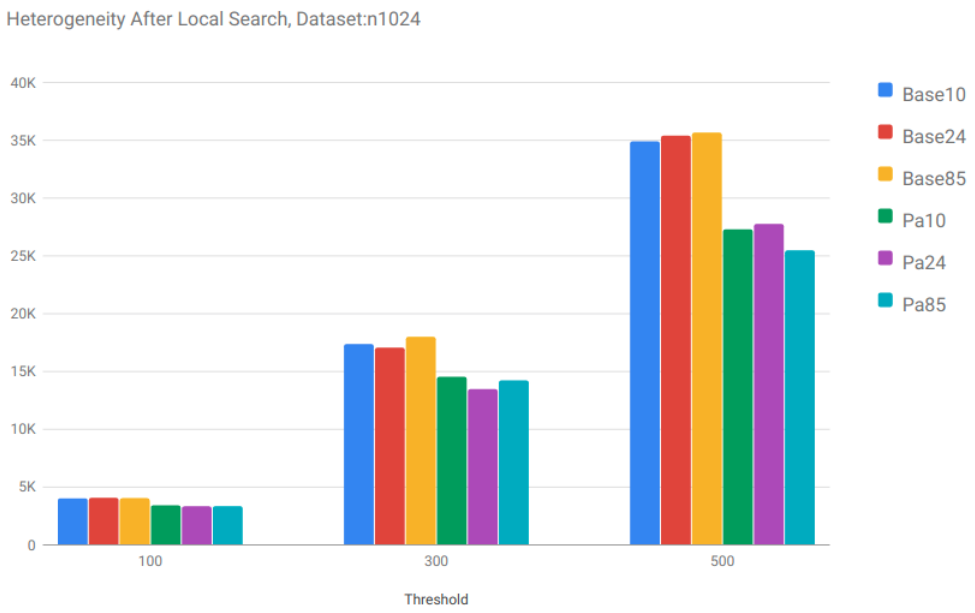


Figure 3.4: Heterogeneity after local search, dataset:n1024

to a large value, then for those datasets that have less areas, the possible region generation is very limited thus making the search process futile.We can see that in figure 3.2 setting threshold to 500 in dataset "n100" is not a fair option. The data irregularity in real-world is also capable of causing such problems, super cities like Los Angeles and San Francisco will probably form one single region as its population surpasses other counties enormously.

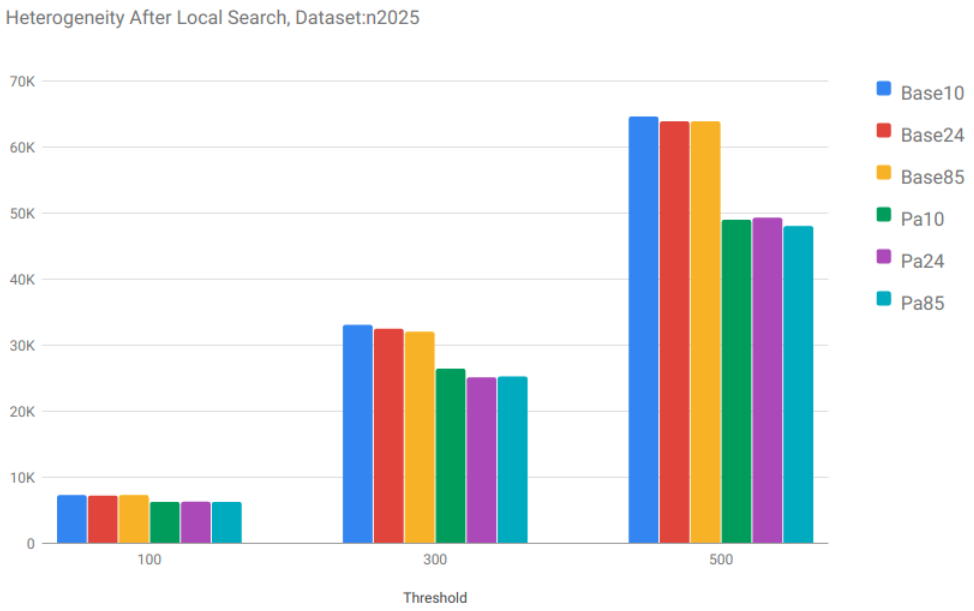


Figure 3.5: Heterogeneity after local search, dataset:n2025

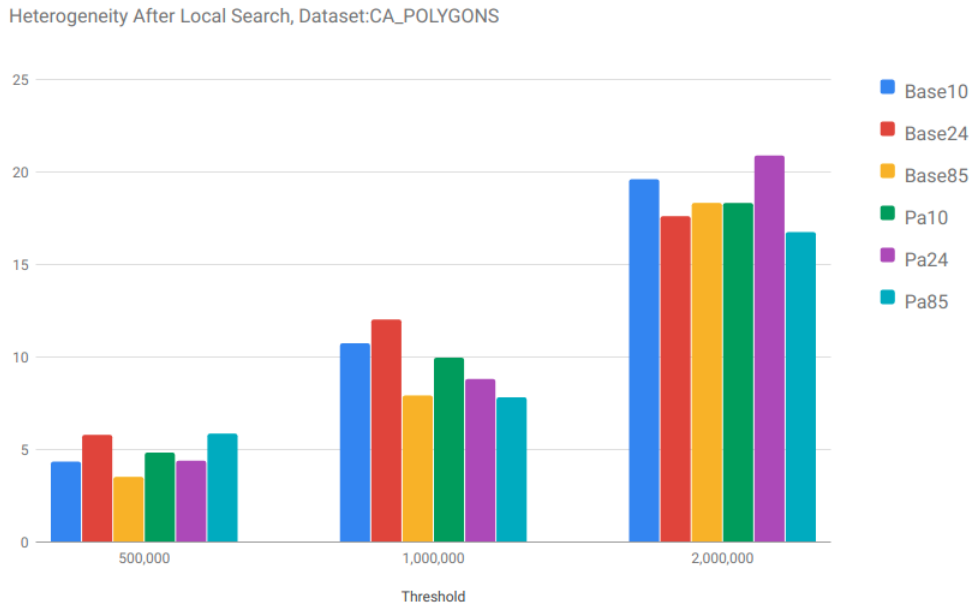


Figure 3.6: Heterogeneity after local search, dataset:CA_POLYGONS

Chapter 4

Conclusion and Future Work

In this thesis, two problems have been proposed spanning the range from spatial data management to spatial data regionalization analysis. Accordingly, solutions of certain quality are demonstrated in a systematic approach, including incorporating various systems and parallel algorithmic exploitation.

The first part of the thesis introduced CleanUpOurWorld: a scalable data management framework that enables collecting and tracking human litter data from different data sources. The framework enables its users to track litter data of different types around the world, which is a necessary step to address the existing environmental problems and life quality problems in large population cities. The framework enables scientists and organizations to add new data and extract existing data based on different data filters that include spatial locations, collecting organization, litter types, and temporal horizons. The added data are formatted, cleaned, and integrated with the existing backend to enable large-scale visualization that is limited in existing technologies, e.g., GIS software. The efficient access

and visualization of this data enables scientific modeling of the data as well as supporting the community efforts to address litter data problems through smart technologies. There are some observations that can be tentatively drawn from our frontend application. For instance, litter data mainly lie around the seashores in each country that has data record. Meanwhile, the trash types are mainly plastics from these records that have specific type information. There are some future works that can be extended based on CleanUpOur-World. The aggregation is still limited to 6 levels, while more levels of aggregation might be demanded, developing dynamic frameworks to collect aggregated information at real time might provide more insights for environmental scientists. The frontend performance is limited due to some garbage collection mechanism, optimization on such applications can be exploited. While in the backend, more automated scripts should be utilized to facilitate the process of importing and organizing the data.

In the regionalization portion of the thesis, a novel algorithm has been proposed to address the max-p-regions problem based on some baseline work. It inherits the framework that is proposed in the baseline paper, while it also leverages a more dense construction inspired by DB-scan and parallelism in a meta-heuristic approach to comprehensively ameliorate the solution quality as well as maintaining the time cost at the same acceptable rate. The heterogeneity acquires a steady reduction based on our "dense-parallel" approach, resulting from the facilitation of both phases. There are also some potential works that can be further studied. One is to investigate the specific impact of our algorithm phase by phase,meaning that the influence that each phase has made on time and heterogeneity could be investigated, this gives insights when these two approaches might be applied to

other methods respectively. Besides, the real-world data "US_COUNTY" is not well experimented due to the time schedule of this thesis, this can make up the space that this thesis has left on real-world data analysis, as it incorporates 3k areas and might provide more promising experimental results compared to "CA_POLYGONS". The global interpreter lock of the legacy code we utilized limits us to use multi-process techniques based on python to exploit parallelism, however it is not the best option to use such a script language for parallelism exploitation because of the communication and implementation mechanism, reconstructing the algorithm based on a parallelism-friendly language should be another direction for reducing time cost. Further, the correlations between the performance and the parameters such as tabu list length, threshold and dataset size can be explored to make our approach better applied to the real applications of this topic.

Bibliography

[1] Luc Anselin. How (not) to lie with spatial statistics. *American journal of preventive medicine*, 30(2):S3–S6, 2006.

[2] Philippe Badeau, François Guertin, Michel Gendreau, Jean-Yves Potvin, and Eric Taillard. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 5(2):109–122, 1997.

[3] Andreas Bortfeldt, Hermann Gehring, and Daniel Mack. A parallel tabu search algorithm for solving the container loading problem. *Parallel computing*, 29(5):641–662, 2003.

[4] Andrew D Cliff, Peter Haggett, J Keith Ord, Keith A Bassett, Richard Davies, Karey L Bassett, et al. *Elements of Spatial Structure: A Quantative Approach*, volume 6. Cambridge University Press, 1975.

[5] Ocean Conservancy. https://oceanconservancy.org/, 2019.

[6] Lets do it! World. https://www.letsdoitworld.org/, 2018.

[7] Juan C Duque, Luc Anselin, and Sergio J Rey. The max-p-regions problem. *Journal of Regional Science*, 52(3):397–419, 2012.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[9] Waste Exposure and Skin Diseases. https://www.ncbi.nlm.nih.gov/pubmed/28209048, 2017.

[10] C-N Fiechter. A parallel tabu search algorithm for large traveling salesman problems. *Discrete Applied Mathematics*, 51(3):243–267, 1994.

[11] Manfred M Fischer. Regional taxonomy: a comparison of some hierarchic and non-hierarchic strategies. *Regional Science and Urban Economics*, 10(4):503–537, 1980.

[12] Alice Ferguson Foundation. Trash free initiative; alice ferguson foundation. http://fergusonfoundation.org/trash-free-potomac-watershed-initiative/, Mar 2019.

[13] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel computing*, 27(12):1641–1653, 2001.

[14] AD Gordon. A survey of constrained classification. *Computational Statistics & Data Analysis*, 21(1):17–29, 1996.

[15] Chenjuan Guo, Bin Yang, Ove Andersen, Christian S Jensen, and Kristian Torp. Ecosky: Reducing vehicular environmental impact through eco-routing. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1412–1415. IEEE, 2015.

[16] Plastics in the Ocean Affecting Human Health. https://serc.carleton.edu/NAGTWorkshops/health/case_studies/plastics.html, 2012.

[17] Joseph JáJá. *An introduction to parallel algorithms*, volume 17. Addison-Wesley Reading, 1992.

[18] M Keane. The size of the region-building problem. *Environment and Planning A*, 7(5):575–577, 1975.

[19] F Thomson Leighton. *Introduction to parallel algorithms and architectures: Arrays· trees· hypercubes*. Elsevier, 2014.

[20] Alan T Murray and Tung-Kai Shyy. Integrating attribute and space characteristics in choropleth display and spatial data mining. *International Journal of Geographical Information Science*, 14(7):649–667, 2000.

[21] Fionn Murtagh. A survey of algorithms for contiguity-constrained clustering and related problems. *The computer journal*, 28(1):82–88, 1985.

[22] US National Oceanic and Atmospheric Administration (NOAA). https://www.noaa.gov/, 2019.

[23] Impacts of Mismanaged Trash. Impacts of mismanaged trash — trash-free waters — us epa. https://www.epa.gov/trash-free-waters/impacts-mismanaged-trash, Mar 2019.

[24] Stan Openshaw. A regionalisation program for large data sets. *Computer Applications*, 3(4):136–147, 1973.

[25] Stan Openshaw and Liang Rao. Algorithms for reengineering 1991 census geography. *Environment and planning A*, 27(3):425–446, 1995.

[26] Lawmakers reauthorize NOAA Marine Debris Program. https://www.americangeosciences.org/policy/news-brief/lawmakers-reauthorize-noaa-marine-debris-program, 2018.

[27] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[28] Full of Disease: The Worlds Open Dumps Are Growing. Smelly, Contaminated. https://www.theguardian.com/global-development/2014/oct/06/smelly-contaminated-disease-worlds-open-dumps, 2014.

[29] Ocean Trash Isnt Just Bad For the Environment Its Bad for Your State of Mind. https://www.washingtonpost.com/news/energy-environment/wp/2015/07/08/why-clean-beaches-are-so-important-if-you-want-a-relaxing-vacation/, 2017.

[30] Jordan Tigani and Siddartha Naidu. *Google BigQuery Analytics*. John Wiley & Sons, 2014.

[31] Marine Debris Tracker. http://www.marinedebris.engr.uga.edu/, 2019.

[32] R Webster and Peter A Burrough. Computer-based soil mapping of small areas from sample data: Ii. classification smoothing. *Journal of Soil Science*, 23(2):222–234, 1972.

[33] Steve Wise, Robert Haining, and Jingsheng Ma. Regionalisation tools for the exploratory spatial analysis of health data. In *Recent developments in spatial analysis*, pages 83–100. Springer, 1997.

[34] Litterati A Litter Free World. https://www.litterati.org/, 2019.

[35] Marine Litter A Growing Threat Worldwide. https://www.eea.europa.eu/highlights/marine-litter-2013-a-growing, 2017.

[36] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.