PIMS Jobs Committee

$Math^{Industry}$

2020 Final Report

PREFACE

The inaugural $Math^{Industry}$ ("Math to power industry") event took place virtually during August, 2020. The workshop is the flagship event of the $Math^{Industry}$ program, which aims to facilitate meaningful scientific interactions between academic and industry in the mathematical sciences.

The Pacific Institute for the Mathematical Sciences (PIMS) and partners offered the $Math^{Industry}$ workshop in 2020 as a rapid response program to the economic impacts of the COVID-19 pandemic. The workshop trained young mathematical scientists for jobs in important industry sectors in western Canada. The program began with a training bootcamp (software best practices, business communications, project management) and included group collaborations with industry partners. This volume represents the immediate results obtained by those groups. Effective business decision-making requires expertise in modelling, computation, statistics, optimization and other areas of the mathematical sciences. This volume contains 10 reports demonstrating how mathematics and statistics can be used effectively. During the workshop, teams developed tools which can be used to automatically detect when parking lots have available parking spaces, create optimal work schedules for employees, and analyze how making changes to a cylindrical heater may alter the heater's performance. Some teams analyzed housing price data to determine what features most impact real estate markets, or developed pricing models for oil which take into account constraints due to congestion in the networks which deliver oil and the recent trends of oil prices to fall below zero. One team used data to approximate the relative amounts of two different radon isotopes present in samples, despite the difficulties encountered in taking precise measurements of these samples. Another team found ways to make blockchain technology more efficient by reducing the size of the computation required to make a transaction, and another developed metrics to measure the performance of politicians. All of these projects were motivated by an industry partner's desire to advance their technology, perform more efficiently, or commercialize their ideas in development. In short, these teams provided business solutions which are benefitting Canada's economy.

The $Math^{Industry}$ workshop was made possible by the hard work of the organizing committee, the industry partners who provided business problems and mentorship, and the academic mentors. We acknowledge their participation and contributions. We would also like to thank Mitacs for their training and internship partnerships and Dhavide Arulliah of QuanSight for supporting the teams.

Acknowledgements

We gratefully acknowledge these people for providing mentorship or other support to $Math^{Industry}$ teams. Their particular contributions are acknowledged in the reports.

Cüneyt Gürcan Akçora, University of Manitoba
Yakine Bahri, University of Victoria
Laleh Behjat, University of Calgary
Brian Bullas, Aerium Analytics
Scott Dalton, Ovintiv
Dave Dong, BC Financial Services Authority
Mina Enayat
Matthew Greenberg, University of Calgary
Heather Hardeman-Vooys, Aerium Analytics
Kai Kaletsch, Environmental Instruments Canada Inc.
Kui Pan, ATCO
Ying Ying Liu, University of Manitoba
German Luna Pattiarroy, The Divi Project
Matt McDonald, Fotech Solutions
Daniel McReynolds, Aerium Analytics
Nisha Mohan, ATCO
Firas Moosvi, University of British Columbia
Reza Peyghami, York University
Vakhtang Putkaradze, ATCO and University of Alberta
Edwin Reid, McMillan-McGee Corp.
Nima Safaian, Cenovus Energy
William Spat, IOTO International
Anatoly Swischuk, University of Calgary
Justin Tendeck, IOTO International
Parimala Thulusiraman, University of Manitoba
Edward J. Timko, University of Manitoba
Shawn Wang, University of British Columbia, Okanagan
Wenning Wei, University of Calgary

ORGANIZING COMMITTEE

Ian Allison, PIMS
Kristine Bauer, University of Calgary
James Colliander, UBC and PIMS
Megan Dewar, Tutte Institute & Carleton University
Brittney Durston, PIMS
Jolen Galaugher, University of Manitoba
Allen Herman, University of Regina
Michael Lamoureux, University of Calgary
Matt McDonald, Fotech Solutions
Anthony Quas, University of Victoria
Ruth Situma, PIMS
Raymond Spiteri, University of Saskatchewan

INSTRUCTORS

Ian Allison, PIMS
Mary Baetz, Mitacs, Baetz consulting services
Kristine Bauer, U. Calgary
Ananya Chattoraj, U. Calgary
France Cloutier, Mitacs, Clarrus/EM Sciences
Diane Fletcher, Mitacs, VADIS Consulting Group & Clarrus/EM Sciences
Jolen Galaugher, U. Manitoba
Lynne Lamarche, Mitacs
Leonard Olien, U. Calgary
Liam Wrubleski, Wrubleski consulting services
Marc Wrubleski, Wrubleski consulting services

PARTICIPANT LIST

Wali Mohammad Abdullah
Nishant Agrawal
Mahsa Azizi
Yakine Bahri
Dana Berman
Noah Bolohan
Brian Chan
Erik Chan
Carlos Contreras
Guillermo Martinez Dibene
Daniel Di Benedetto
Xilai Fu
Leimin Gao
Jillian Glassett
Chantelle Hanratty
Yiwei Huang
Anton Iatcenko
Keran Li
Danyi Liu
Emily Rose Korfanty
Ivan Lau
Shang Li
Benjamin MacAdam
Evan MacNeil
Adili Masanika
Alexandra McSween
Sarah Nataj
Jaeun Park
Boya Peng
Ana Roldan-Contreras
Mohsen Seifi
Neha Sharma
Abishek Kumar Shukla
Elham Soufiani
Amelia Spivak
Stephen Styles
Diaaeldin Taha
Ryan Theissen
Edward Timko
S. Parisa Torabi
Yi Sui
Dongying Wang
Li Wang
Wenning Wei
Liam Wrubleski
Yanhong Xu
Yao Yao
Tingzhou Yu
Jianou Zhang
Aaron (Xiang) Zheng
Junjie Zhu

CONTENTS

OPTIMIZATION OF EMPLOYEE WORKING SCHEDULE AMID COVID-19

A. MASANIKA[1], A. SPIVAK[2], C. HANRATTY[3], W. M. ABDULLAH[4]

ABSTRACT. We report on a two-week COVID-19 motivated project to create employee schedules aimed at limiting the percentage of the workforce in the workplace at any time while keeping associated commute time to a minimum. The project was structured as an optimization problem. Genetic algorithms were explored, adapted and implemented. Preliminary results suggest this as a promising avenue justifying further efforts.

1. INTRODUCTION

Due to the outbreak of COVID-19 around the world, and government policies implemented as a response to the outbreak, many corporations have chosen to let their employees work from home to prevent the spread of the disease. In order to re-open the economy safely with respect to the disease threat, one of the recommendations from health authorities is to allow only a limited percentage of workers in the workplace at any specific time. Given these constraints, it is useful for companies to arrange flexible work schedules so that the employees are at the workplace during reasonable working hours. With changed schedules and perhaps shorter work shifts come the risk of increased employee commute time or increased ratio of commute time to work shift length. Thus it is desirable that these new schedules be designed to reduce commute time as well. Another objective in reducing commuting time is directly COVID-19 related: less time commuting poses a reduced risk of an employee being exposed to the disease.

In this project, we design a mathematical model to address such a scheduling problem. We implement a genetic algorithm to generate employees' working-at-office schedule for a model business satisfying certain reasonable criteria described in what follows and in the following section.

We fix a workplace, the Salesforce Tower in San Francisco with a variable number of employees. The city is divided into 39 neighborhoods numbered 0-38 and each employee's residence is identified with a number in this interval. Using Uber traffic data for n employees whose neighborhood locations are randomly generated, we seek to reduce commute time while maintaining a fixed range on work shift length and aiming to keep the maximum percentage of employees in the workplace at any time less than 35%.

We organize the rest of this report in the following way. In Section 2, we discuss the mathematical representation of the problem followed by a description of our algorithm to solve the stated problem. Results from numerical experiments on a standard collection of

test instances are provided in Section 3. Conclusions drawn from these preliminary studies follow in Section 4.

2. Mathematical Model and the Genetic Algorithm

The mathematical representation of the problem and the algorithm deployed to solve the problem are discussed in this section.

2.1. **Model.**

This section is devoted to providing a mathematical model of the optimization problem. We used the following parameters in our model.

- The total number of employees $n = 200 \sim 1000$
- Zone of the company S_c
- Employees'addresses $S_e^i (i \leq i \leq n)$
- The maximum percent of workers at the office aimed for is $\alpha = 25 \sim 35\%$
- Working time window $7AM - 9PM$
- Employee's weekly working hours $T = 15 \sim 25$ hours
- Weekdays $D = \{M, T, W, Th, F\}$
- A tuple of schedules for the employees $x = (x_1, \ldots, x_n)$

The objective function is

$$f(x) = f_1(x) \times f_2(x) \tag{2.1}$$

such that

- $f_1(x)$ is the mean daily travel time for the employees in minutes
- $f_2(x)$ is the highest percent of employees in the office at any time

The goal is to minimize $f(x)$. Any slight increase of $f_1(x)$ and/or $f_2(x)$ (i.e., closer to α) will increase the value of $f(x)$. Our goal is to minimize total commute time subject to the constraint that the workforce presence in the office at any time should not exceed 35% of the total number of employees. To this end, we chose our objective function to be $f = f_1 \times f_2$ and we examine the output produced by the algorithm for solutions that meet the desired percentage.

The following criteria were stipulated:

- Each employee must have precisely one shift per day
- No specific working sub-groups that have to be present at a given time (e.g., incorporating meeting schedule)

2.2. **The Algorithm.**

To solve our problem we use a genetic algorithm. Our goal is to minimize the objective function (also known as *fitness*) $f(x)$. We have presented our algorithm (pseudocode) to solve the problem in Algorithm 2.1.

Algorithm 2.1 Our Algorithm (*DataSet*)

1: **Initialization:** Possible solutions are created from the given *DataSet*
2: **Evaluation:** Evaluate an individual using the objective function $f(x)$
3: **loop** <until the algorithm meets the stop criteria>
4: Selection: Pick two individuals as parents
5: Recombination: Apply crossover between parents
6: Mutation: Randomly (15%) change the newly created individual (child)
7: Evaluation: Apply Step 2 (Evaluation) to the parents and the child. If the child is better than any of the parents then replace that parent by the child

2.2.1. *Description.*

- **Initialization:** There are a number of individuals in the initial population. An individual x is a solution or a weekly work at office schedule for all employees. First we set the population size or the number of individuals required in the population. Then our algorithm starts generating individuals. For an individual, the algorithm selects a start time and a shift length for each working day for each employee (see, Figure 3). During initialization, our algorithm evaluates the value of objective function for each individual and finds the *best* individual among initial population.
- **Evaluation:** Now using Equation 2.1, we can calculate the mean daily travel time for the employees in minutes as well as the highest percent of employees in the office at any time. Therefore, we can get the value of the objective function $f(x)$ by multiplying these values.
- **Generation:** The number of iterations allowed in the algorithm is known as generation. On the other hand, we can consider this as the stop criteria. In each generation, our algorithm has the following phases: Recombination, Mutation and Evaluation.
- **Recombination:** In our algorithm, we use crossover as a recombination procedure. This is also known as one of the "mating" processes for two selected parents. First, we pick two individuals A and B randomly from the population. Then the child is produced by taking each employee's schedule from either of its parents in the following manner. Define

$$p(A,B) = \frac{f(A)}{f(A)+f(B)}$$

to be the probability of A "passing on its genes" compared to B, based on relative fitness (which parent has the lowest objective function value). Then each row in $A \times B$ should be coded to have a $p(A,B)$ chance of taking the corresponding row from A and a $1-p(A,B)$ chance of taking the row from B.
- **Mutation:** We allow a 15% chance of mutation for which one row of a matrix takes on a random row within the starting parameters. Once offspring are generated, then we keep the best two out of three of the offspring and two parents.
- **Evaluation and selection:** In this step, the algorithm compares the objective function values of the parents and the child. Let f_A, f_B, f_C denote the value of the objective functions for the parents A, B and the child C respectively. If f_C is less than f_A and/or f_B, then C replaces the parent with larger f value. Again, if C beats

either of its parent, then we compare C with the *best* solution and update the *best* accordingly. Therefore, the population converge to a better solution and the population size remain the same.

- **Output:** The algorithm returns *best* as a solution after meeting the stop criteria.

3. Numerical Testing

In this section, we provide results from numerical experiments on selected test instances.

3.1. **Data Sources.** The data set for the experiments was generated from the Uber Movement website [4]. Our original intention was to consider a city in Canada which would be familiar to the PIMS program participants. Our first assumption was that mean travel time for any trip would not vary much with the day of the work week but rather, significant differences would be found based on the time of the day of travel. We focused on the downloadable CSV files entitled Travel_Times_by_Hour_of_Day (Weekdays Only) since as the name indicates, this is precisely the data we were looking for.

Unfortunately regardless of the city chosen, there were represented only between 0-5 hours of departure for trips between any fixed origin and any fixed destination and those hours were not well chosen to capture commute times. This is reflected in our finding that it appears that for whatever city one searches for on the Uber Movement, regardless of its size and population, and whatever point of departure one chooses in that city, downloaded file, contains exactly 1,048,576 rows or trips.

Because the data was limited, we turned then to the Filtered Data on the website. The Filtered Data does not provide zones but it does include trips to a chosen address from many neighbourhoods, given by name. In particular, we chose San Francisco city in California, USA, where other data, including geographic maps, extensive census information, and information on distances between major sites, were available and could be evaluated for reliability. With the combination of our original Uber data, the neighbourhood names from the filtered data, and information from the 2016 US census, we were able to get for each of the 39 neighborhoods, average travel times for the five different times of the day: AM Peak Hours (7am-10am), Midday (10am-4pm), PM Peak Hours (4pm-7pm), Evening (7pm-12am), Early Morning (12am-7am).

We chose as our office location the single largest office building in San Francisco, the Salesforce Tower (see, Figure 1). We were able to get approximately 75% of the data points this way, i.e., average commute time to and from each of the 39 neighborhoods for each of five periods of the workday as listed in the previous paragraph. The remainder were obtained from the available data by comparison with regard to distances and identifying similar traffic conditions.

3.2. **Input and output formats of our algorithm.** In Figures 2 and 3 each row represents one of the 200 randomly chosen employees. In Figure 2, the entries in the left-hand column identify the neighborhood in which the corresponding employee lives. For example, the

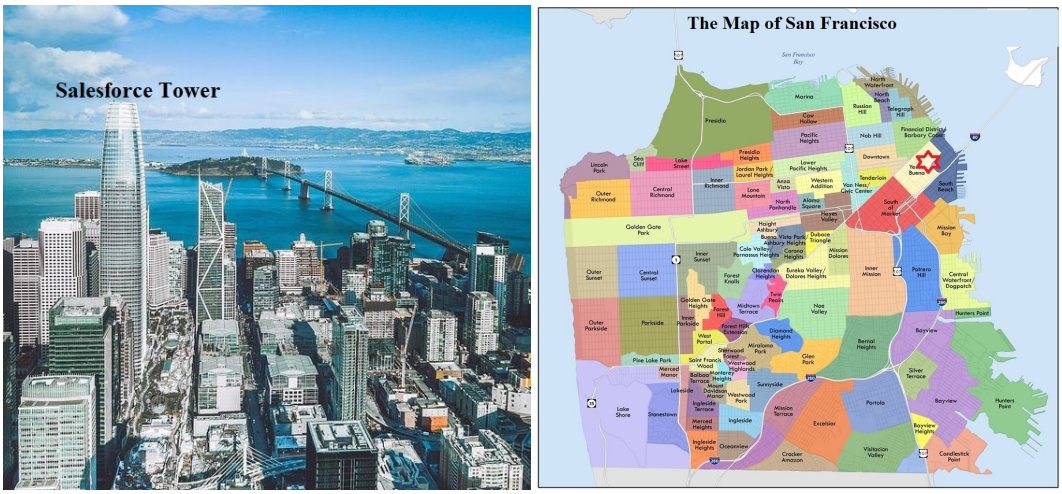


FIGURE 1. Location of the model work place and neighborhood

first employee lives in neighborhood 29, and the entries represent the commute time for each hour in a workday. Figure 3, shows the output of our algorithm indicating the start time and shift length for each employee.

	Mon 07	Mon 08	Mon 09	Mon 10	Mon 11	Mon 12	Mon 13	Mon 14	Mon 15	Mon 16	...	Fri 08	Fri 09	Fri 10	Fri 11	Fri 12	Fri 13	Fri 14	Fri 15	Fri 16	Fri 17
29	16.5	25.0	25.0	25.0	19.7	19.7	19.7	19.7	19.7	19.7	...	25.0	25.0	25.0	19.7	19.7	19.7	19.7	19.7	19.7	22.8
7	23.5	39.9	39.9	39.9	32.2	32.2	32.2	32.2	32.2	32.2	...	39.9	39.9	39.9	32.2	32.2	32.2	32.2	32.2	32.2	38.6
9	11.9	23.5	23.5	23.5	15.0	15.0	15.0	15.0	15.0	15.0	...	23.5	23.5	23.5	15.0	15.0	15.0	15.0	15.0	15.0	17.5
5	3.2	4.8	4.8	4.8	5.8	5.8	5.8	5.8	5.8	5.8	...	4.8	4.8	4.8	5.8	5.8	5.8	5.8	5.8	5.8	6.8
13	22.9	39.6	39.6	39.6	26.1	26.1	26.1	26.1	26.1	26.1	...	39.6	39.6	39.6	26.1	26.1	26.1	26.1	26.1	26.1	33.7

FIGURE 2. The input format

	Monday Start Time	Monday Shift Length	Tuesday Start Time	Tuesday Shift Length	Wednesday Start Time	Wednesday Shift Length	Thursday Start Time	Thursday Shift Length	Friday Start Time	Friday Shift Length
0	17	4	16	5	12	4	14	4	16	3
1	14	4	17	4	11	3	8	5	8	3
2	7	4	10	5	17	3	15	3	10	5
3	11	3	11	3	17	4	10	3	15	5
4	13	3	8	3	11	5	7	5	13	5
...	...	...	...	...	...	...	...	...	...	...
195	17	4	13	5	8	4	15	4	8	4
196	7	3	14	3	7	5	16	5	11	3
197	17	4	15	4	12	3	16	5	7	5
198	17	4	13	3	9	4	9	5	11	4
199	16	4	9	4	9	3	10	5	17	3

FIGURE 3. The output format

Population Size	Generations	Number of Employee	Initial f(x)	Final f(x)
10	200	100	17.35508133	15.199756
20	200	100	16.93616633	15.6158
10	100	100	16.40639	15.63501333
20	100	100	16.52378	15.94443533
10	10	100	16.577708	16.19043533
6	1000	200	17.15139375	16.458522
20	10	100	16.92872733	16.479722
6	25	200	18.14690558	16.9305695
6	100	200	17.31742667	17.0782235
6	20	200	18.330935	17.17866192
6	30	200	17.25992	17.1831715
6	15	200	17.94546767	17.24568025
20	200	50	19.50798667	17.26757333
6	5	200	18.18971975	17.37362075
10	200	50	20.332552	17.39896
50	100	50	19.24222667	17.43018667
6	10	200	17.52234525	17.52045333
200	1000	20	20.79795	17.99886667
20	200	20	20.22783	18.15529
50	10	20	20.2029	18.2314
10	100	50	19.73834133	18.286548
20	10	50	18.508756	18.33804
10	100	20	23.232458	18.659325
50	200	20	21.63375	18.661875
50	100	20	21.721583	18.79447
20	100	50	19.9824	19.03313867
10	10	50	20.07519733	19.29353067
20	10	20	21.68475	19.55602
10	200	20	22.821833	19.64085
20	100	20	22.793	20.04075
10	10	20	22.96775	22.8288

FIGURE 4. Algorithm Results

3.3. **Experimental Environment.** The experiments were performed on PC with 3.4 GHz Intel Xeon CPU, 8 GB RAM running Windows. The implementation language is Python and the code was compiled with a Jupyter Notebook version 6.0.3 compiler. Source code available on GitHub at [1].

3.4. **Results.** After implementing the algorithm in Python, we ran a variety of tests to establish its utility. These tests are recorded in Figure 4 in order of decreasing $f(x)$. Each test was run on a choice of $P =$ population size, $G =$ number of generations, and $E =$ number of employees. We recorded the value of $f(x)$ initially, and after the chosen number of generations.

These tests demonstrated the general trend that $f(x)$ decreases with number of generations, G. Recall that a lower $f(x)$ denotes a stronger schedule. This can be observed in Figure 5 where fix the population size at 6, fix the number of employees at 200, and plot number of generations against $f(x)$.

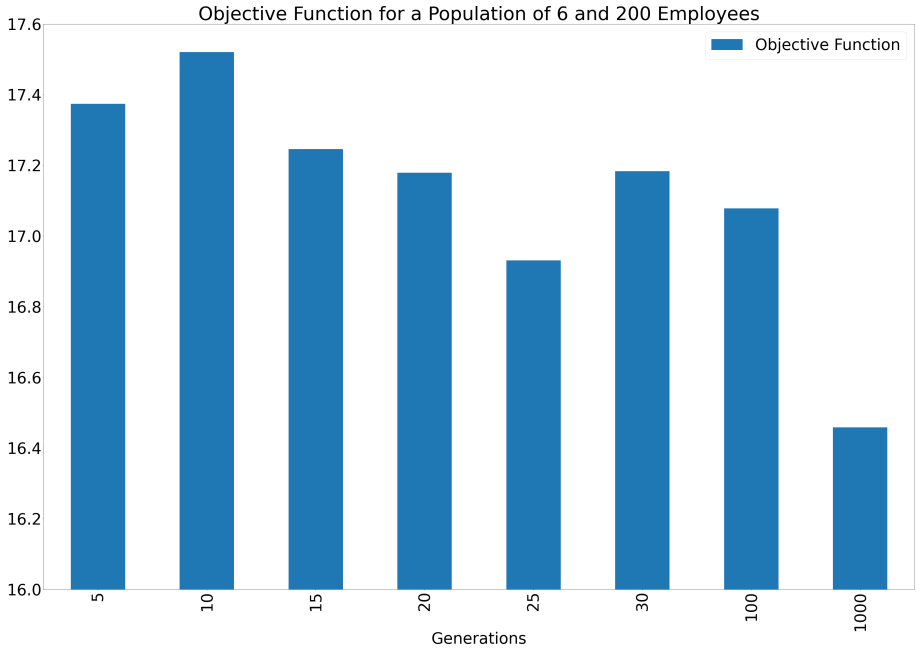
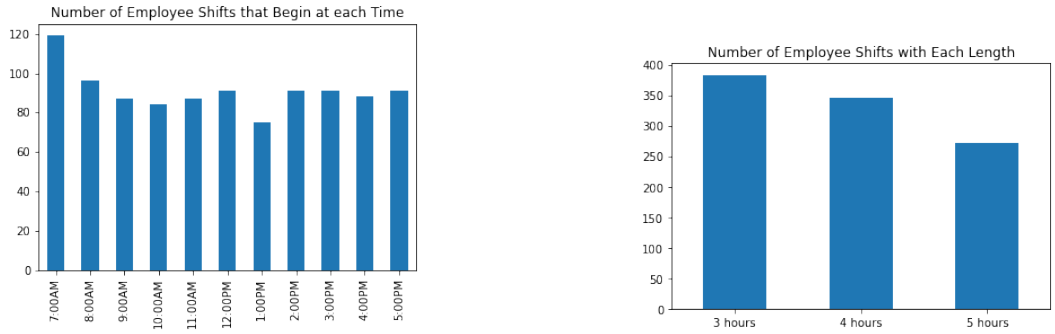


FIGURE 5. Decrease in Objective Function over Many Generations

While there is a general downwards trend, we notice that some higher generation sizes also yield higher $f(x)$ (for instance, when $G = 10, 30$). This perceived discrepancy happens because each test was run on a different initial population. These fluctuations suggest that $G < 100$ is not sufficient to optimize the schedule. However, for $G = 1000$, there is a clear decrease in $f(x)$, compared to the other tested values.

In Figure 6, we take a closer look at the schedule produced after 1000 generations. Figure 6A shows the distribution of start times in this schedule. We can see that the start times are relatively balanced with the exception of the first shift in a day. We posit that this bias lowers the value of f_2. In particular, since only employees who start at 7am can be in the office at 7am, more employees are free to start at this time without negatively effecting the percent of employees in the office.

Figure 6B depicts the distribution of shift lengths in this schedule, and shows that shorter shift lengths are favoured. This will also lower the value of f_2, as shorter shift lengths means an employee will contribute to the percent of employees in the office during fewer time slots.



(A) Start Times (B) Shift Lengths

FIGURE 6. Analysis of a 200 Employee Schedule after 1000 generations.

We conclude our analysis by considering the quality of this 1000 generations schedule. To do this, we compare this schedule to 20 randomly generated schedules, that is 20 schedules who did not undergo optimization.

In a randomly generated schedule, total commute time ranged from 43-44 minutes. After 1000 generations, the total commute time was reduced to 42.6 minutes. In a randomly generated schedule, the maximum percentage of employees in the office at any time ranged from 41 to 47 percent. Recall that the goal is $f_2 < \alpha = 35\%$. After 1000 generations, this was reduced to $f_2 = 38\%$, which is much closer to the target 35%. The test creating the 1000 generation schedule was run over night. We believe that given the time to run more generations, the algorithm could produce an even better schedule.

4. CONCLUSION

The goal of this project was to create a workforce schedule that reduces commute time, and the maximum percent of employees in the office. Such a schedule is difficult to calculate by hand for a workplace with a large number of employees. We modeled this problem to minimize the function $f(x) = f_1(x) \times f_2(x)$ where $f_1(x)$ is mean commute time and $f_2(x)$ is highest percentage of employees in the office. We deployed the genetic algorithm to find a schedule that minimizes this function. After running our algorithm for 200 employees and 1000 generations, we found an improved schedule. This shows that it is possible to automate schedule creation for large offices.

This work is an initial investigation during a two week workshop period. There are more aspects that merit investigation as described below.

Data set: As part of a longer project we would seek out other sources of data in place or to use along with the Uber Movement data.

Initial Population Generation: Currently our algorithm assigns every employee a shift every day. We would like to investigate letting employees have days without shifts, as this could decrease commute time.

The Objective Function: We ran our algorithm with one choice of objective function. We would like to try different objective functions to balance the optimization of f_1 and f_2.

The Mating Procedure: We would like to compare different mating procedures to consider larger or smaller "chromosomes". That is instead of taking entire rows from either parent, it would be interesting to compare taking individual elements from the matrix.

Time: One constraint of the genetic algorithm is its long run time. We ran a variety of tests over night, but we would like to run longer tests with higher populations and number of generations.

Algorithm: In our two week investigation we had time to consider one algorithm. This demonstrated that it is possible to use an algorithmic approach to creating these schedules. However, the genetic algorithm has a long run time so it would be valuable to compare its results to the results of other algorithms.

ACKNOWLEDGEMENT

We thank the organizing committee of the PIMS $Math^{Industry}$ Workshop 2020 for their many valuable training, guidelines and suggestions that helped to complete this project. We also thank Professors Shawn Wang and Mina Enayat who were our academic mentors. Finally, we thank Vakhtang Putkaradze, Kui Pan and Nisha Mohan from ATCO for their continuous support as our mentors from industry.

REFERENCES

1. Team ATCO, *The source code of the implemented genetic algorithm*, `https://github.com/Team-Atco/COVID-Schedule-Optimization`.
2. Jason Hicken, Juan Alonso, and Charbel Farhat, *Introduction to multidisciplinary design optimization*, `http://adl.stanford.edu/aa222/Home.html`, 2012.
3. John H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*, MIT Press, Cambridge, MA, USA, 1992.
4. Uber Technologies Inc, *Uber movement*, `https://movement.uber.com/`.
5. Zulkifli Nopiah, Muhammad Khairir, Shahrum Abdullah, M. Baharin, and A. Arifin, *Time complexity analysis of the genetic algorithm clustering method*, Proceedings of the 9th WSEAS international conference on Signal processing, robotics and automation (2010), 171–176.

[1]UNIVERSITY OF REGINA, CANADA, [2]UNIVERSITY OF CALIFORNIA, BERKELEY, USA, [3]UNIVERSITY OF ALBERTA, CANADA, [4]UNIVERSITY OF LETHBRIDGE, CANADA
E-mail address: [1]ajm192@uregina.ca, [2]ameliaspivak@gmail.com, [3]hanratty@ualberta.ca, [4]w.abdullah@uleth.ca

PRACTICAL OPTION VALUATION WITH NEGATIVE UNDERLYING PRICES

ANA ROLDAN-CONTRERAS, ELHAM SOUFIANI, GUILLERMO MARTINEZ DIBENE, MOHSEN SEIFI, NISHANT AGRAWAL, YAO YAO, AND ANATOLIY SWISHCHUK

Abstract. Here we propose two alternatives to Black 76 to value European option future contracts in which the underlying market prices can be negative or mean reverting. The two proposed models are Ornstein-Uhlenbeck (OU) and continuous time GARCH (generalized autoregressive conditionally heteroscedastic). We then analyse the values and compare them with Black 76, the most commonly used model, when the underlying market prices are positive.

1. Introduction

In March 2020, the prompt month WTI futures contract settled below zero for the first time in the contract's history. Many market participants apply the Black 76 model or some variation when calculating the value of the options on this futures contract as a relatively straightforward, parametric valuation method. This calculation model is hard wired into many Commodity Trading and Risk Management Systems. Traders and risk managers rely on its straightforward and reproducible output.

However, Black 76 requires positive underlying market prices. The negative prompt month settlement price caused considerable consternation among energy traders and risk managers.

More generally, OTC options are also available on basis or differential prices. These transactions are options on the difference between two published indexes such as NYMEX Henry Hub and AECO (for natural gas) or Cushing WTI and Houston (for crude oil). As such, these instruments frequently have negative underlying market prices.

Our task is to propose alternative models to Black 76 to valuate option prices when the underlying future contracts can assume negative values.

2. Definitions

A **primary security** (or securities for short) is any asset that can be traded independently from any other asset, such as stocks. A **derivative security** or (or derivatives) are legal contracts conferring financial rights or obligations upon the holder.

A **forward contract** is an agreement to buy or sell a risky asset (such as crude oil or natural gas) at a determined future date T, known as **delivery**

date, at a specified price K, known as **delivery price**. The price of the asset (or commodity) at time t is known as **forward price** and denoted by $F(t,T)$. Notice $K = F(0,T)$.

Similarly, a **future contract** (or futures for short) involves an underlying asset, which we typically take as a forward contract, and a specified **delivery date** T. A future price set at time t with delivery date T will be denoted as $f(t,T)$.

An **European option** is a derivative security contract that gives the holder the right, but not the obligation to buy or sell the underlying asset, for a price K fixed in advance, known as **exercise** or **strike** price, at a specified future time T_e, known as **exercise** or **expiry date.** An option contract with expiry date T_e stops being valid after this time. The option is known as a **call option** if the holder has the right to *buy* the asset, while a **put option** gives the holder the right to *sell* the asset.

Forwards and futures are legal agreements between two parties giving obligations between them, in contrast, options are legal agreements giving rights to the holder. Because of this advantage intrinsic in options (the holder may trigger the contract should it be in their favour) is that they are to be purchased. We are concerned with valuing them; specifically, we are interested in valuing European call options for futures prices.

3. Proposed alternative models to Black 76.

Black 76 model is obtained from the more general Black-Scholes model (1973). Black-Scholes is a model for the price of a stock at time t and it is given by the following Stochastic Differential Equation (SDE)

$$dS_t = \mu S_t dt + \sigma S_t dW_t,$$

where $0 \leq t \leq T$ represents time (T is the expiry date), $\mu \in \mathbf{R}$ is a number known as the "drift", $\sigma > 0$ is the "volatility" and $(W_t)_{t\geq 0}$ is Wiener process (or Brownian motion). In this model, S_0 is deterministic (not random) and known in advance. Using Itō's formula, it can be deduced that

$$S_t = S_0 e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W_t}.$$

This shows that under the assumptions of Black-Scholes, the stock price will be positive (assuming $S_0 > 0$) for all times.

3.1. Ornstein-Uhlenbeck (Vasicek) model (1930/1977).

The first alternative we propose to Black 76 is given by the following Ornstein-Uhlenbeck SDE

$$dS_t = a(b - S_t)dt + \sigma dW_t, \tag{1}$$

where $a, \sigma > 0$ and $b \in \mathbf{R}$. Here a is known as the "reversion rate", b as the "mean" and σ as the "volatility." Again, using Itō's formula, it can be

shown that the solution to the OU SDE is given by

$$S_t = e^{-at}S_0 + b(1 - e^{-at}) + \sigma e^{-at}\int_0^t e^{as}dW_s. \tag{2}$$

This is a Gaussian random variable with mean $e^{-at}S_0 + b(1 - e^{-at})$ and variance $\sigma^2(1 - e^{-at})/2a$. It is readily seen it can assume negative values and as $t \to \infty$, this Gaussian random variable converges in distribution to a Gaussian with mean b and variance $\sigma^2/2a$, the rate of convergence is given by a. The value of an European call option at time t with delivery date T_e, rate of risk-free investment r, and strike price K is given according to

$$C(F, T_e) = e^{-r(T_e-t)}\left[\xi_+(t, T_e)\Phi\left(\frac{\xi_-(t, T_e)}{\zeta}\right) + \zeta\Phi'\left(\frac{\xi_-(t, T_e)}{\zeta}\right)\right] \tag{3}$$

in which Φ is the distribution function of a standard Gaussian random variable and

$$\xi_\pm(t, T_e) = e^{\pm aT_e}(F(t, T_e) - b) - K$$
$$\zeta = \sigma\sqrt{\frac{1 - e^{-2aT_e}}{2a}}$$

The future prices of this model will be modelled using (1).

3.2. **Continuous Time GARCH model:** Some times the commodity prices exhibit different behavior with respect to time, which is known as Mean-Reversion. It means that, unlike stock prices that tend to change around zero, they tend to return to a non-zero long-term mean. Therefore for a risky asset S_t which has a mean reverting stochastic process, we have the following SDE:

$$dS_t = a(b - S_t)dt + \sigma S_t dW_t \tag{4}$$

where W is a standard wiener process, $\sigma > 0$ is the volatility, the constant $b \in \mathbb{R}$ is the mean reversion level (the long term mean), and $a > 0$ measures the rate (or the strength) of our mean reversion.The closed form of the above equation for a European Call has been provided in section (4).

4. Methodology and results

4.1. **OU model.** According to (1), we need to calibrate the parameters a, b and σ. Using (2) (in which S is substituted for the future price F) it can be seen that observations of the future price are in a linear relation plus normally distributed error terms. As such, least-squares linear regression can be used. In Fig. 3 we do the calibration of the parameters for the OU model using Natural Gas future prices provided by Ovintiv. We can see the prices are around the mean, which is an assumption of validity of the model.

4.2. **WTI Dataset.** For WTI crude oil futures, we compare the option prices calculated by the Black-76 model and the Vasicek model. Let $C(t, T_e)$ be the value for the European call option written on a forward F. Then the Black-76 formula for a European call option price is:

$$C(t, T_e) = e^{-r(T_e - t)}[F(t, T)N(d_1) - KN(d_2)], \tag{5}$$

where $d_{1,2} := \frac{ln(F/K) \pm \frac{1}{2}\sigma^2(T_e - t)}{\sigma\sqrt{T_e - t}}$.
The Vasicek formula for a European call option is similar to equation (6) with slight differences:

$$C(F, T_e) = e^{-r(T_e - t)} \left[\xi_+^*(t, T_e)\Phi\left(\frac{\xi_-^*(t, T_e)}{\zeta}\right) + \zeta\Phi'\left(\frac{\xi_-^*(t, T_e)}{\zeta}\right)\right] \tag{6}$$

in which Φ is the distribution function of a standard Gaussian random variable and

$$\xi_\pm^*(t, T_e) = e^{\pm aT_e}(F(t, T_e) - b^*) - K$$

$$\zeta = \sigma\sqrt{\frac{1 - e^{-2aT_e}}{2a}}$$

where $b^* = b - \lambda\sigma/a, \lambda \in \mathbb{R}$ is a market price of risk.
We use the above formula for Black 76 and Monte Carlo simulation to get the graphs 1 of option prices. Each graph in Figure 1 shows the option prices with different strike price K. Except for the chart with the price date of 2020-04-20, when we have a negative future price, the others are all positive. From the graphs we can see that option prices on futures computed by the Vasicek model are very close to the prices calculated by the Black-76 model when future prices are positive. . When future prices are negative

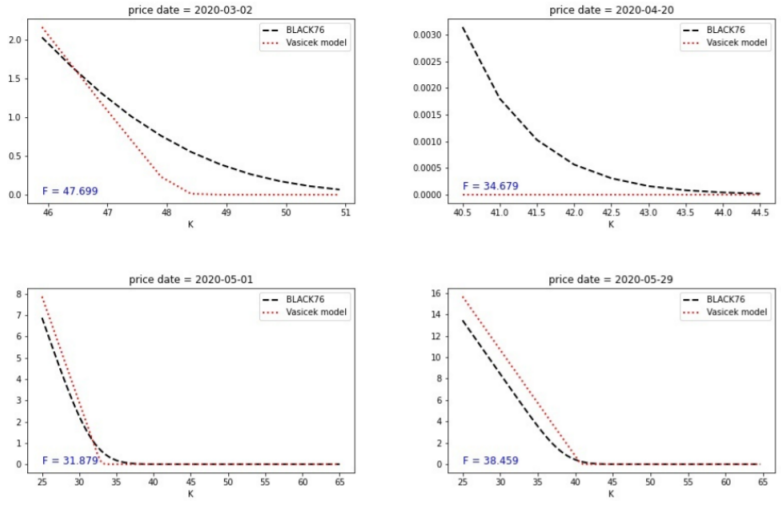


FIGURE 1. Black76 vs Vasicek models for Call option prices

we can employ Vasicek model again to come up with the call option prices. Here Black 76 model would fail as it does not accept the negative prices. Figure 2 shows the price of the option for various strike prices. These prices have been calculated using Monte Carlo simulation.

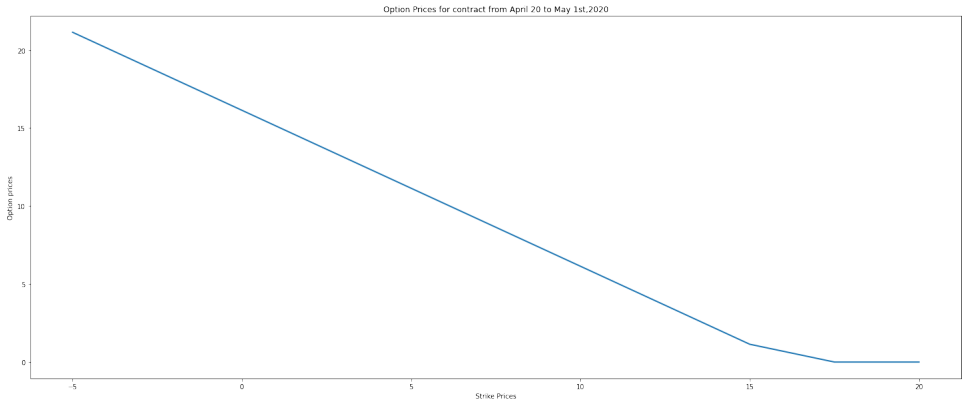


FIGURE 2. Option prices if futures prices becomes negative

4.3. **NYMEX Natural Gas Dataset.** For the Natural Gas (NG) dataset, we will see that all the underlying future prices are positive. However, they exhibit a non-zero mean-reversion process over the time (figure 3).

In this case, for the corresponding option prices, we used the Continuous-Time GARCH model (as in equation 4).

In this methodology, first we should consider the model (4) under a risk-neutral probability P^*. Therefore, in a risk-neutral world our model will take the following look:

$$dS_t = a^*(b^* - S_t)dt + \sigma S_t dW_t^* \tag{7}$$

where:

$$a^* := a + \lambda\sigma, b_* := \frac{ab}{a + \lambda\sigma}$$

and W_t^* is defined as

$$W_t^* := W_t + \lambda \int_0^t S(u)du.$$

Here, $\lambda \in \mathbb{R}$ is the *market price of risk*.

For this model (7) we have an explicit option pricing formula for European Call option [4]:

$$\begin{aligned} C_T^* =& e^{-(r+a^*)T} S(0)\Phi(y_+) - e^{-rT} K\Phi(y_-) + \\ & b^* e^{-(r+a^*)T} \left[(e^{a^* T} - 1) - \int_0^{y_0} z F_T^*(dz) \right] \end{aligned}$$

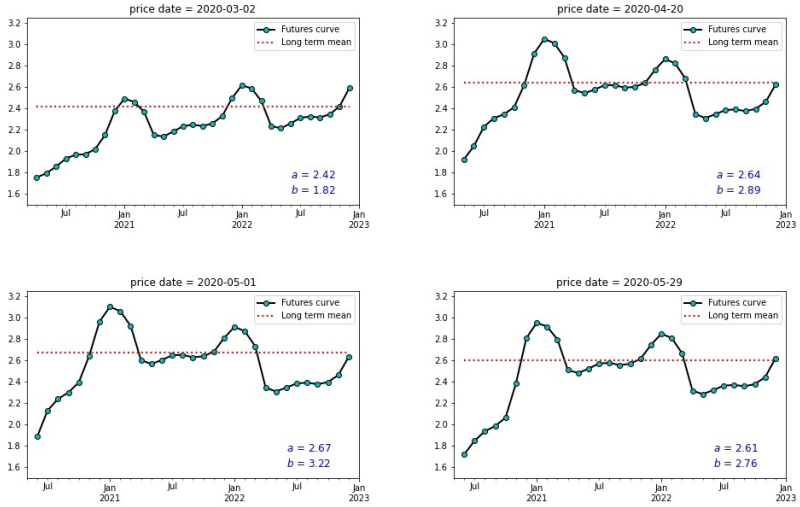


FIGURE 3. Calibrations of parameters for different initial "price dates." Here the x-axis is the expiry date (T) and the y-axis is the price per unit. The dotted line is the mean value b.

where, y_0 is the solution of:

$$y_0 = \frac{\ln \frac{K}{S(0)} + \left(\frac{\sigma^2}{2} + a^*\right)T}{\sigma\sqrt{T}} - \frac{\ln\left(1 + \frac{a^* b^*}{S(0)}\right) \int_0^T e^{a^* s} e^{-\sigma y_0 \sqrt{s} + \frac{\sigma^2 s}{2}} ds}{\sigma\sqrt{T}}$$

with,

$$y_+ := \sigma\sqrt{T} - y_0, \quad \text{and} \quad y_- := -y_0,$$

and, $F_T^*(dz)$ is the probability distribution under the risk-neutral probability P^*, as in [4].

4.3.1. *Methodology and results.* In this approach, to avoid the huge computations regards to the explicit formula, we used Least Square Regression method for calibrating the parameters by following the methodology in [5],

$$F_{i+1} = \tau F_i + \mu + sd(e),$$

to have the following equations:

$$F_x = \sum_{i=1}^{n} F_{i-1}, \qquad F_y = \sum_{i=1}^{n} F_i,$$
$$F_{xx} = \sum_{i=1}^{n} F_{i-1}^2, \qquad F_{yy} = \sum_{i=1}^{n} F_i^2,$$
$$F_{xy} = \sum_{i=1}^{n} F_{i-1} Fi$$

and then the following relationships can be considered:

$$\tau = \frac{nF_{xy} - F_x F_y}{nF_{xx} - F_x^2},$$
$$\mu = \frac{F_y - \tau F_x}{n},$$
$$sd(e) = \sqrt{\frac{nF_{yy} - F_y^2 - \tau(nF_{xy} - F_x F_y)}{n(n-2)}}.$$

For our purpose, we used the Euler approximation to simulate the future prices in order to approximate the corresponding European Call option prices.

$$F_{i+1} = F_i \exp a^*\delta + b^*(1 - \exp -a^*\delta) + \sigma F_i \sqrt{\frac{1 - \exp -2a^*\delta}{2a^*}} N_{0,1} \tag{8}$$

Here, $\delta > 0$ is a time space, and the F_i prices are the exact discrete solution of equation (4). Hence, we can find the following relations between the parameters:

$$a = -\frac{\ln \tau}{\delta}, b = \frac{\mu}{1-\tau}, \sigma = sd(e)\sqrt{\frac{-2\ln \tau}{\delta(1-\tau^2)}}$$

Finally, for the risk neutral parameters, the following adjustment has been applied:

$$a^* = a + \lambda\sigma, b^* = \frac{ab}{a + \lambda\sigma}$$

According to our dataset, there was not any access to the market option prices to estimate the market price of risk. Therefore, the following formula has been taken into account:

$$\lambda := \frac{\frac{dF}{F} - r}{\sigma}$$

where, $\frac{dF}{F}$ is a returns on futures prices, r is the interest rates, and σ is the implied volatilities.

The future prices were simulated 20 times (an exercise of this is shown in figure 4), and the average of them is applied in the payoff function. Then,

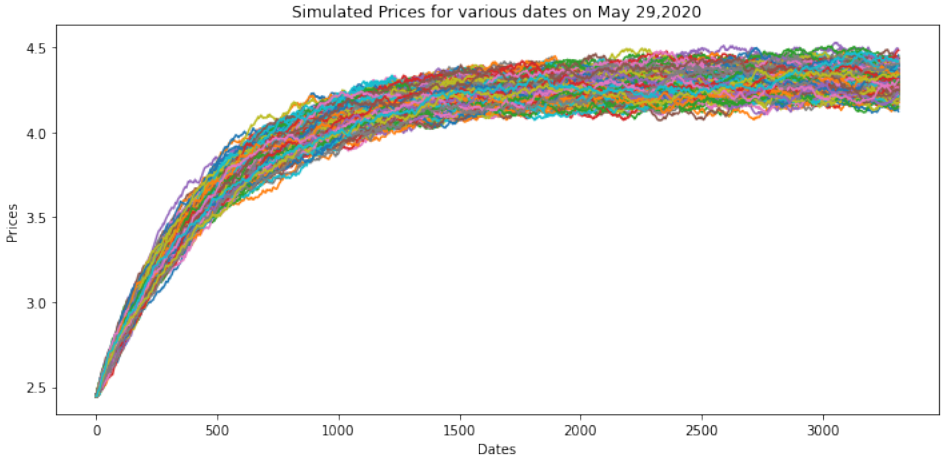


FIGURE 4. The evolution of simulated future price with respect to time

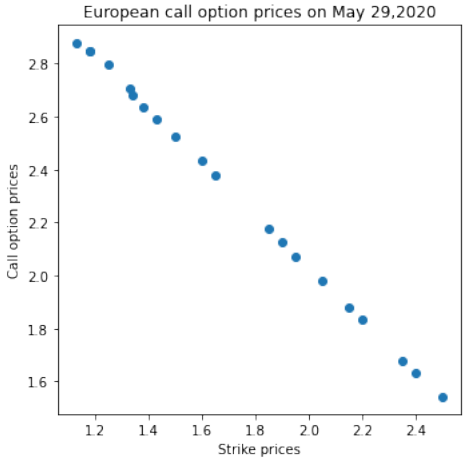


FIGURE 5. In this picture, we can see the evolution of the calculated option prices, according to the Continuous-Time GARCH model, with respect to their related strike prices is depicted

the discount of the average of payoffs considered as the requested call option prices with continuous-time GARCH model approach (results can be seen through figure (5) and (6)).

Here, the risk-neutral parameters a^* and b^* have been estimated as 1.68528518 and 2.64820985 respectively.

	maturity	f price	volatility	strike	B76_call	GARCH_call
0	2020-07-31	1.939	0.672	1.90	0.233045	2.140476
1	2020-08-31	1.987	0.680	1.45	0.593479	2.581413
2	2020-09-30	2.067	0.764	1.10	0.989349	2.940450
3	2020-10-31	2.388	0.573	1.65	0.799208	2.374987
4	2020-11-30	2.810	0.433	2.05	0.817705	1.986476
5	2020-11-30	2.810	0.433	2.05	0.817401	1.986476
6	2020-11-30	2.810	0.436	2.55	0.475949	1.491682
7	2020-11-30	2.810	0.436	2.55	0.475772	1.491682
8	2020-12-31	2.954	0.437	2.25	0.804359	1.777884
9	2021-02-28	2.797	0.412	1.43	1.373119	2.598228

FIGURE 6. In this table, the accuracy of our model comparing to the known Black-76 model has been exhibited for the first 10 strike prices

5. CONCLUSION

In this project, we worked with some useful alternative models which are helpful for valuation of options on future contracts. While Black 76 is the most commonly used model for pricing option future contracts in Industry, it is necessary to have alternative models for the valuation when the prices' behaviour differs from the prices describe by the same model. For WTI future option prices, we have shown that the models O-U and Vasicek have similar prices to those given by Black 76 when future prices are positive; and give a valuation also when the future prices are negative. This is useful when irregular events take place. Also, for Natural Gas future option prices, continuous time GARCH also displays comparable values to Black 76. Further it allows us to calibrate a mean reversion parameter to describe in a better way future option prices and their behaviour.

As a recommendation, it would be useful for industry to keep track of this two models to know how to react in unusual situations and double check their own valuation prices. This models have been shown to be simple to understand, clear to calculate and comparable with what the industry uses. With respect to the data and results, in table 1 are some suggestions for the valuation according to the data's nature:

Future Prices	**Mean-Reversion**	**Model**
Positive	0	GBM or Black76
Positive	b	Continuous-time GARCH
Negative and Positive	0	OU process
Negative and Positive	b	Vasicek model

TABLE 1. Recommended model according to sign of prices and mean reversion behaviour.

ACKNOWLEDGEMENT

We appreciate Professor Anatoliy Swishchuk for his continuous teaching, and endless patience. We thank Scott Dalton for his time, the database provided and his willingness to share his industry expertise. Finally, we want to thank PIMS committee and Specially Professor Kristine Bauer for her enthusiasm to help us in every step.

REFERENCES

[1] S.Dalton, *Ovintiv, Dates from March, 2020 to May, 2020.* Data basis for future prices, rates and implied volatility for WTI and Natural Gas.
[2] *Prentice Hall, NJ. (1997)* Options, Futures, and Other Derivatives
[3] A. Swishchuk, Alternatives to Black-76 Model for Options Valuation of Futures Contracts, *Lectures' Notes, PIMS* $Math_Industry Workshop$, *August*2020.
[4] A. Swishchuk, Explicit Option pricing furmula for a Mean-reverting asset in Energy Market, *Journal of Numerical and Applied Mathematics, Vol.1 (96), 2008*, *DOI: 10.13140/RG.2.2.32915.91682.*
[5] Thijs van den Berg, Calibrating the Ornstein-Uhlenbeck (Vasicek) model, *Published May 28, 2011.*
[6] G. E. Uhlenbeck, L. S. Ornstein, On the theory of Brownian Motion, *Phys. Rev. 36 (5): 823-841, 1930.*
[7] O. Vasicek, An equilibrium characterization of the term structure., *J. of Finan. Economics, 5 (1977), pp. 177-188.*
[8] R. Weron, Market price of risk implied by Asian-style electricity options and futures. *Energy Economics 30 (2008).*

MCMILLAN–MCGEE: MODELLING INDUCTION HEATER

ANTON IATCENKO, YAKINE BAHRI, NOAH BOLOHAN, BENJAMIN MACADAM, AND RYAN THIESSEN

Industry mentor: Edwin Reid Academic mentor: Yakine Bahri.

Abstract. In this project we consider a cylindrical induction heater of the type built by the McMillan–McGee corporation. Our goal is to develop a model for the electric field induced inside the heater, and understand the effects of changing materials and physical dimensions. We studied the electrical field intensity through the Maxwell equations, which allowed us to find an exact expression for the field intensity and calculate the power flowing into the casing of the pipe.

1. Problem Description

Contaminated soils are a significant environmental and safety concern. Many contaminants have the ability to flow into aquifer systems, thereby contaminating the public water supply. The depth at which some contaminants occur renders the use of excavation prohibitively expensive. Therefore, other methods are employed to remove contaminants in-situ, where depth is not a factor.

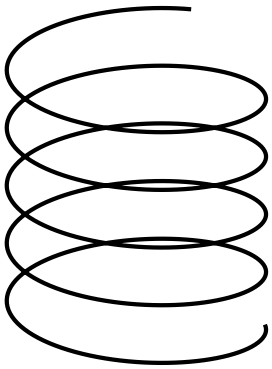
One such method includes heating the soil using electricity in order to vaporize the contaminants, which are subsequently extracted from the soil. Methods of heating the soil are also utilized in connection with heating subterranean heavy oil reservoirs or bitumen deposits to reduce the viscosity of the hydrocarbons so that it can be recovered more easily. A typical approach is to bury a tall cylindrical heating element in the ground, and thus heat it up at the desired depth.

However, cost considerations limit the heating elements and their housings to a small diameter. Moreover, the heating equipment used in such operations are sunk costs, as they are typically left in the ground after a remediation project is completed. Therefore, there is a need for an economical method and a device for heating soil that provides a large heating surface area, enables the selective heating of vertical extents of the element to different temperatures, and is capable of achieving soil temperatures sufficient to remediate contaminants with high boiling points, while allowing for recovery of at least some of the heating equipment after operations have concluded.

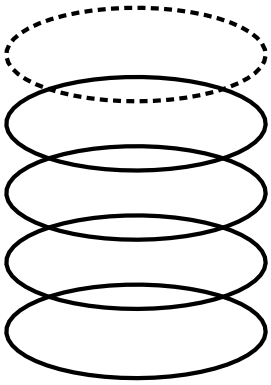
One example of such a device consists of a conductive casing, such as a steel pipe, which is heated by the induction of an electric current within its wall as a result of the passing of an alternating current through a conductor located inside the casing. The alternating current has to be of a sufficiently high frequency in order to exploit the skin effect: it limits the penetration of the current into the pipe wall to a very thin shell where, given a conductor in helical form, the current flows in a circumferential direction. That is, the current density in the casing is greatest near the inner surface of the casing. The skin effect results in establishing an appreciable resistance in the casing, under which the passage of current through the resistance generates heat.

2. Geometry of The Problem

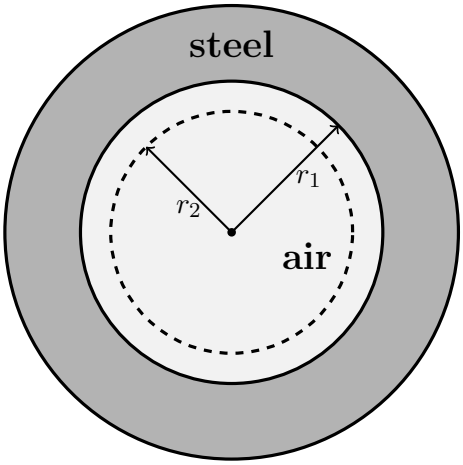
In order to simplify our model, we may assume that we have conducting rings inside the induction heater instead of a conducting helix.



Idealization 1:
conducting helix



Idealization 2:
conducting rings



View from above

3. Methodology

We start with the description of the mathematical model. The electromagnetic field is governed by the Maxwell equations, that relate the electric and magnetic field intensities to the current induced in the work coil. In this work, we assume our domain to have two distinct regions: the inside of the cylindrical shell and inside the steel housing itself. The governing equations are mathematically the same in both regions; the need for separation comes from the difference in physical parameters.

We write the equations in the cylindrical coordinates system to take advantage of the rotational symmetry. Next, we derive an elliptic partial differential equation satisfied by the angular component of the electric field intensity. We use the Fourier series representation in the vertical variable, which reduces the original PDE to a diagonal system of Bessel ODEs. Consequently, the Fourier coefficient of the solution are linear combinations of the Bessel functions. The required constants in the solution are found from the boundary conditions at the origin and at infinity, and the interface conditions between the interior and the housing of the heating element.

The second part of this work consists of numerically evaluating and visualizing the solutions. Moreover, we are able to evaluate the power flowing into the casing using the Poynting vector field.

4. Setup

The electromagnetic fields are described by the Maxwell's equations:

$$\nabla \times \mathbf{E} = -i\omega\mu\mathbf{H}, \tag{4.1a}$$

$$\nabla \times \mathbf{H} = i\omega\epsilon\mathbf{E} + \mathbf{J}, \tag{4.1b}$$

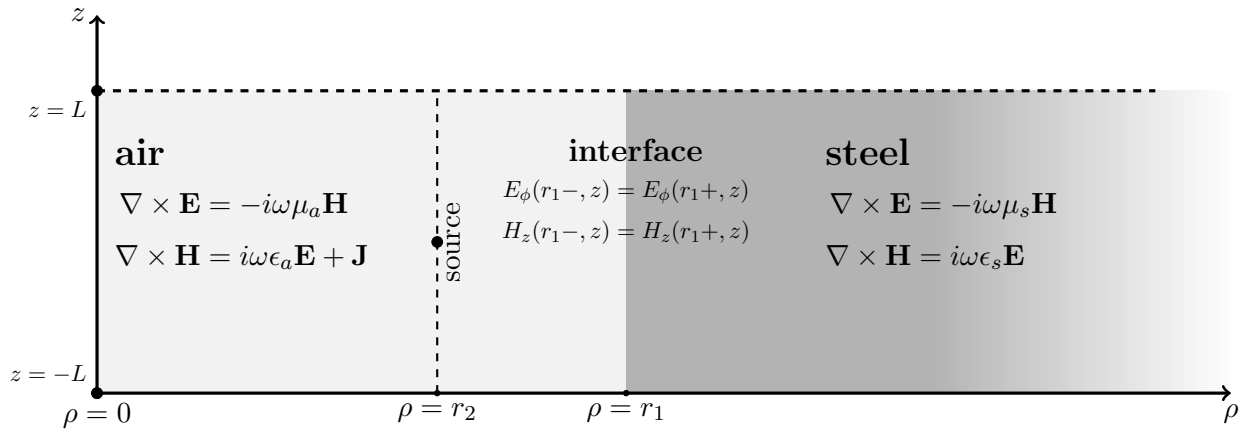
where

- $\mathbf{E}$ is the electric field intensity (volts/meter)
- $\mathbf{H}$ is the magnetic field intensity (amps/meter)
- $\mathbf{J}$ is the electric current density (amps/meter^2)
- μ is the permeability of the medium (henrys/meter)
- ϵ is the permittivity of the medium (farads/meter).

We take the current source to be an infinite collection of rings of radius r_2, spaced at the intervals of $2L$. We will work with a single loop located at $z = 0$, and impose periodic boundary conditions in $z \in [-L, L]$. The current density is represented as

$$\mathbf{J}(\rho, z) = \hat{\boldsymbol{\phi}}\, I_0 \frac{\delta(\rho - r_2)\delta(z)}{2\pi\rho}, \tag{4.2}$$

where I_0 is the total current (constant, in amps). We will solve a coupled system of PDEs: both are (4.1), but with different values of the physical parameters μ and ϵ. The setup is summarized in the figure below.



Boundary conditions: E_ϕ is finite at $\rho{=}0$, vanishes as $\rho \to \infty$ and $2L$-periodic in z.

5. Symmetry Reduction

Since the system is invariant under rotations, we have $\partial_\phi = 0$ for both regions. We write out the Maxwell's system (4.1) in components, and reduce it:

$$\frac{1}{\rho}\cancel{\frac{\partial E_z}{\partial \phi}} - \frac{\partial E_\phi}{\partial z} = -i\omega\mu_a H_\rho \tag{5.1a}$$

$$\frac{\partial E_\rho}{\partial z} - \frac{\partial E_z}{\partial \rho} = -i\omega\mu_a H_\phi \tag{5.1b}$$

$$\frac{1}{\rho}\left(\frac{\partial}{\partial \rho}(\rho E_\phi) - \cancel{\frac{\partial E_\rho}{\partial \phi}}\right) = -i\omega\mu_a H_z \tag{5.1c}$$

$$\frac{1}{\rho}\cancel{\frac{\partial H_z}{\partial \phi}} - \frac{\partial H_\phi}{\partial z} = i\omega\epsilon_a E_\rho \tag{5.1d}$$

$$\frac{\partial H_\rho}{\partial z} - \frac{\partial H_z}{\partial \rho} = i\omega\epsilon_a E_\phi + I_0 \frac{\delta(\rho - r_2)\delta(z)}{2\pi\rho} \tag{5.1e}$$

$$\frac{1}{\rho}\left(\frac{\partial}{\partial \rho}(\rho H_\phi) - \cancel{\frac{\partial H_\rho}{\partial \phi}}\right) = i\omega\epsilon_a E_z \tag{5.1f}$$

The system (5.1) decouples into two independent systems: $\big[$(5.1a), (5.1c), (5.1e)$\big]$ and $\big[$(5.1d), (5.1f), (5.1b)$\big]$. The latter lacks a source, so the solution must be zero:

$$E_\rho = E_z = H_\phi = 0.$$

The former is actually nontrivial, and will be the subject of the work in the following sections.

6. Solving The Equation: Inner Domain

Differentiating (5.1a) and (5.1c) with respect to z and ρ respectively, and substituting the results into (5.1e) gives

$$\frac{\partial^2 E_\phi}{\partial z^2} + \frac{\partial}{\partial \rho}\left(\frac{1}{\rho}\frac{\partial}{\partial \rho}\big(\rho E_\phi\big)\right) = -\omega^2 \mu_a \epsilon_a E_\phi + \frac{i\omega\mu_a I_0}{2\pi}\frac{\delta(\rho - r_2)\delta(z)}{\rho} \tag{6.1}$$

We set

$$\alpha = \frac{i\omega\mu_a I_0}{4\pi L} \qquad g = \frac{E_\phi}{\alpha} \qquad k_a = \omega\sqrt{\mu_a \epsilon_a} \tag{6.2}$$

and write

$$\frac{\partial^2 g}{\partial z^2} + \frac{\partial^2 g}{\partial \rho^2} + \frac{1}{\rho}\frac{\partial g}{\partial \rho} + \left(k_a^2 - \frac{1}{\rho^2}\right) g = \frac{2L\delta(\rho - r_2)\delta(z)}{\rho}. \tag{6.3}$$

Our next step is to take the Fourier series in the z-variable (this will automatically enforce periodicity):

$$g(\rho, z) = \sum_{n\in\mathbb{Z}} \hat{g}_n(\rho) e^{i\pi n z/L} = \sum_{n\in\mathbb{Z}} \hat{g}_n(\rho) e^{ik_n z}, \tag{6.4}$$

where $k_n = \pi n/L$ is the Fourier wave number. Then the coefficients satisfy

$$\frac{1}{\rho^2}\left(\rho^2\frac{\partial^2}{\partial \rho^2} + \rho\frac{\partial}{\partial \rho} + \big(k_a^2 - k_n^2\big)\rho^2 - 1\right)\hat{g}_n(\rho) = \frac{\delta(\rho - r_2)}{\rho}. \tag{6.5}$$

Let $\lambda_n^2 = k_n^2 - k_a^2$, and set $x = \lambda_n \rho$ and $\hat{g}_n(\rho) = \hat{u}_n(\lambda_n \rho) = \hat{u}_n(x)$. Then the equation above becomes

$$\underbrace{\left(x^2\frac{\partial^2}{\partial x^2} + x\frac{\partial}{\partial x} - \big(x^2 + 1\big)\right)}_{\text{modified Bessel operator of order 1}}\hat{u}_n(x) = \frac{x}{\lambda_n}\delta\big(x/\lambda_n - r_2\big). \tag{6.6}$$

For $x \neq \lambda_n r_2$, the equation above is the modified Bessel ODE of order 1, so we expect the solution to take the form

$$\hat{u}_n(x) = \begin{cases} A_n I_1(x) + B_n K_1(x), & 0 < x < \lambda_n r_2, \\ C_n I_1(x) + D_n K_1(x), & \lambda_n r_2 < x < \lambda_n r_1, \end{cases} \tag{6.7}$$

where I_1 and K_1 are the modified Bessel functions of order one, of the first and second kind respectively.

To find the constants A_n, B_n, C_n and D_n we begin with enforcing the continuity and jump conditions at $x = \lambda_n r_2$. The continuity condition dictates that

$$\big(C_n - A_n\big) I_1(\lambda_n r_2) + \big(D_n - B_n\big) K_1(\lambda_n r_2) = 0. \tag{6.8}$$

The jump condition[1] dictates that

$$\big(C_n - A_n\big)I_1'(\lambda_n r_2) + \big(D_n - B_n\big)K_1'(\lambda_n r_2) = \frac{1}{\lambda_n r_2}. \tag{6.9}$$

We can write (6.8) and (6.9) together as a linear system:

$$\underbrace{\begin{bmatrix} I_1 & K_1 \\ I_1' & K_1' \end{bmatrix}}_{=:M} \begin{pmatrix} C_n - A_n \\ D_n - B_n \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\lambda_n r_2} \end{pmatrix} \tag{6.10}$$

where it is understood that all entries of M are evaluated at $\lambda_n r_2$. We can invert M using the Wronskian identity for the Bessel functions, and thus obtain

$$\begin{pmatrix} C_n - A_n \\ D_n - B_n \end{pmatrix} = M^{-1} \begin{pmatrix} 0 \\ \frac{1}{\lambda_n r_2} \end{pmatrix} = -\lambda_n r_2 \begin{bmatrix} K_1' & -K_1 \\ -I_1' & I_1 \end{bmatrix} \begin{pmatrix} 0 \\ \frac{1}{\lambda_n r_2} \end{pmatrix} = \begin{pmatrix} K_1 \\ -I_1 \end{pmatrix} \tag{6.11}$$

That is,

$$C_n - A_n = K_1(\lambda_n r_2) \qquad \text{and} \qquad D_n - B_n = -I_1(\lambda_n r_2). \tag{6.12}$$

Recall that Bessel function K_1 has a singularity at the origin, so we must set $B_n = 0$ for all. Consequently,

$$D_n = -I_1(\lambda_n r_2) \tag{6.13}$$

An additional condition on C_n and A_n will be provided by the continuity requirements at the interface $\rho = r_1$.

7. Solving The Equation: Outer Domain

We now develop the solution in the darker region $r_1 < \rho < \infty$. The symmetry reduction is identical to the one before, and so are the two resulting independent subsystems. The one featuring E_ρ, E_z and $H_\phi = 0$ is once again without a source, so we conclude that $E_\rho = E_z = H_\phi = 0$ for all domains.

Equation for the other three components is similar to (6.1), except we don't even have a delta source. In place of (6.1) we have

$$\frac{\partial^2 E_\phi}{\partial z^2} + \frac{\partial}{\partial \rho}\left(\frac{1}{\rho}\frac{\partial}{\partial \rho}(\rho E_\phi)\right) = -k_s^2 E_\phi, \tag{7.1}$$

where we have set $k_s = \omega\sqrt{\mu_s \epsilon_s}$. We proceed by writing the solution as a Fourier series in z:

$$E_\phi(\rho, z) = \sum_{n\in\mathbb{Z}} \hat{E}_n(\rho) e^{ik_n z}, \qquad k_n = \frac{\pi n}{L}. \tag{7.2}$$

[1]See Appendix A for the derivation

Substituting this representation into (7.1) gives

$$\left(\rho^2\frac{\partial^2}{\partial\rho^2}+\rho\frac{\partial}{\partial\rho}+\left(k_s^2-k_n^2\right)\rho^2-1\right)\hat{E}_n=0. \tag{7.3}$$

With the change of variables $\nu_n^2=k_n^2-k_s^2,\ x=\nu_n\rho$ and $\hat{E}_n(\rho)=\hat{v}_n(\nu_n\rho)=\hat{v}_n(x)$, the equation above becomes

$$\left(x^2\frac{\partial^2}{\partial x^2}+x\frac{\partial}{\partial x}-(x^2+1)\right)\hat{v}_n=0, \tag{7.4}$$

which is the modified Bessel equation of order 1. Since we require the solution to vanish at infinity, we can write the solution as

$$\hat{E}_n(\rho)=\alpha F_n K_1(\nu_n\rho), \tag{7.5}$$

where F_n is the constant yet to be determined. (We don't actually have to put α into the expression above, but it makes the next step a bit nicer.)

8. Interface Conditions

We will require the electric field E_ϕ and the magnetic field H_z to be continuous at the interface $\rho=r_1$. The continuity condition $E_\phi(r_1-,z)=E_\phi(r_1+,z)$ is written as

$$\alpha\sum_{n\in\mathbb{Z}}\Big[C_nI_1(\lambda_nr_1)+D_nK_1(\lambda_nr_1)\Big]e^{ik_nz}=\sum_{n\in\mathbb{Z}}\alpha F_nK_1(\nu_nr_1)e^{ik_nz} \tag{8.1}$$

$$C_nI_1(\lambda_nr_1)+D_nK_1(\lambda_nr_1)=F_nK_1(\nu_nr_1). \tag{8.2}$$

Recall that D_n is already known from (6.13), so we have

$$C_nI_1(\lambda_nr_1)-F_nK_1(\nu_nr_1)=I_1(\lambda_nr_2)K_1(\lambda_nr_1) \tag{8.3}$$

To impose the continuity condition $H_z(r_1-,z)=H_z(r_1+,z)$, we first recall that by equation (5.1c) we have

$$H_z(r_1-,z)=\frac{i}{r_1\omega\mu_a}\frac{\partial\big(\rho E_\phi(r_1-,z)\big)}{\partial\rho}\qquad\text{and}\qquad H_z(r_1+,z)=\frac{i}{r_1\omega\mu_s}\frac{\partial\big(\rho E_\phi(r_1+,z)\big)}{\partial\rho},$$

so we must have

$$\frac{1}{\mu_a}\frac{\partial\big(\rho E_\phi(r_1-,z)\big)}{\partial\rho}=\frac{1}{\mu_s}\frac{\partial\big(\rho E_\phi(r_1+,z)\big)}{\partial\rho}. \tag{8.4}$$

After some algebraic manipulations and removal of the derivatives of the Bessel functions we get the following system for the coefficients C_n and F_n:

$$\underbrace{\begin{bmatrix} I_1(\lambda_n r_1) & -K_1(\nu_n r_1) \\ I_0(\lambda_n r_1) & -\gamma_n K_0(\nu_n r_1) \end{bmatrix}}_{=:W_n} \begin{pmatrix} C_n \\ F_n \end{pmatrix} = I_1(\lambda_n r_2) \begin{pmatrix} K_1(\lambda_n r_1) \\ K_0(\lambda_n r_1) \end{pmatrix} \tag{8.5}$$

where $\gamma_n = \frac{\nu_n \mu_a}{\lambda_n \mu_s}$. Unfortunately, matrix W_n defined above is not a Wronskian of the two Bessel functions. It's inverse would not simplify, and therefore it is not worthwhile to compute it by hand; instead, we will solve this system numerically for each n.

Moreover, the matrix W_n as presented above is extremely ill-conditioned. To mitigate this issue, we rewrite our solution in terms of the scaled Bessel functions

$$\tilde{I}_n(z) = e^{-|\Re z|} I_n(z) \qquad \text{and} \qquad \tilde{K}_n(z) = e^{z} K_n(z). \tag{8.6}$$

9. Transferred Power

The time-averaged Poynting vector field is given by

$$\mathbf{P} = \frac{1}{2} \Re\big(\mathbf{E} \times \mathbf{H}^*\big), \tag{9.1}$$

where * denotes the complex conjugate. One way to compute the power flowing through the boundary of the casing is to evaluate the integral

$$P_{\text{Poynt}} = 2\pi r_1 \int_{-L}^{L} (\mathbf{P}(r_1, z))_\rho \, dz. \tag{9.2}$$

Alternatively, we can assume that the current density inside the steel housing is given by $J_\phi = \sigma E_\phi$, and calculate the total power as

$$P_{E^2} = 2\pi\sigma \int_{-L}^{L} \int_{r_1}^{\infty} E_\phi^2 \, r \, dr \, dz. \tag{9.3}$$

Both approaches were implemented with trapezoid rule and give similar results.

10. Physics

Physical constants:

(1) Casing radius $r_1 = 76$mm.
(2) Source (coil) radius $r_2 = 66$mm.
(3) Half-pitch $L = 15$mm.
(4) $\omega = 2\pi \times 12$ kHz is the temporal frequency.

(5) Permeabilities[2]: $\mu_{\text{air}} = 1.26 \times 10^{-7} H/m$, $\mu_{\text{steel}} = 1.26 \times 10^{-6} H/m$

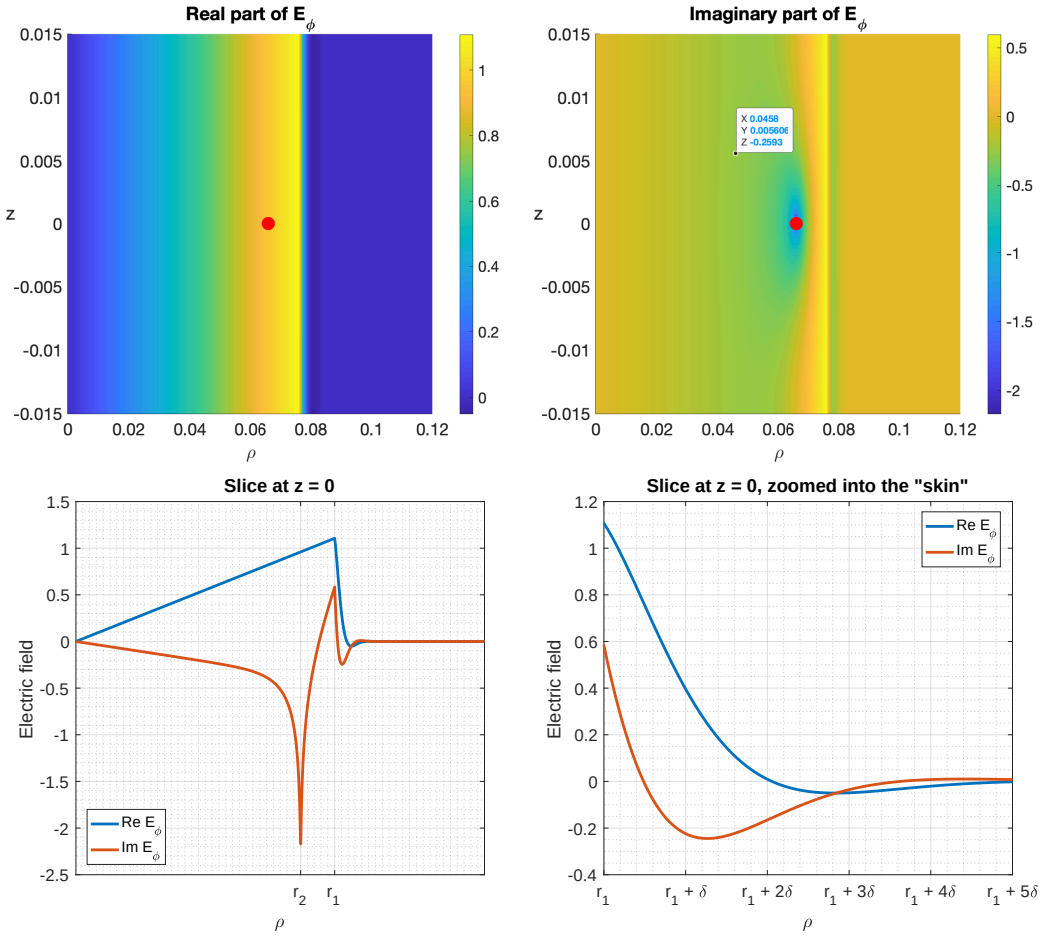
(6) Conductivity $\sigma_{\text{steel}} = 7 \times 10^{6}$ siemens/meter.

(7) Permittivities: $\epsilon_{\text{air}} = 9 \times 10^{-12} F/m$, $\epsilon_{\text{steel}} = \epsilon_{\text{air}} - \dfrac{i\sigma_{\text{steel}}}{\omega}$

With these values, the model gives the power flow into the steel casing of about 68 watts.

11. Plots

Here are a few plots generated by the accompanying Matlab script.



[2]The values of permeabilities are not those quoted for the "room temperature". They were adjusted to account for the effect of the high temperature.

12. Conclusion

We developed an analytical model which helps to compute the electric fields inside the heating element, as well as quantitatively assess its dependence on the physical parameters of the system. This analysis can be used to determine the optimal structure of the heating element, as well as to better understand the underlying physical processes.

A. Appendix : Jump Condition

To derive the jump condition we rewrite equation (6.6) as follows:

$$\frac{\partial}{\partial x}\left(x\frac{\partial u}{\partial x}\right) - \frac{x^2+1}{x}u = \frac{1}{\lambda_n}\delta\big(x/\lambda_n - r_2\big)$$

Integrate both sides in the neighbourhood of the jump:

$$\int_{\lambda_n r_2 - h}^{\lambda_n r_2 + h}\left(\frac{\partial}{\partial x}\left(x\frac{\partial u}{\partial x}\right) - \frac{x^2+1}{x}u\right)dx = \int_{\lambda_n r_2 - h}^{\lambda_n r_2 + h}\delta\big(x/\lambda_n - r_2\big)\frac{dx}{\lambda_n}$$

$$\left(\big(\lambda_n r_2 + h\big)\frac{\partial u}{\partial x}\Big|_{x=\lambda_n r_2 + h}\right) - \left(\big(\lambda_n r_2 - h\big)\frac{\partial u}{\partial x}\Big|_{x=\lambda_n r_2 - h}\right) + \mathcal{O}(h) = 1$$

Let $h \to 0$:

$$\left(\lambda_n r_2 \frac{\partial u}{\partial x}\Big|_{x=\lambda_n r_2 +}\right) - \left(\lambda_n r_2 \frac{\partial u}{\partial x}\Big|_{x=\lambda_n r_2 -}\right) = 1$$

$$\frac{\partial u}{\partial x}\Big|_{x=\lambda_n r_2 +} - \frac{\partial u}{\partial x}\Big|_{x=\lambda_n r_2 -} = \frac{1}{\lambda_n r_2}$$

E-mail address: aiatcenk@sfu.ca

E-mail address: ybahri@uvic.ca

E-mail address: nbolo094@uottawa.ca

E-mail address: benjamin.macadam@ucalgary.ca

E-mail address: rt5@ualberta.ca

Math^Industry Workshop

ANALYZING LEGISLATORS AND POLICY AREAS

DANA BERMAN, JILLIAN GLASSETT, DANYI LIU, DIAAELDIN TAHA, AND AARON (XIANG) ZHENG

Abstract. We used official legislative data on the Canadian and United States governments to see if politicians cluster around policy topics. Both data sets provided different information on legislators and legislation, resulting in two distinct methodologies and results. For the Canadian data set, we identified the distribution of times spoken in parliament per topic and visualized how often the topic came up in session from 2017 to 2019 using t-SNE. We saw that some topics were evenly spread over the three years while other topics clustered around a particular year. We provided insight into how this data can be used to analyze legislative performance. For the U.S. data set, we examined the percentage of bills that became law as a measure of political performance for current legislators. We also used various clustering techniques to analyzed politician's interest in different policy areas based on (co-) sponsored bills. We found that these groupings bear strong connections to both parties and location. We also performed some preliminary analysis on legislation passing rates by topic.

1. Introduction

Performance measurement in the the field of sports has achieved significant growth and development. Using data-based performance evaluation to make key decisions has played an important role in sports such as football, basketball, and ice hockey[1]. Analysis of complex aspects of the game requires a comprehensive mathematical tool to facilitate a continuous cycle of "question and answer" to provide detailed and flexible explanations.

What if we applied the same mathematical tools and concepts to political performance? IOTO International Inc. is a non-partisan analysis company that specializes in using AI to gain insights from political data. Through the PIMS Math^Industry workshop, we were given the opportunity to collaborate with this company. The IOTO team provided us with some data sets mined from official government websites. This report summarizes the work, which occurred over a two-week period.

Out of respect and confidentiality considerations, we will not provide the data sets nor the code used.

2. Problem Statement

IOTO provided us with two main data sets, one from Canada and one from the U.S. After cleaning up the data and conducting an initial analysis, we found that the the data sets were too different to make a direct comparison between the two countries a worthy endeavor. The Canadian data set was based on debate records on the floor of Parliament from 2017

[1]"Decomposing the Immeasurable Sport: A deep learning expected possession value framework for soccer", Javier Fernández, Luke Bornn, Dan Cervone, `http://www.lukebornn.com/papers/fernandezsloan2019.pdf`

through 2019. Beyond basic information on the legislators, the data set contained the date, time, duration and topic that each legislator spoke on; it did not contain any legislative information. The U.S. data set contains the total bills sponsored or co-sponsored by each legislator; it also contains some basic information on each legislator. For each legislator, the record covered the legislator's entire congressional career, and was separated into thirty-two policy topics. Since the both data sets had information on policy topics, we focused on the following question:

Question. *Do politicians cluster around certain topics?*

In addition to this question, we performed a preliminary analysis of the percentage of (U.S.) bills that pass, and on the passing rate of legislator clusters. This was possible after the IOTO team mined some additional data on bill status. Given that we are studying the Canada and U.S. separately, we will split our report into two sections, one per country. Each section will detail our methodology and results. We will summarize all the results at the end.

3. Canada

Hansard is the name of the official reports of the Parliamentary debates in Britain and several other Commonwealth countries including Canada. It is named after the Hansards, a family of printers who worked with the Parliament at Westminster in the late 18th century.[2]

We received access to a data set that is derived from the Canadian Parliamentary Hansard speech data and were tasked with analysing the topics that were discussed in the debates. The data set included such information as MPs names, party affiliation, and timestamped summaries of the topics the MPs discussed on the floor of the Canadian Parliament. We restricted our analysis to the years 2017, 2018, and 2019. Our analysis involved 88,235 data points covering around 7,000 topics.

3.1. Motivating Questions.

For the sake of brevity, we showcase our work for only two questions:

(1) Is it possible to detect activity in the Parliament? Is it possible to tell when a particular subject is becoming a popular debate topic?
(2) Can we tell which topics are usually discussed with other topics or on their own?

3.2. Methodology and Results.

3.2.1. *Detecting activity and popularity.* We used the number of times an event of interest occurs in the Canadian Parliament during a specified time window as a sign of activity. This is a very natural indicator of activity that is prevalent throughout both the scientific and the popular literature. For instance, a sharp increase in a location of the number of cases confirmed to have contracted a particular disease is a sign that an outbreak might be taking place.

We created two plots showing the spikes in debates from 2017 to 2019 in the Canadian Parliament. Both plots are of a 30-day moving average to reduce the amount of noise. The

[2] *Encyclopaedia Britannica*, "Hansard," (accessed September 01, 2020), `https://www.britannica.com/topic/Hansard`

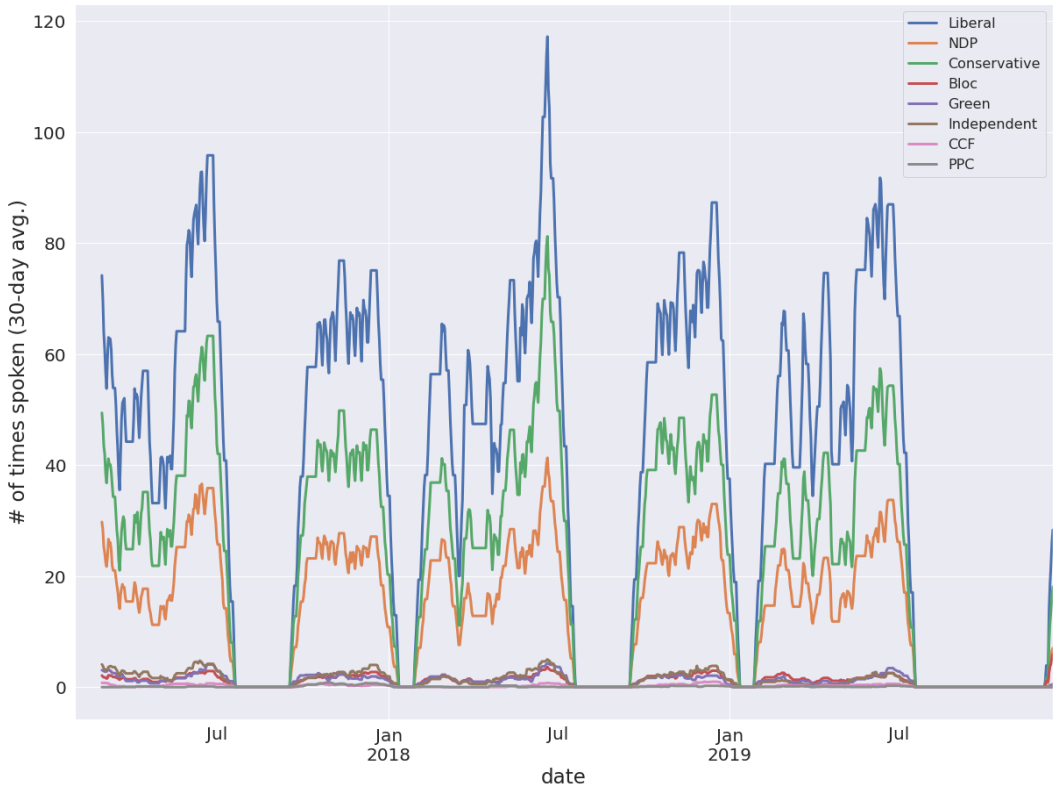


FIGURE 1. A plot of the 30-days moving average of the number of times each party spoke on the floor of the Canadian Parliament. (Years: 2017, 2018, and 2019.)

first plot, Figure 1, shows the number of times each party spoke; it shows how the Liberal, Conservative, and NDP parties dominate the speaking times (in order from greatest to least). All other parties overlap each other near the bottom of the graph.

Our second plot, Figure 2, took the ten most popular[3] topics discussed in the Canadian Parliament during 2018 and 2019. When considering some of these spikes in context of what was going on at the time, the results are not surprising. For example, two topics had a spike during the spring of 2019: Political Influence and SNC-Lavalin Group Inc.[4,5]

Taking these results, we decided to look into the co-occurrence of topics.

3.2.2. *Detecting co-occurrence of topics.* The debates in the Hansard dataset we analyzed discussed a little over 7,000 significant topics. We counted the number of times every topic was mentioned during each single hour the Parliament was in session. The dataset that resulted from this counting procedure was both high dimensional and large in size.

[3]based on amount time spoken on floor; more time equals more popular

[4]*Wikipedia*, `https://en.wikipedia.org/wiki/SNC-Lavalinaffair`

[5]When considering the top 20 topics, there is also a spike for the topic "Aboriginal People".

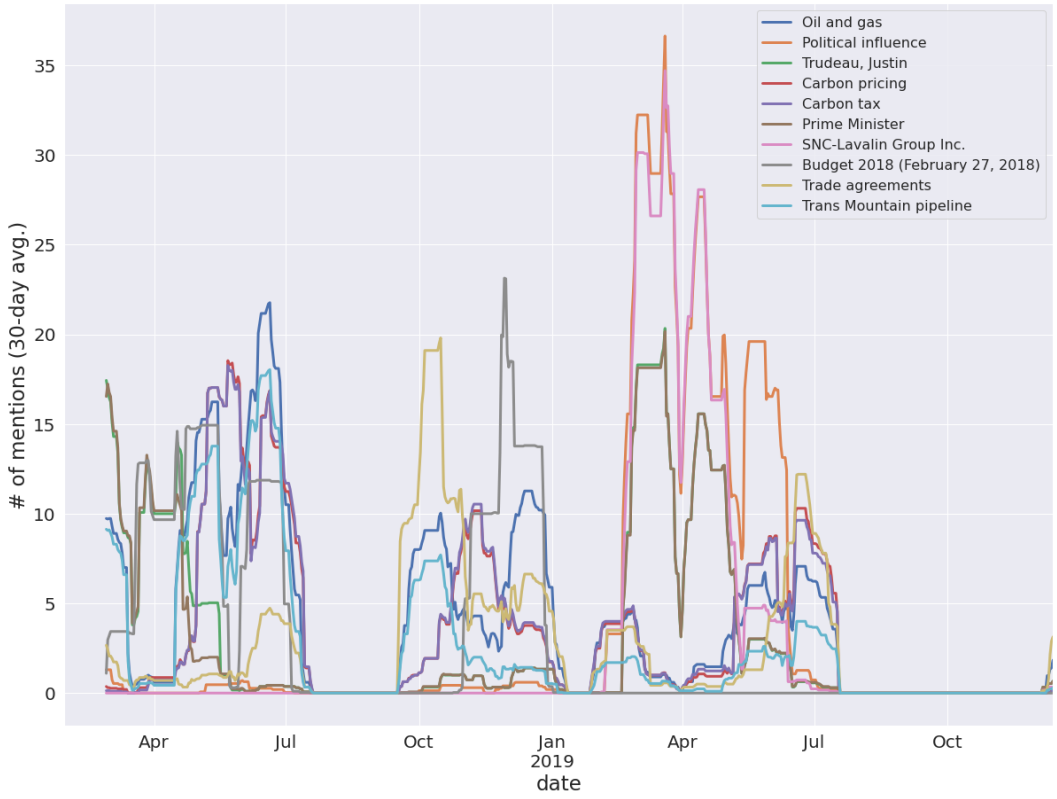
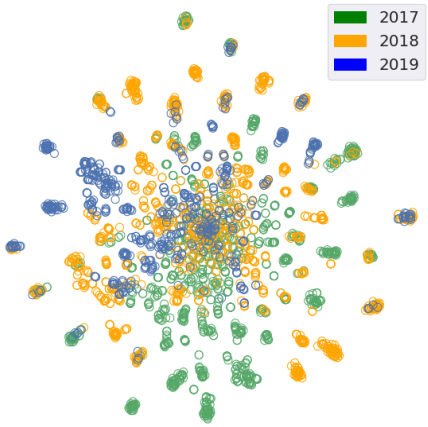


FIGURE 2. A plot of the 30-days moving average of the number of times each of the top-10 topics was mentioned on the floor of the Canadian Parliament. (Years: 2018, 2019.)

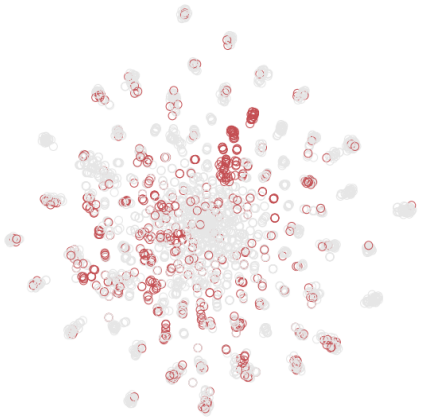
Due to time constraints and computational limitations, we restricted our attention to qualitatively analyzing the dataset through visual means. We used the t-distributed stochastic neighbor embedding algorithm (t-SNE) to create two-dimensional visualizations of our dataset that we could easily interact with. The t-SNE is a nonlinear dimensionality reduction technique that embeds high dimensional datasets in lower dimensions. The algorithm tries to keep close points in the original dataset also close in lower dimensional embedding. It is considered the state of the art in visualizing very high dimensional data.

Figure 3, which contains four plots, show some of our results from using t-SNE. Since the dataset was undersampled into hours for these plots, each dot in the plots below is a vector that has the number of times each (significant) topic was mentioned during one of those hours; multiple topics can be represented by the same dot. The top-left plot (Figure 3a) colors each dot by year. This top left plot will be used with each other plot to understand our results.

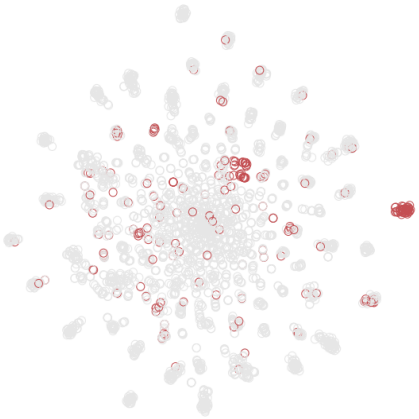
The topics will have different levels of "spread" over the three years. Figure 3b shows the trade agreements have some clustering but is relatively evenly distributed. However, Figure 3d almost entirely clustered in 2019, showing that it was a topic brought more



(A) The years 2017, 2018, and 2019 highlighted.



(B) The topic "trade agreements" highlighted.



(C) The topic "imprisonment and prisoners" highlighted.

(D) The topic "criminal prosectution" highlighted.

FIGURE 3. The t-SNE map of the topics discussed in the Hansard dataset of the Canadian Parliament for the years 2017, 2018, and 2019.

often in 2019 than the previous two years. Looking at all four plots, we can see not only the occurrence of specific topics, but also when two topics may co-occur. For instance, Figures 3b and 3c have a bit of overlap in right about the center, while Figures 3c and 3d has less.

3.3. Conclusion and Future Directions. Using both a line graph and a t-SNE plot to look at occurrence–and co-occurrence–gives us a better idea of topic trends in the Canadian Parliament. An interesting direction to take this is to compare these occurrences with the occurrences of bills introduced in Parliament around the same time; the same thing can be done with bills passed. This can give some perspective on how the speaking time on a topic may affect a bill being introduced and/or passing. This idea can be taken a step further by focusing on specific legislators who spoke on a topic. Is there a correlation with speaking time and the number of bills introduced? What about frequency a legislator spoke on a topic? Or number of bills passed? To continue this analysis, we need to gather more data about legislation from the Canadian Parliament from 2017-2019, making sure this includes the status of a bill, the (co-)sponsors of the bill, when the bill was introduced, and more.

4. U.S.

The Library of Congress was established in 1800 as a collection of books intended for the use of Congress. Despite its humble beginnings, haunted by a lack of funding, space shortages and fires, the Library has acquired a "symbolic role as a repository and promoter of the democratic tradition"[6]. In recent years, by collaborating with the House, the Senate and U.S. Government Publishing Office [7], the Library provides an official online source for legislative information[8]. The site includes a portrait of the current legislators in the form of aggregated data. Information on each member of Congress includes: name, party, number of sponsored/co-sponsored legislation, number of legislation by policy area, and more.

4.1. Motivating Questions. We attempt to use this compact data set in order to answer the following questions:

(1) How do politicians cluster around policy areas?
(2) How does this clustering, and other pertinent data, relate to a congress member's ability to pass bills to law.

4.2. Methodology and Results. As mentioned previously, our focus will be on policy area clustering. However, since the end goal is to study legislator performance, we begin by presenting some clear trends that were observed in our initial exploration. Recall that our data contained a portrait of the 437 representatives and 100 senators in the 116^{th} Congress. For each legislator, we were able to compute the percentage of bills that became law out of all the bills that have been sponsored or cosponsored by this given legislator. We will use this percentage as a measure of political performance.

A rapid overview revealed two factors which influence the percentage of bills that become law for each legislator. Firstly, we found that representatives[9] who worked in the house in 2019-2020 are more likely to (co)-sponsor bills that become law; this may need to

[6] *History of the Library*, `https://www.loc.gov/about/history-of-the-library/`

[7] *HLibrary of Congress to retire Thomas,*Adam Mazmanian `https://fcw.com/articles/2016/04/28/thomas-loc-retired.aspx`

[8] See `congress.gov`.

[9] representative here meaning a member of the House of Representatives

be compared to previous representatives to make further conclusions. Additionally, (co)-sponsorship from an experienced or "seasoned" legislator increases the probability of a bill becoming law. These findings can be visualized in Figure 4.

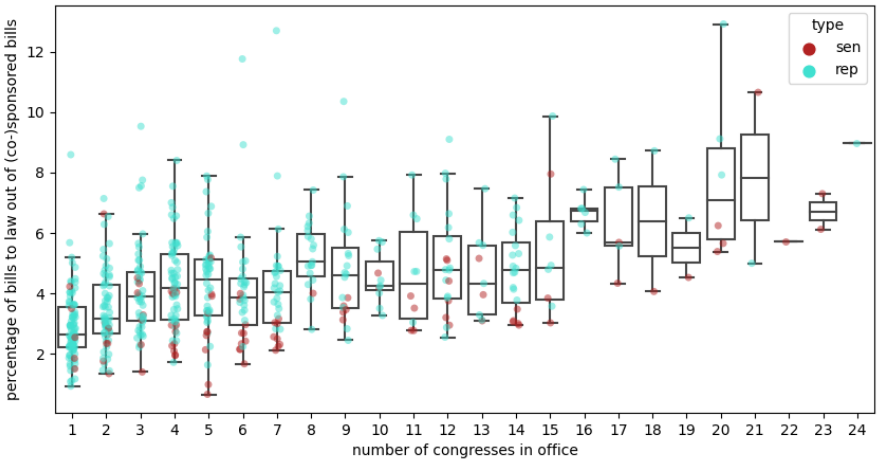


FIGURE 4. Percentage of bills that became law in terms of number of congresses spent in office.

Other features, such as party affiliation, do not appear to have an obvious effect on legislator efficiency. In order to find new trends, we must mine the data to find what lies beneath the surface.

4.2.1. *Legislators and policy areas.* When exploring the data, we chose to focus on the legislators' interest in various policy areas. To account for factors such as varying number of years in office, we consider the following ratio:

$$\frac{\textit{number of bills (co)-sponsored in a given policy area}}{\textit{total number of bills (co)-sponsored}}$$

Since some policy areas inherently require more bills than others, we then subsequently normalize these ratios.

Remark. We excluded 4 legislators from our data sets since they were newly elected and had therefore had little information. We set a cut-off off 100 bills total (co-)sponsored.

We begin to visualize our data by using t-SNE[10]. This method allows us to plot our data in a two dimensional space in such a way that preserves *nearness*. This plot is presented in Figure 5a, where we coloured the points by party. The separation between Democrats and Republicans demonstrates that policy area clustering is a reasonable indicator of political position.

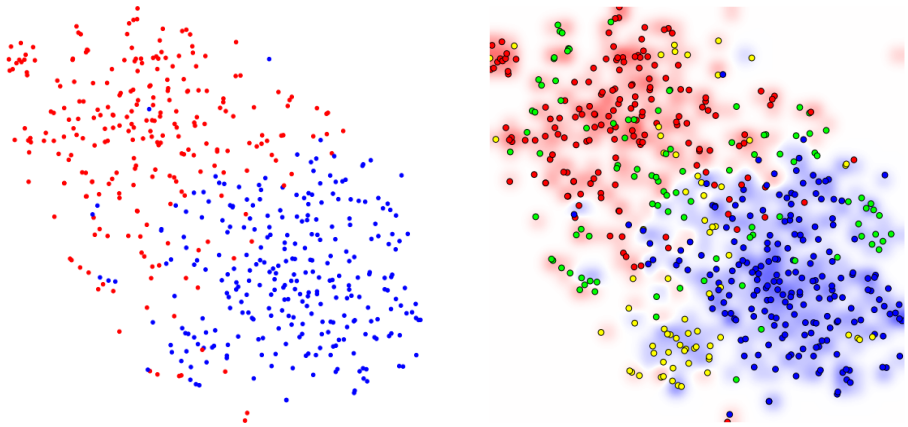
[10] `scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

We cluster our data using two methods: k-means[11] and gmm[12] (Gaussian mixture model). In order to maintain a balance between how much variation can be attributed to the clusters and our ability to interpret these clusters, we chose to cluster our data into four groups. After comparing our results for each method, we determined that the gmm yields better results. The kmeans algorithm produced clusters with a highly uneven number of legislators; there was a cluster containing only a single legislator.

In order to further refine our clustering, we chose to only consider a subset of policy areas. By reducing the number of features, we improved some clustering scores. This was done through feature selection.

4.2.2. *Feature Selection.* To determine which policy areas, we would re-run gmm while employing a greedy algorithm. Considering two clustering scores (Silhouette[13] and Calinski-Harabasz[14] score), we ran two greedy algorithms that, at any given step, pick the best feature to add. We plotted the scores against the added features at each step and select our desired features accordingly. Finally, we considered those policy areas that were selected with the greedy algorithm for both the Silhouette score and the Calinski-Harabasz score.

Clustering according to the selected data yields four groups of legislators. We obtained a sense of how well our data is clustered through another t-SNE visualization. In Figure 5b, we coloured our data to represent each cluster. The points are superimposed onto a heat-map based on Figure 5a.



(A) Democrats in blue Republicans in red

(B) Groups 1, 2, 3, 4 are respectively red, green, blue and yellow

FIGURE 5. t-SNE plots

[11] scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[12] scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

[13] scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouettescore.html

[14] scikit-learn.org/stable/modules/generated/sklearn.metrics.calinskiharabaszscore.html

In order to understand each group, we present Figure 6 which depicts the mean ratio of (co-)sponsored bills in each cluster.

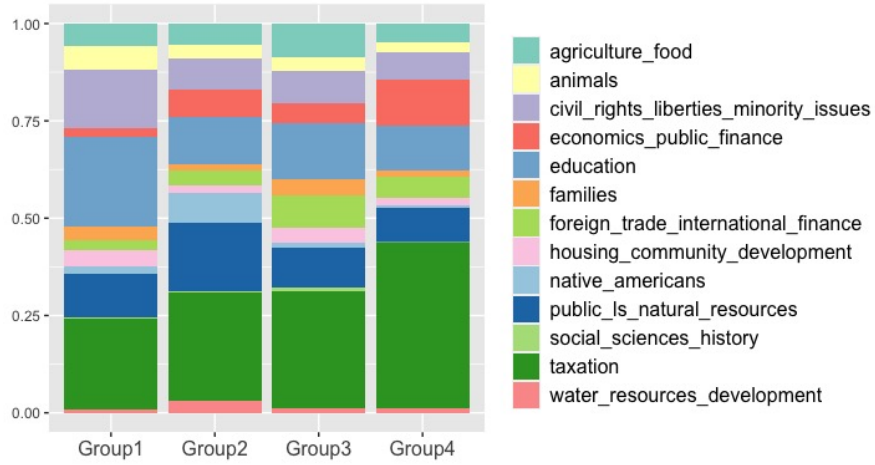


FIGURE 6. Average ratios of each policy area (feature selection)

By excluding senators, we can present the data geographically. In Figure 7, we colour each congressional district according to the cluster of the corresponding representative.

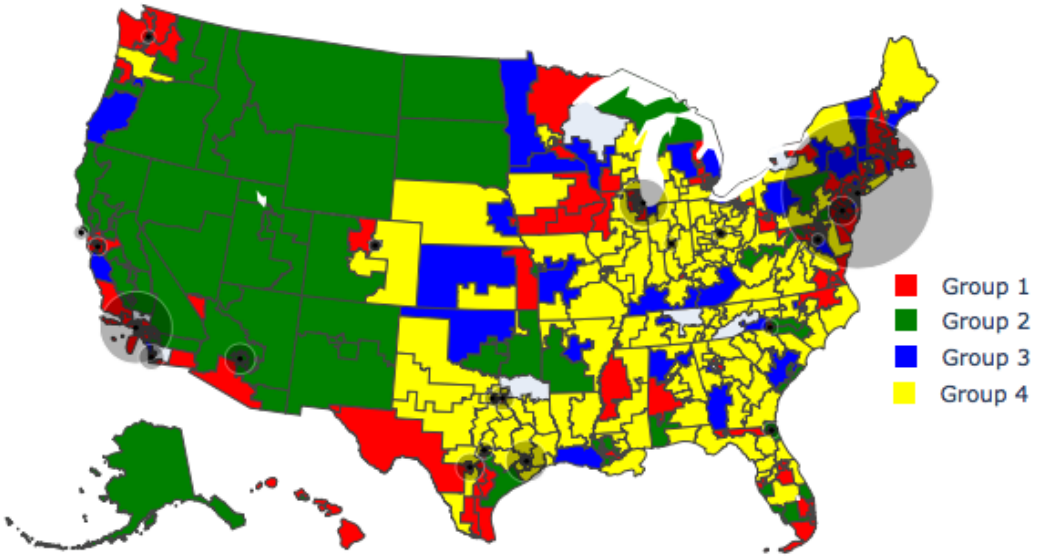


FIGURE 7. Map of House representatives by group

In Figure 7, we also included the location of the 20 largest U.S. cities with a translucent disk representing their population. The largest U.S. cities almost all fall within Group 3 (blue). This might be due to the fact that urban areas tend to have similar policy interests.

4.2.3. *Clusters and legislator performance.* Taking a look at the mean percentage of bills that became law for each cluster, we obtain minimal variability. However, it makes sense to evaluate cluster performance based on each policy area. To this end, we consider a data set containing information on bills introduced to congress since January 3rd 2019. Our data set contains information on almost 15 000 bills, including their sponsor, topic and current status (e.g. introduced, passed House, became law). For each topic, we group our data by clusters and evaluate the percentage of bills that became law. There are significant differences between the percentage of bills that became law in each cluster.

Out of the bills analyzed, only 156 have become law. Due to the small sample size, it is premature to draw any conclusions without considering information like the average number of bills passed in Congress. For this reason, we only summarize our results for what has historically been most popular policy area: health (see Table 1).

		Group 1	**Group 2**	**Group 3**	**Group 4**
Health	*total*	759	288	337	343
	to law	1	1	6	1
	percent	0.13	0.35	1.78	0.29

TABLE 1. Total number of bills sponsored in each group and amount that made it to law

Despite information on bill sponsorship in the health policy area not being included to produce our clustering, we observe a dramatic difference between Group 1 and Group 3.

4.3. CONCLUSION AND FURTHER DISCUSSION. Our results suggest that legislators' interest in policy areas is an indicator of multiple factors such as legislative performance, political position and even geographic location. We believe that our results are promising and that more work is in order. In particular, we would like to further refine our clustering by improving data preprocessing and considering other clustering methods. Moreover, we are interested in what can be found by analyzing bill specific information instead of aggregated data.

5. SUMMARY

We have shown two different approaches to answering the question if politicians cluster around topics. From our initial analysis, we do see indication that they do cluster around certain topics.

In the Canada data, you can see clusters around debate topics based on speaking time and when it occurred. The next steps is to start seeing how this data compares to Canada's legislation data, specific on bills presented and bills passed.

In the U.S. data, we see that there is clustering around parties, and from there some clustering around policy areas. However there are indicators that the clusters are not strictly down party lines and may be influenced by the region the legislator represents.

After our results were computed,[15], an article from the New York Times implies a similar trend[16].

There are many directions to go from based on our initial results. We have started looking at bills passed for U.S. data; this can be further explored by topic, legislator, region, etc. Additionally, we can use the Canada data with its legislative data to see if there are correlations. We could determine which topic debated was "most successful" and under that topic which legislator has the best record.

While analyzing legislator performance is further explored, it is important to consider the ethical implications. Detailed sports analysis has already affected players; some players in the NBA have avoided risky shots to keep good stats. We want to avoid this when analyzing legislators while still positively changing the how politics is currently viewed and discussed.

Acknowledgements

To succeed completion of this project is inseparable from the meticulous help of our mentors. We would like to express our deep gratitude to Dr. William Spat (the founder of IOTO International Inc.), Dr. Laleh Behjat (Professor of the University of Calgary), and Dr. Reza Peyghami (Professor of the University of York). All three gave us patient guidance, enthusiastic encouragement, useful comments, and interesting read materials to expand our view and deepen our understanding of the project topic.

We would also like to thank Justin Tendeck and the rest of the IOTO Team for our data support. They allowed use access to the data they mined, explained their API system, and provided us additional data when needed.

Finally, we want to thank to the PIMS Organization, with the help of organizations like Mitacs and Quansights, for creating this workshop. We especially appreciate Kristine Bauer and the rest of the Math$^{\text{Industry}}$ committee for organizing and running this successful workshop.

E-mail address: dana.berman@mail.mcgill.ca

E-mail address: jillian.glassett@wsu.edu

E-mail address: danyi2@ualberta.ca

E-mail address: diaaeldin.taha@aucegypt.edu

E-mail address: zhengx34@myumanitoba.ca

[15] On August 28th, 2020, We presented many of the results in this paper–plus some additional ones–in the PIMS Math$^{\text{Industry}}$ Showcase; this was a record session that may be posted on www.m2pi.ca

[16] *The New York Times*, "The True Colors of America's Political Spectrum Are Gray and Green", Tim Wallace and Krishna Karra, September 2, 2020, https://www.nytimes.com/interactive/2020/09/02/upshot/america-political-spectrum.html

MODELLING CANADIAN HEAVY CRUDE CONGESTION PRICING

MAHSA AZIZI, ERIK CHAN, XILAI FU, NIMA SAFAIAN, S. PARISA TORABI, AND WENNING WEI

Abstract. In this brief report, we establish a mixed-integer, linear programming model to estimate the congestion surcharge from price spread for crude oil prices over a fixed transport network with one or more consumer nodes. For the one node model, the North American crude oil market is discussed. Pseudo data simulated by an Ornstein-–Uhlenbeck stochastic process is used to study the multi-node model.

1. Introduction

Spatial price integration in oil commodity markets studies the price movement in the market between market participants that are geographically separated. Crude oil markets are generally considered well-integrated as the different types of oil (e.g., light and heavy oil) are fungible assets and there is a vast network of pipelines and train capacity available to deliver oil efficiently to its destination.

There are many different classifications of oil commodities on the global market distinguished by its viscosity and sulphur content. Among them, we focus on West Texas Intermediate (WTI), a high-quality oil priced out of Cushing, Oklahoma which serves as the primary benchmark in North America, and Western Canada Select (WCS), Canada's largest heavy oil stream.

Alberta produces approximately 4 million barrels per day (MMb/d) of crude oil of which approximately 3 MMb/d is classified as heavy oil. The majority of WCS is consumed in the Midwestern United States with the remaining excess stock destined for domestic consumption sent to the US Gulf Coast, see [4]. The US Gulf Coast is considered as a marginal consumption point where the last barrel is consumed. Canadian crude oil can reach the Gulf Coast through different pipeline options or rail from Edmonton or Hardisty, Alberta, see Figure 1. The transportation cost of the last barrel to the Gulf Coast sets the price of WCS.

The total off-take capacity of the pipeline from Alberta is on average less than the supply, this leads to "call-on-rail" (using expensive rail to transport oil) or shut-ins (mandatory curtailment of oil production). The price response to this supply-capacity difference is significant and appears as a regular breakdown in the price relationship between WCS and WTI that we define as a period of congestion pricing. This is exasperated by periods of congestion leading to the formation of transient "submarkets".

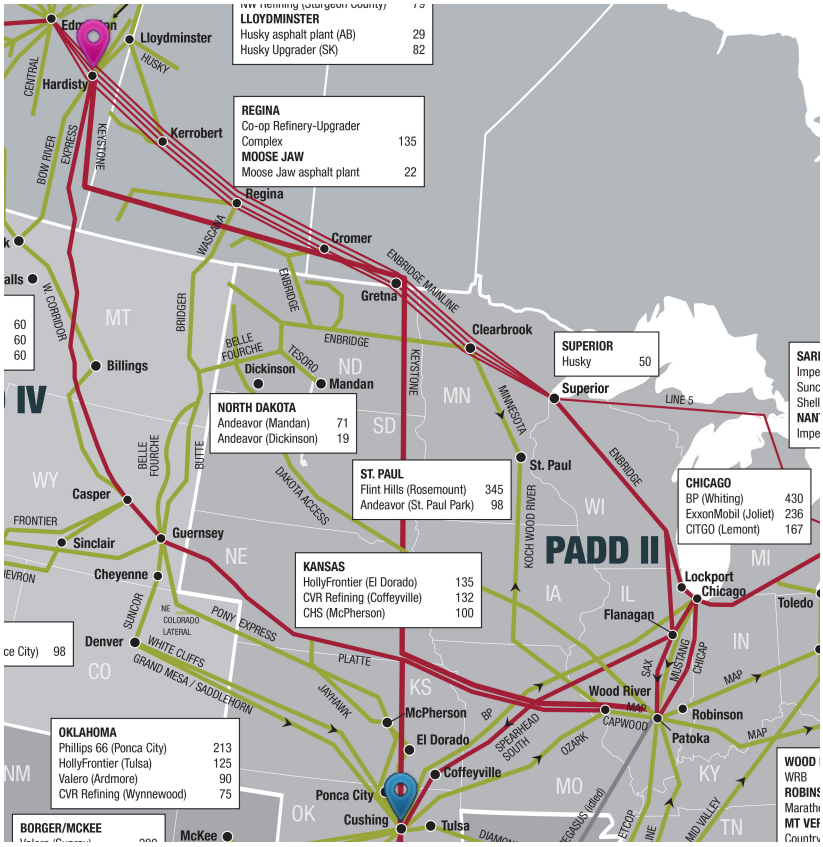


FIGURE 1. Map of crude oil pipelines [5]. The pink marker is located at Hardisty, Alberta and the blue marker is located at Cushing, Oklahoma.

1.1. **Documented congestion.** In the first half of 2018, western Canadian heavy oil production increased steadily while pipeline capacity did not increase. As a result, some production was shifted to rail, which is more expensive. Consequently, Canadian crude benchmarks faced large price discounts, and the difference between WCS and WTI increased compared to 2017 [7].

In the second half of 2018, the WCS–WTI differential widened more than normal. The reason behind that was the supply of western Canadian oil production rose to 4.30 MMb/d while takeaway capacity on existing pipelines remained constant at around 3.95 MMb/d. Moreover, the demand from the United States decreased due to the shutdown of some refineries in the Midwest, the largest export market for Canadian heavy crude oil [9], for maintenance.

Canadian crude oil exports via pipeline increased 3% from 3.1 MMb/d in 2018 to 3.2 MMb/d in 2019. At the same time, exporting crude oil via rail increased by 16% from 2018 to 2019. This growth in rail exports was largely due to pipeline capacity constraints in the Western Canada Sedimentary Basin [6]. Also, on October 28,

2019, around 9,120 barrels of oil were leaked from the Keystone pipeline causing a shut down for ten days [3].

In 2020, due to the COVID-19 pandemic, activities such as travelling were restricted in countries around the world to reduce the spread of the disease. The demand for Canadian crude oil products decreased significantly and the Canadian energy sector shut down production as low crude oil prices persisted [8]. The reduced supply removed pipeline capacity constraints in the transportation network and reduced the price spread between WTI and WCS as a consequence.

1.2. **Main result.** In this report, in collaboration with Cenovus Energy Inc., we build a mixed-integer linear (MIL) programming model to estimate the congestion surcharge from the price spread between different geographic regions over a transportation network connecting consumers and producers.

We start with a baseline model with a single consumer node that only considers the price spread between Hardisty and Cushing in 2018–2020. Taking Hardisty as the producer node and Cushing as the consumer node, we detect periods of congestion and the associated surcharge during this period and show that this model matches documented periods of congestion well. We then extend the model to multiple consumer nodes. Due to a lack of data, we simulate price data using an Ornstein–Uhlenbeck (OU) stochastic process (calibrated using historical WTI–WCS spot prices) over a network of consumers (refineries) in the absence of sufficient real-world data, and estimate the time and value the congestion surcharge.

This report consists of the following sections. In section two, a mixed-integer, linear programming model is established for a transportation network to estimate congestion. In section three, the one-node model is used to study the North American crude oil market for crude oil travelling from Alberta to the United States. In section four, a general multi-consumer model is applied to simulated data from an OU process. In section five, a summary of the results and possible future research are discussed.

2. A mathematical model for congestion surcharge

In [2], Birge et al. shows that in an equilibrium integrated market, the price of a commodity at different locations can be decomposed into a baseline price for the commodity, a transportation cost, a congestion surcharge, and a variable contained by a *neutral band* of values the local equilibrium price can take without exhibiting arbitrage due to transportation costs, see Lemma 2.1.

We denote by $\mathcal{T}$, a set of times measured in days, and $\mathcal{S}$, a set of consumer nodes over a transportation network connecting the nodes $\mathcal{S}$ to some collection of producers. Note that each consumer node may represent a cluster of nodes in close spatial proximity, such as all consumers residing in the same city.

Lemma 2.1 ([2, Proposition 3])**.** *The set of equilibrium prices* $\{\lambda_s^t,\ \forall s \in \mathcal{S}, \forall t \in \mathcal{T}\}$ *over a market with fixed transportation network structure and transportation (link) costs can be decomposed into*

$$\lambda_s^t = \eta^t + \rho_s + \varepsilon_s^t + \omega_s^t, \quad \forall s \in \mathcal{S}, t \in \mathcal{T},$$

In this lemma, $\varepsilon_s^t \in [-\alpha_s, \alpha_s]$ represents the neutral band, η^t is the baseline price for the commodity taken to be the spot price at a neutral node unaffected by congestion, and $\omega_s^t \geq 0$ is the congestion surcharge.

In our model, we set η_t to be the equilibrium price of the producer (assumed to be at a single node). Then, the formula in Lemma 2.1 reduces to $\xi_s^t = \rho_s + \varepsilon_s^t + \omega_s^t$ where $\xi_s^t = \lambda_s^t - \eta^t$ denotes the price spread between consumer s and producer.

Therefore, to estimate the congestion surcharge in the transportation network, we construct the following mixed-integer, linear programming problem.

$$
\begin{aligned}
\text{minimize: } & \sum_{s\in\mathcal{S}} \alpha_s \\
\text{subject to: } & \xi_s^t = \rho_s + \varepsilon_s^t + \omega_s^t, \quad \forall s \in \mathcal{S}, t \in \mathcal{T}, \\
& -\alpha_s \leq \varepsilon_s^t \leq \alpha_s, \quad \forall s \in \mathcal{S}, t \in \mathcal{T}, \\
& 0 \leq \omega_s^t \leq \gamma_s^t M, \quad \forall s \in \mathcal{S}, t \in \mathcal{T}, \\
& \varepsilon_s^t \geq \alpha_s - (1-\gamma_s^t)M, \quad \forall s \in \mathcal{S}, t \in \mathcal{T}, \\
& \psi^t \leq \sum_s \gamma_s^t \leq |\mathcal{S}|\, \psi^t, \quad \forall t \in \mathcal{T}, \\
& \sum_t \psi^t \leq \beta T, \quad \forall t \in \mathcal{T}, \\
& \psi^t, \gamma_s^t \in \{0,1\}, \quad \forall s \in \mathcal{S}, t \in \mathcal{T}.
\end{aligned} \tag{2.1}
$$

Here ξ_s^t denotes the price spread between the price of the commodity at the consumer node s and the producer, M is a sufficiently large upper bound for the congestion surcharge, ψ (resp. γ) is an indicator function taking the value 1 representing congestion in the network (resp. representing congestion at s) and taking the value 0 in the absence of congestion (resp. in the absence of congestion at s). Finally, $\beta \in [0,1]$ is a fixed parameter set by the user that represents the proportion of the time that congestion is allowed to manifest, i.e., time periods when the surcharge term ω_s^t can take positive values.

When there is only one consumer node in $\mathcal{S}$, the above mixed-integer, linear programming problem is simplified to

$$
\begin{aligned}
\text{minimize: } & \alpha \\
\text{subject to: } & \lambda^t = \rho + \varepsilon^t + \omega^t, \quad \forall t \in \mathcal{T}, \\
& -\alpha \leq \varepsilon^t \leq \alpha, \\
& 0 \leq \omega^t \leq \psi^t M, \\
& \varepsilon^t \geq \alpha - (1-\psi^t)M, \\
& \sum_t \psi^t \leq \beta T, \\
& \psi^t \in \{0,1\},
\end{aligned} \tag{2.2}
$$

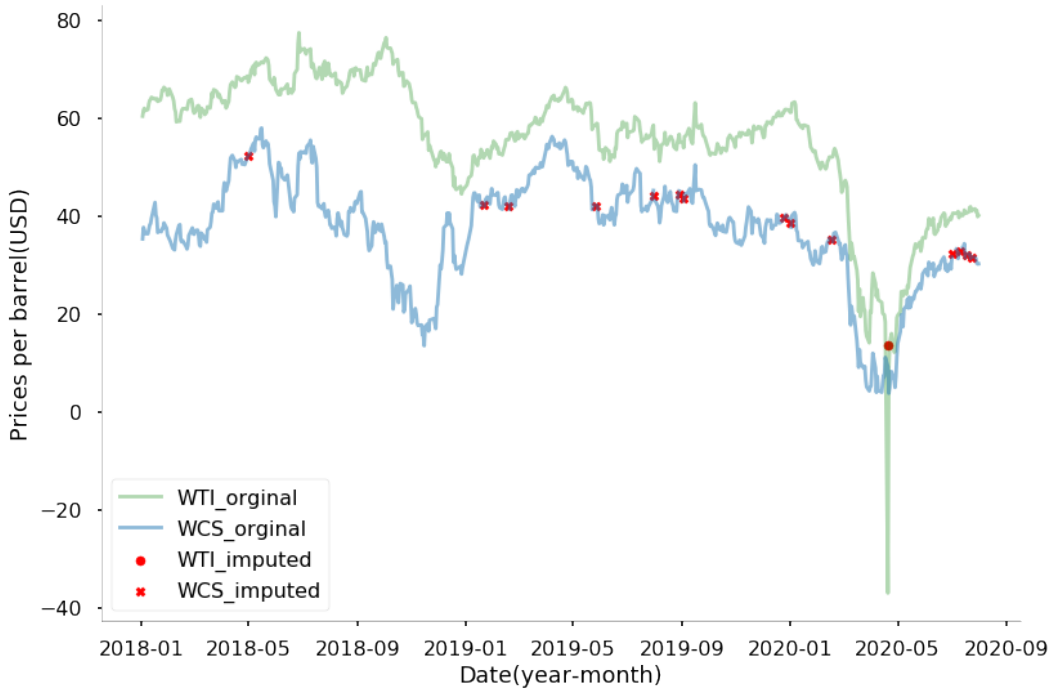


FIGURE 2. The price of WTI and WCS from January 2018, to August 2020. Missing data in WCS is filled-in using the average of its nearest neighbours. An outlier (-37USD) in WTI at Apr. 20, 2020 is replaced by interpolation.

i.e., γ is no longer necessary since the parameter determines which node experiences congestion conditional on congestion occurring somewhere on the network.

3. Congestion surcharges in the crude oil market

In this section, we use the above programming model to estimate the congestion surcharge in the crude oil market in North America. We analyze WCS from Hardisty, Alberta and WTI from Cushing Oklahoma. We take Hardisty as the producer node with WCS as the local price, and Cushing to be the consumer node with a local price set to be 5USD lower than the WTI spot price to account for the difference between heavy and light oil. See The spot price data between WCS and WTI from January 2018 to August 2020 can be found in Figure 2.

Using model (2.2), we set the price spread to be WTI$-$WCS-5 as described above, set $M = 100$ as an arbitrarily large upper bound for the congestion surcharge, and for $\beta \in \{0.2,\ 0.4,\ 0.6,\ 0.8\}$, we get the congestion surcharges shown in Figure 3. The pink shaded regions represent time periods with documented pipeline disruption or "call-on-rail", while the green shaded region corresponds to a period with sufficient pipeline capacity but with reduced demand. Therefore, we can see

by visual inspection that selecting $\beta = 0.4$ or $\beta = 0.6$ the congestion surcharge estimated by the model seems to match the historical market behavior well.

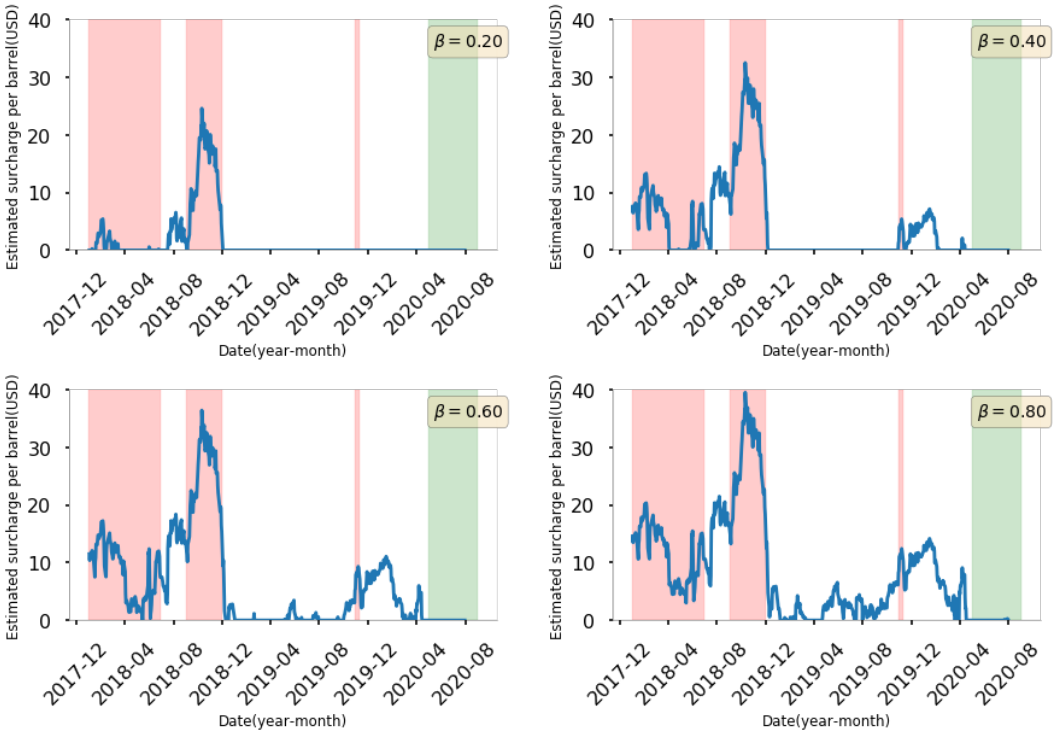


FIGURE 3. Estimated congestion surcharge for a single consumer node with different choices of β. Periods of historical congestion are denoted by the pink shaded regions and a period of decreased demand is denoted by the green shaded region.

4. Multi-node model with pseudo data

In this section, we estimate the congestion surcharge in a transportation network with one producer node and several consumer nodes. Due to the lack of data, we simulate price spreads by using three correlated Ornstein–Uhlenbeck stochastic processes for the consumer nodes. In this section, we provide a brief description of the OU process and direct the reader to [1] for further details. We say a stochastic process is an OU process if it follows the stochastic differential equation,

$$\mathrm{d}x_t = \alpha(\mu - x_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t.$$

.

The parameters α, μ, and σ for the OU process are calibrated using the WTI–WCS spread spot price. Using the method of maximum likelihood estimation, we obtain

the following parameters: $\alpha = 0.0119$, $\mu = 16.275$, $\sigma = 2.053$. These parameters and three correlated Brownian motions following the correlation matrix

$$\text{corr} = \begin{bmatrix} 1 & 0.8 & 0.7 \\ 0.8 & 1 & 0.56 \\ 0.7 & 0.56 & 1 \end{bmatrix},$$

are used to simulate three sample price spreads for the three refineries and the one producer. See Figure 4.

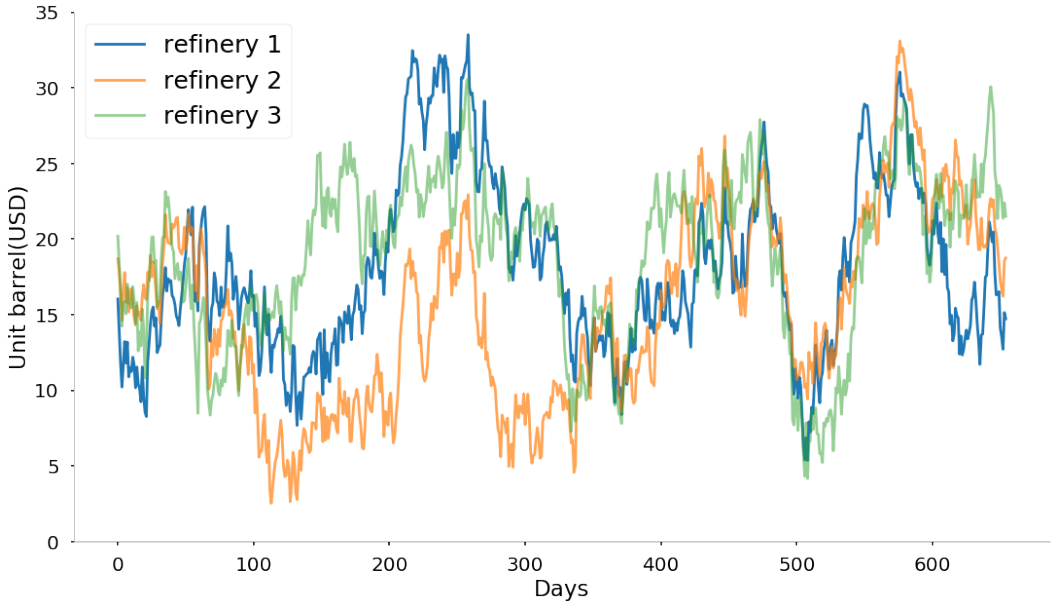


FIGURE 4. Simulated paths for three refineries (consumer nodes $s \in \mathcal{S}$)

Using model (2.1) with $\mathcal{S}$ containing three consumer nodes, the congestion surcharge for $\beta \in \{0.2, 0.4, 0.6, 0.8\}$ are simulated in Figure 5.

5. CONCLUSION

In this project, we were able to generate congestion premiums using a mixed-integer, linear programming model using the spread price between WTI and WCS as an input. These estimated congestion premiums are consistent with the amount the spot price increased during periods of historical congestion. Using these estimates, one could model the spot price by first simulating a baseline mean reverting process for the congestion-free price and then add the estimated surcharge. We further generalized the model to a multi-node model. This multi-node model can be used to further examine the congestion premium on more complicated networks and study the graph-theoretic shape of the crude oil market by adding and removing refineries to study the behaviour of congestion before and after changes are made to the network structure.

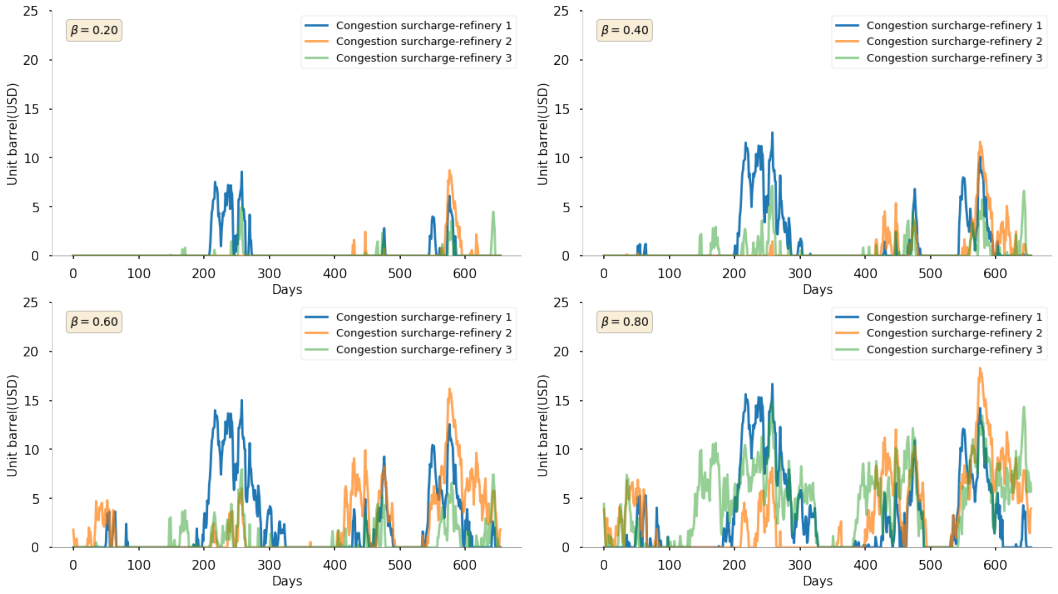


FIGURE 5. Multi-node congestion surcharges for three refineries with different choices of β.

ACKNOWLEDGEMENTS

We wish to thank Nima Safaian for his support and guidance throughout this project. We would like to further thank PIMS for organizing a wonderful workshop.

REFERENCES

1. M. T. Barlow, *A diffusion model for electricity prices*, (2002), Available At SSRN: https://ssrn.com/abstract=330729. 6
2. John R. Birge, Timothy C. Y. Chan, J. Michael Pavlin, and Ian Yihang Zhu, *Spatial price integration in commodity markets with capacitated transportation networks*, available at SSRN: https://ssrn.com/abstract=3544530. 3
3. CNN, *The latest Keystone Pipeline oil leak is almost 10 times worse than initially thought*, available online at: https://www.cnn.com/2019/11/20/us/keystone-pipeline-leak-10-times-worse-trnd/index.html. 3
4. D. Hackett, L. Noda, and S. Grissom, *Pacific basin heavy oil refining capacity*, SPP Research Papers, School of Public Policy, University of Calgary, 2013. 1
5. The Canadian Association of Petroleum Producers, *Canadian pipeline map*, available online at: https://www.capp.ca/explore/oil-and-natural-gas-pipelines/. 2
6. Canadian Energy Regulator, *Crude oil annual export summary – 2019*, available online at: http://www.cer-rec.gc.ca/nrg/sttstc/crdlndptrlmprdct/stt/crdlsmmr/crdlsmmr-eng.html. 2
7. ———, *Crude oil markets*, available online at: https://www.cer-rec.gc.ca/nrg/ntgrtd/ftr/2018/chptr2-eng.html?=undefined&wbdisable=true. 2
8. ———, *Market snapshot: Almost one million barrels per day of western Canadian oil supply was cut by mid-May 2020 because of low global oil prices*, available online at: https://www.cer-rec.gc.ca/nrg/ntgrtd/mrkt/snpsht/2020/06-04lmstnmllnbrrlsct-eng.html. 3

9. ———, *Western Canadian crude oil supply, markets, and pipeline capacity*, available online at: https://www.cer-rec.gc.ca/nrg/sttstc/crdlndptrlmprdct/rpt/2018wstrncndncrd/index-eng.html. 2

University of Saskatchewan, Dept. of Computer Science, Thorvaldson Building, Saskatoon SK, S7N 5C9, Canada
E-mail address: mahsa.azizi@usask.ca

University of Calgary, Dept. of Math. and Stat. 2500 University Drive NW, Calgary AB, T2N 1N4, Canada
E-mail address: erik.chan@ucalgary.ca

University of Alberta, Dept. of Math. and Stat. 11840 90 St NW, Edmonton AB, T5B 3Y6, Canada
E-mail address: xilai@ualberta.ca

Brock University, Dept. of Physics. 500 Glenridge Avenue, St. Catharines On, L2S 3A1, Canada
E-mail address: sparisatorabi@gmail.com

University of Calgary, Dept. of Math. and Stat. 2500 University Drive NW, Calgary AB, T2N 1N4, Canada
E-mail address: wenning.wei@ucalgary.ca

TRAFFIC MONITORING WITH DISTRIBUTED ACOUSTIC SENSING

B. CHAN, E. R. KORFANTY, S. NATAJ, J. PARK, B. PENG, AND J. ZHANG

INDUSTRY MENTOR:
Matt McDonald, Chief Technology Officer, Fotech

ACADEMIC MENTORS:
Prof. Parimala Thulasiraman and Ying Ying Liu
Department of Computer Science, University of Manitoba

ABSTRACT. Using vehicle traffic data collected with *Distributed Acoustic Sensing* (DAS) facilities from Fotech, we demonstrate that k-means clustering and Kalman filtering can be used to determine the locations of individual vehicles, and track the trajectories of vehicles over time. The implementation provided can handle traffic data for several vehicles travelling in the same direction, with no car passing another. We consider more complex traffic flow situations as future directions for improvement to our method.

1. BACKGROUND AND OVERVIEW

Smart city applications have experienced a notable increase in interest over the last few years. A smart city is a metropolis that employs a variety of electronic *Internet of things* sensors to collect data. The data is used to improve the operations, resources and services across the city [8]. One of the branches of smart city development consists of the development of real-time traffic monitoring with the aim of reducing traffic congestion. Processing of fiber-optic distributed acoustic sensing (DAS) data is in high demand within different smart city applications [15], [11]. DAS signal data measures the strain in fiber optic cables that are placed underground. For traffic monitoring applications, a fiber optic cable is run along a road. When vehicles pass by, mechanical vibrations are created in the ground, causing slight deformities in the fibre optic cable. These deformities cause a phase differences in the back-scattered light from laser pulses propagating through the fiber optic cable. The DAS system is able to detect these minute differences, and creates a signal linearly proportional to the strain in the cable at each point along its length [16]. DAS technologies have significant potential for traffic monitoring applications within a city. DAS has proven to be successful in the similar topic of train monitoring, though DAS signals from a road can be much more complex than DAS signals from trains. In [16], an algorithm was presented for using DAS signals to track the positions of trains over time, provided that there is a sufficiently large separation between trains

This work was funded by PIMS and Mitacs.

at all times. The algorithm applies machine learning techniques to detect the train edges, uses a distance-based optimization method for assigning train edges to train objects, and uses a Kalman filter to track the train objects over time. DAS signals from vehicles on a road, however, often appear in greater quantity, and in closer proximity than those from trains. This presents challenges in the application of this algorithm to vehicle traffic monitoring.

The goal of this project is to investigate a new methodology for identifying and tracking vehicles, by making use of clustering algorithms and Kalman filtering techniques in data mining and signal processing literature. We hope to provide a foundation which can be extended upon to handle more complex traffic flow settings. In Section 2 the proposed methodology for the solution of this problem is discussed. In Section 3, the result of implementing this approach is presented. DAS data is provided by Fotech, and Python libraries and packages are used for coding. In Section 4 the future direction and possible improvements are discussed. Finally, a summary of the project is given in Section 5.

2. Approach and Methodology

Our proposed methodology consists of three main steps: locating peaks in the DAS signals, using clustering methods to determine vehicle positions, and tracking these vehicle positions over time. Investigations relating to these three steps are presented in Section 3. Here, we provide an overview of the process by which these three steps can be combined into a real-time traffic monitoring method, summarized in Figure 1.

DAS signals are read one at a time. A single DAS signal reading is referred to as a *shot*, and consists of one sample from each position along the fiber optic cable, for a given instant in time. A fixed number of shots, which will be referred to as the *window size*, are analyzed at any given time. When a new shot is read, the analysis window is shifted by one time step, in order to include the most recent *window size* many shots.

Kalman filter models can be used to track vehicle positions and velocities over time. In the case of more than one vehicle, we propose the use of more than one Kalman filter model simultaneously. For each of the vehicles identified, a tracker object is created, and equipped with a Kalman filter model for that vehicle. These tracker objects will be referred to as *Kalman trackers*. Each Kalman tracker must be initialized with the position of the corresponding vehicle. Initial velocities and positions are set to null, and are corrected as new data becomes available in each time step.

In order to use DAS signal data to identify the positions of vehicles at a given point of time, peaks in the signal strength must first be located. The goal of this procedure is to separate the signals representing vehicles from background noise. The detected peaks can then be further analyzed to infer vehicle positions.

A clustering algorithm can be used to identify individual vehicles using the detected peaks data. Many clustering algorithms provide a point, or points, representative of each cluster. These are often referred to as *cluster centers*, or *cluster centroids*. Cluster centroids can be excellent candidates for vehicle positions.

To track the positions of identified vehicles, the locations of cluster centers can be used to update the Kalman trackers. If there is more than one Kalman tracker, an optimization problem must be solved in order to associate a given cluster center with an existing tracker. The strategy for assignment of cluster centers to trackers is an important aspect managing more than one Kalman tracker simultaneously.

Finally, the process of reading a shot, shifting the analysis window, finding and clustering peaks, and assigning cluster centers to trackers for trajectory correction, can be repeated in a loop for real-time tracking of vehicle positions.

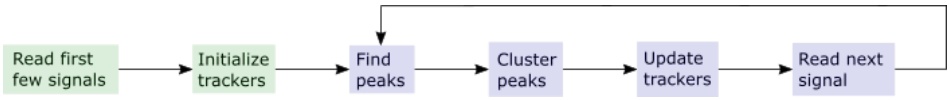


FIGURE 1. Flowchart of the proposed method for identifying and tracking vehicles from DAS signals in real-time.

3. Implementation and results

In our approach, we analysed small subsets of the data collected from Fotech instruments, to reduce the computational overhead. When comparing clustering algorithms, we looked at vehicle trajectories for both smaller subsets of 10 shots, and larger subsets of a few hundred shots. When using Kalman trackers to trace out vehicle trajectories, we used circular shifting, and analyzed 5 to 20 shots worth of data at a time. This technique is used to achieve the results in Section 3.3, where an example implementation is provided to illustrate how our methodology can be used to handle simultaneous tracking of several vehicles.

In preparation, we first detected peaks across the aforementioned segments of data, as described in Section 3.1. In section 3.2, clustering algorithms are employed to cluster the detected peaks. Furthermore, we discuss whether the resulting clusters align with our objectives. In particular, we investigate whether the resulting clusters effectively distinguish one vehicle from another. Using these observations, we make conclusions about which clustering methods may be more effective for identifying locations of vehicles from peaks in DAS signals.

3.1. **Peak detection techniques.**

The elastic strain wave found in each shot contains noise which is unnecessary for our purpose [10]. Hence, only the positions at which a compression wave obtains a local extremum exceeding a certain threshold, which we call a peak, were processed through the clustering methods presented in the Section 3.2. Once we found the peaks in each shot (Figure 2), we collected these peaks over the time period of interest (Figure 3a and 4a).

There are various peak detection methods available on Python `scipy` library:

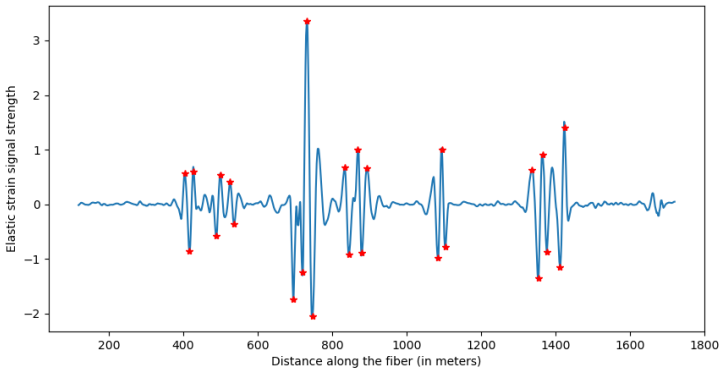
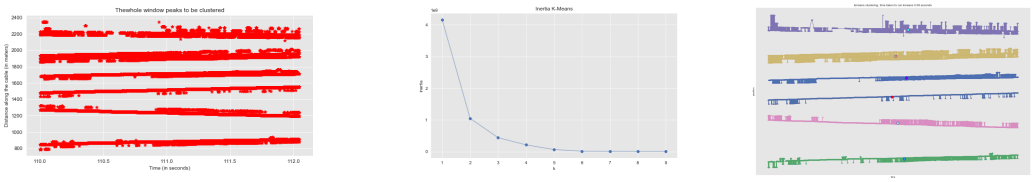


FIGURE 2. An example of the peaks (marked with red stars) on an elastic strain wave

- `scipy.signal.find_peaks_cwt`,
- `scipy.signal.find_peaks`,
- `scipy.signal.argrelextrema`.

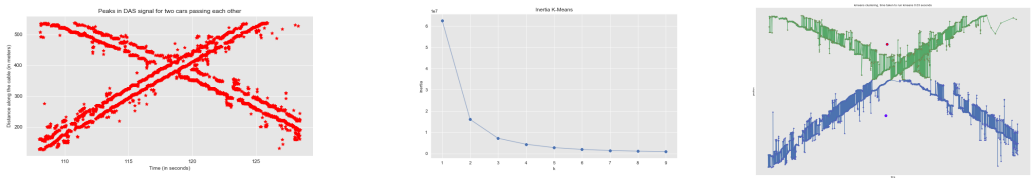
In our implementation, we used signal processing `scipy.signal.find_peaks_cwt` to find the peaks for the subsets of DAS signal data to be analyzed. This method is based on a wavelet transformation [4]. The algorithm performs the discrete convolution of the dilation and translation of the Ricker wavelet function with the raw data. We have tested various peak detection methods, including three listed above, but a thorough comparison between the peak detection methods is beyond the scope of this investigation, and is included in the future work for improving our implementation.

3.2. **Comparison of clustering algorithms.** After collecting the peaks data, we applied clustering techniques to partition the peaks data, in hopes of achieving partitions representative of individual vehicles. We required that the number k be equal to the number of cars in the data segment to be determined. The clustering used the data for the positions and times of peaks. We employed the Python scikit-learn library for implementing clustering algorithms [12]. One promising approach to clustering for this DAS application is k-means clustering. Although this algorithm is computationally intensive [6], implementing it is straightforward. The first difficulty in applying any clustering algorithm is to obtain a priori knowledge of the number of clusters in each section of streaming data. To overcome this challenge, we apply the algorithm in a loop to compare the performance of the model for different numbers of clusters. In other words, we run the algorithm for different values of k, starting at $k = 1$ and increasing k by one each iteration before every re-run. The algorithm stops when the inertia attributes of the model no longer decrease significantly (inertia is sum of squared distances of samples to their closest cluster center). Figure 3b depicts the result of applying k-means clustering in a loop to find the best number



(A) The data set (B) The loop (C) k-means clustering

FIGURE 3. k-means clustering of the data set containing of 6 vehicles passing the road in the same direction



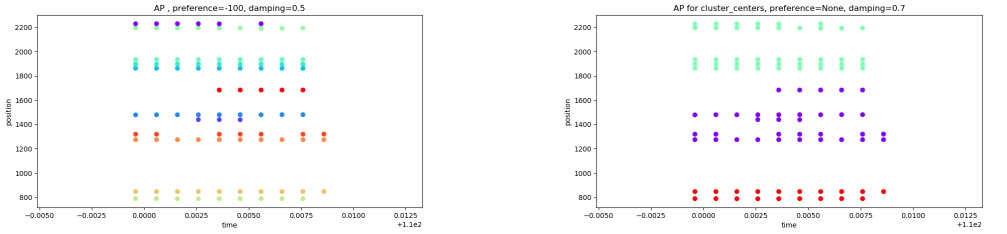
(A) The data set (B) The loop (C) k-means clustering

FIGURE 4. k-means clustering of the data set containing 2 vehicles passing the road in different directions

of clusters in the data set (Figure 3a). This data set represents six cars moving in same direction. Applying k-means in a loop shows that the best number of clusters is $k = 6$. Figure 3c shows that k-means algorithm successfully clusters the peak data into six clusters. After this, the centroids of the clusters can be used for in applying Kalman filters.

Another challenge in implementing k-means clustering in our data set is that the algorithm does not give an appropriate number of clusters for data segments presenting cars moving in different directions; it can not accurately distinguish vehicles passing each other on the road. This happened because k-means clustering usually cannot handle non-convex sets. By combining k-means with hierarchical clustering, we may overcome this difficulty [9]. Figure 4 shows the result of applying k-means in a data set with two vehicles passing other, for which it is not successful in finding the right number of clusters. However, this might be expected, because a pair of intersecting lines consists of a single connected component. More specifically, k-means is best suited to convex sets, but a pair of intersecting, thick lines is not convex.

Another potential algorithm to use for clustering in this DAS application is Affinity Propagation (AP). Unlike k-means, AP does not require the number of clusters to be determined before running the algorithm. This method finds *exemplars* (members of the input set that are representative of clusters). As an input, the algorithm requires some parameters to be provided: *similarity* (affinity function), *preference* and *damping factor*. Similarity defines how well-suited a point is to be the exemplar for another. Here we define it as negative Euclidean distance, which is the negative



(A) first time, using peak positions (B) second time, using center samples

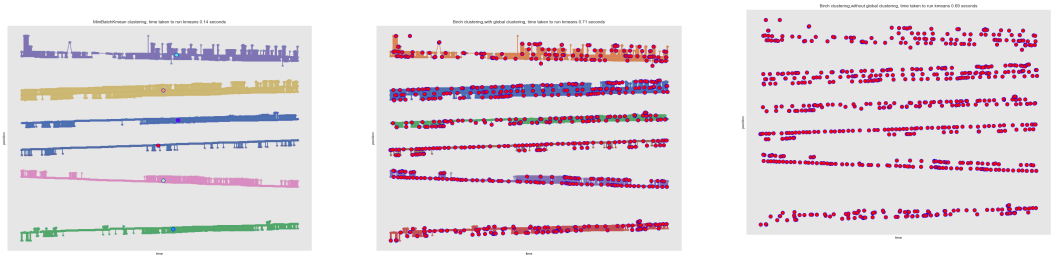
FIGURE 5. AP clustering of the data set containing 6 vehicles passing the road in the same direction

squared distance between the two data points. Preference represents the suitability of each data point to be an exemplar; points with larger preference values are more likely to be chosen as exemplars. Unfortunately, AP is very slow for a larger window size, such as the data shown in Figure 3a, due to the computational expense of the algorithm [13]. When examining AP, we apply it for a smaller data set. Figure 5a shows the result of applying AP in for a small window size, when the preference is any number between -10 and -100, and the damping factor is between 0.5 and 1. Since it gave more clusters than expected, we ran the algorithm a second time to merge clusters together using the cluster centers, presented in Figure 5b, which results in four clusters; this is still not representative of the number of vehicles, even for this small time window.

Some other clustering algorithms that could be useful in this DAS application are BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) and Mini Batch k-means methods. BIRCH applies hierarchical clustering on large data sets. This method is memory efficient; it can typically find a good clustering with a single scan of the data [17]. Mini Batch k-Means has less computational cost in comparison to k-means, as it uses random batches of data as opposed to using all available data [14]. The implementation of Mini Batch k-Means and BIRCH algorithms are available in the scikit-learn library [12]. The partial fit method of these models, included in the sklearn package, provides a way to do online clustering for steaming data. For references about online clustering, see [3] and the references therein. Figure 6 compares these two algorithms for the same data set as in Figure 3a.

Finally, it should be noted that for any of these clustering methods, tailoring the similarity measurement to this specific application may significantly improve results. For example, [7] provides a path-based similarity measure for AP. This, and the incorporation of velocities and accelerations, are potential improvements to the similarity measurements.

3.3. **The real-time process.** In this section, we provide an example to illustrate the feasibility of the methodology described in Section 2, making use of a circular shifting technique to mimic a real-time implementation. The proposed method was



(A) Mini Batch k-means (B) BIRCH (C) sub cluster centres in BIRCH

FIGURE 6. Mini Batch k-means and BIRCH clustering of the data set containing 6 vehicles passing the road in different directions

applied to a subset of DAS data consisting of five vehicles travelling along the fiber optic cable. The vehicles do no pass each other, and travel at similar velocities. The sample data consists of 5 seconds' worth of DAS signals measured over a 1529 meter length of fiber optic cable. For the particular DAS system, 5 seconds corresponds to 500 shots. A window size of 5 shots was used for analysis.

Peaks in the DAS signals were detected using `find_peaks_cwt`, and the method described in Section 3.1. Peaks for which the absolute value of the signal strength was less than 10 times the signal reading median were deemed to be noise, and were excluded from the computations.

The k-means clustering algorithm was used, with $k = 5$ set to match the number of vehicles. As illustrated in Section 3.2, the k-means algorithm was effective, efficient, and reliable. It was chosen for this implementation for these reasons, and to provide a clear example which can be used as a starting point for further investigations. See Section 4 for remarks on generalizing to a varying number of vehicles.

Five Kalman trackers were created, and the initial positions were set to be those of the cluster centers found from the peaks in the first 5 shots. The simultaneous use of more than one Kalman tracker comes with a challenge; namely, how does one decide which cluster center corresponds to which vehicle in subsequent time steps? In this example, the rule for deciding which cluster center should be used for updating which Kalman tracker was based on distance, and can be summarized as follows: If cluster center c is the closest cluster center to tracker t, and there is no other tracker t' which is closer to c than t, use c to correct tracker t. Otherwise, do not correct tracker t.

Figure 7 shows the results of this implementation. The five trackers are successful in tracing trajectories through collections of detected peaks corresponding to vehicles. This shows that a clustering-based approach to DAS signal data may provide a feasible solution for the tracking of vehicle positions and velocities. In particular, the use of cluster centers for the simultaneous management of several Kalman trackers was shown to be successful in this example, and the method has enough flexibility to serve as a starting point for treatment of more complex traffic flow situations.

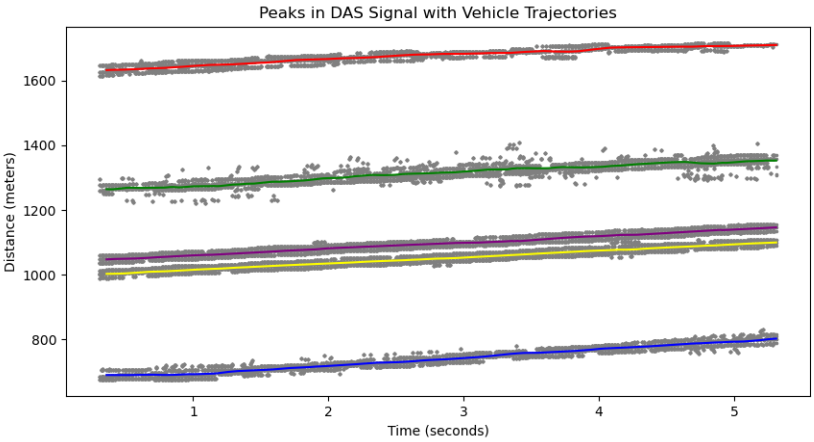


FIGURE 7. Trajectories of five vehicles tracked simultaneously using the proposed methodology. Detected peaks in DAS signal are plotted in grey, and the five colours show the Kalman tracker trajectories.

4. Discussion

In Section 3.3, we successfully implemented the proposed methodology using k-means clustering for subsets of data with simple traffic flow; specifically, for a fixed number of vehicles moving along the length of the fiber optic cable, in one direction, with no vehicle passing another. There are many ways in which this example can be improved. The main area of flexibility is in the choice of clustering algorithm. It may be beneficial to compare the performance of this tracking method when implemented using different similarity measurements, online clustering methods, hierarchical clustering methods, and methods which do not require the number of clusters to be specified. The pre-processing and peak-finding techniques used may affect which clustering algorithms are the most successful when tracking, and should be considered in a thorough comparison. It may also be interesting to consider the use of Hough lines instead of peaks [1].

It is possible that the most effective clustering algorithm could be one that requires the number of clusters to be specified. A possible method for using such an algorithm with a changing number of cars is the following: If the current number of trackers is k, compare the results of finding $k-1$, k, and $k+1$ many clusters, and pick the best result. Further investigation is required to determine a method for comparing clustering results, and deciding which should be used.

Another clustering algorithm of interest is CURE [5]. The benefit of CURE is that it allows for more flexibility in the shapes and sizes of the clusters, and is also robust in the presence of outliers. This could make CURE a suitable clustering algorithm to use for tracking, as peak detection methods can result in outliers.

When vehicles pass each other, the trajectories intersect, appearing as the crossing of lines of peaks in DAS signal. The algorithm for correcting trackers provided in

Section 3.3 does not include logic to consistently achieve accurate tracking in this situation. A potential improvement to the algorithm would be to weight corrections to trackers more lightly when approaching an intersection of trajectories.

It is also necessary to allow for the creation and deletion of trackers. This would allow the method to be applied to a situations with a varying number of vehicles. One possible approach is the following: If a cluster center is not assigned to a tracker, create a new tracker starting from that position. If a tracker is not updated for a specified number of iterations, delete the tracker.

5. Conclusion

Among the new techniques rising with the smart city trend, DAS technology provides a feasible solution for monitoring of traffic and transportation. In this work, we investigated a methodology for identifying positions of vehicles from DAS signal data, and tracking the positions and velocities of vehicles over time. The method for identifying positions was based on the application of clustering algorithms in the data mining literature to peaks in DAS signal, and the method for tracking the positions was based on the simultaneous use of several Kalman filter models. Peaks in signal data were detected using a standard signal processing tool based on a wavelet transform, and low intensity peaks were excluded as noise. The performance of various clustering algorithms was compared for a subset of peaks. k-means clustering was used to successfully identify and track five vehicles, in the case where no vehicle passes another. The implementation of the proposed approach showed how several Kalman filter models can be managed simultaneously, by using cluster centers to correct the trackers' trajectories. This investigation has provided the groundwork for the creation of a tool that uses DAS signals to locate and track individual vehicles. The method offers flexibility in the specific implementation, and has lead to many ideas for further investigation. In particular, the comparison of clustering algorithms suggested that Mini Batch k-means clustering can be applied to reduce computational cost of the implementation provided for k-means. We hope that this method will be improved upon, and lead to a new application of DAS technology to real-time traffic monitoring.

Acknowledgements

The team would like to thank Fotech and our industry mentor Matt McDonald for introducing and supporting our industry project and for his mentorship and advice throughout this project. In particular, we thank his guidance through the intricacies of the DAS system and for teaching us approaches for detecting vehicle locations, and tracking vehicle trajectories, and also for providing us the data and primary codes. We would also like to extend our thanks to our academic mentor Prof. Parimala Thulasiraman (Department of Computer Science, University of Manitoba) for her mentorship and introducing to us approaches that related our industry problem in the data-mining literates. Furthermore, we would like to thank Ying Ying Liu (Department of Computer Science, University of Manitoba) for providing us with resources that helped us with the project. Lastly, we thank PIMS, and Mitacs and

the organizing committee who put the PIMS $Math^{Industry}$ workshop together for making this investigation possible.

References

[1] N. Aggarwal and W. C. Karl. *Line detection in images through regularized hough transform.* IEEE Transactions on Image Processing, vol. 15, no. 3, pp. 582-591, 2006.

[2] F. Auger, M. Hilairet, J. M. Guerrero, E. Monmasson, T. Orlowska-Kowalska and S.Katsura. *Industrial Applications of the Kalman Filter: A Review.* IEEE Transactions on Industrial Electronics, vol. 60, no. 12, pp. 5458-5471, 2013.

[3] H. Cho and M. K. An. *Co-Clustering Algorithm: Batch, Mini-Batch, and Online.* International Journal of Information and Electronics Engineering, vol. 4, no. 5, 2014.

[4] J. Gregoire, D. Dale and R. Bruce van Dover. *A wavelet transform algorithm for peak detection and application to powder x-ray diffraction data.* The Review of scientific instruments, vol. 82, 015105, 2011.

[5] S. Guha, R. Rastogi, and K. Shim. *Cure: An efficient clustering algorithm for large databases.* Information Systems, vol.26, pp. 35-58, 2001.

[6] M. Inaba, N. Katoh, and H. Imai. *Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-Clustering: (Extended Abstract)*, Proceedings of the tenth annual symposium on Computational geometry (SCG '94), Association for Computing Machinery, New York, NY, USA, pp. 332–339, 1994.

[7] Y. Jiang, Y. Liao, and G. Yu. *Affinity propagation clustering using path based similarity.* Algorithms, vol. 9, no. 3, p. 46, 2016.

[8] C. S. Lai, Y. Jia, *et al A Review of Technical Standards for Smart Cities.* Clean Technologies, vol. 2, no. 3, pp. 290–310, 2020.

[9] R. Lior, and O. Maimon. *Clustering methods. Data mining and knowledge discovery handbook.* Springer US, pp. 321-352, 2005.

[10] H. Liu, J. Ma, W. Yan, W. Liu, X. Zhang and C. Li. *Traffic Flow Detection Using Distributed Fiber Optic Acoustic Sensing.* IEEE Access, vol. 6, pp. 68968-68980, 2018.

[11] A. Masoudi and T. P. Newson. *Contributed review: Distributed optical fibre dynamic strain sensing.* Review of Scientific Instruments, vol. 87, 011501, 2016.

[12] F. Pedregosa, G. Varoquaux, *et al*, *Scikit-Learn: Machine Learning in Python.* Journal of Machine Learning Research, vol. 12, 2012.

[13] R. Refianti, A. B. Multiara, and S. Gunawan. *Time affinity complexity comparison between affinity propagation algorithms.* Journal of Theoretical and Applied Information Technology, vol. 95, no. 7, pp. 1497-1505, 2017.

[14] D. Sculley. *Web-Scale k-Means Clustering.* Proceedings of the 19th international conference on World wide web (WWW '10), Association for Computing Machinery, New York, NY, USA, pp. 1177–1178, 2010.

[15] L. Shiloh, A. Eyal and R. Giryes. *Efficient Processing of Distributed Acoustic Sensing Data Using a Deep Learning Approach.* Journal of Lightwave technology, vol. 37, no. 18, pp. 4755-4762, 2019.

[16] C. Wiesmeyr, M. Litzenberger, M. Waser, A. Papp, H. Garn, G. Neunteufel, and H. Döller. *Real-Time Train Tracking from Distributed Acoustic Sensing Data*, Applied Sciences, no. 2, p. 448, 2020.

[17] T. Zhang, R. Ramakrishnan, and M. Livny. *BIRCH: an efficient data clustering method for very large databases.* Proceedings of the 1996 ACM SIGMOD international conference on Management of data, Association for Computing Machinery, vol. 25, no. 2, pp. 103–114, 1996.

TRAFFIC MONITORING WITH DISTRIBUTED ACOUSTIC SENSING

Brian Chan, Department of Mathematics, UBC
E-mail address: btchan@telus.net

Emily R. Korfanty, Department of Mathematics and Statistics, University of Victoria
E-mail address: emilykorfanty@uvic.ca

Sarah Nataj, Department of Mathematics, University of Manitoba
E-mail address: sarah.nataj@umanitoba.ca

Jaeun Park, Department of Mathematics, University of Wisconsin-Madison
E-mail address: park367@wisc.edu

Boya Peng, Department of Statistics, University of Alberta
E-mail address: pboya@ualberta.ca

Jianou Zhang, Department of Statistics, University of Saskatchewan
E-mail address: jiz127@usask.ca

AUTOMATED RECOGNITION OF ROAD LINES IN AERIAL IMAGES

CARLOS CONTRERAS, KERAN LI, YI SUI, LI WANG, TINGZHOU YU, AND JUNJIE ZHU

Abstract. Training of object detection models requires a large number of images with identification of the regions of interest. Currently, the selection of regions of interest is done manually, which could be inefficient and inaccurate. In this project, we developed an automated recognition and labeling tool for road lines in aerial road and parking lot images, based on image processing techniques and unsupervised machine learning methods. The tool we created here contains three major steps, including color filtering, edge detection, and objects recognition and labeling. We demonstrate how this tool works by presenting an example where the yellow parking lot lines are successfully identified and labeled. We conclude this project with a summary of future work could be done to improve this tool.

1. Introduction

Identifying road lines and marks is an important task in monitoring traffic and parking lot occupancy. For instance, determining the real-time occupancy percentage of a parking lot. One of the challenges here is to identify objects in the images, such as yellow lines to delimit parking spaces. This process can be done manually using computing vision programs, although this is not time- efficient and robust.

Machine learning methods can help automate the recognition process. If objects were previously classified for some images, we could use supervised machine learning to classify the same objects in different images. In order to obtain a set of images with identified objects, such as yellow lines, we can use unsupervised machine learning in combination with image processing techniques. The difficult task is to distinguish yellow lines from other yellow marks (such as handicap marks and no parking yellow stripes) and yellow patterns in the ground or vehicles. The goal of this project is to identify yellow and white lines in images of parking lots and roads using image processing techniques and unsupervised machine learning.

2. Methods

In this section, we introduce the method we used for automated road lines recognition. The method contains 3 main steps, including color filtering, edge detection, and Hough line detection and shape classification with box labeling. Note that we only mention yellow line recognition here for demonstration purpose; however, the method could be applied to any color line recognition. We implement this method on Python with the help of various libraries, such as OpenCV [1], Scikit-Learn [7], and Pillow [3]. We will present some of the results in Section 3.

2.1. **Color Filtering.**

Since our goal is to recognize the yellow lines, it would be beneficial to identify all yellow pixels first. We select a standard RGB (Red, Green, Blue) value for the yellow color and look for all pixels in the images with RGB values close to that of the standard yellow. Here are some methods we have attempted.

2.1.1. *HSV Filter.*

One way to extract the yellow pixels is to convert the RGB values to HSV (hue, saturation, value), and we identify yellow pixels with predefined range of hue, saturation, and value/brightness.

Carlos Contreras: carlos.contreras@ualberta.ca
Keran Li: keran.li1@ucalgary.ca
Yi Sui: ysui@sfu.ca
Li Wang: lwang@math.ubc.ca
Tingzhou Yu: tingzhouyu@uvic.ca
Junjie Zhu: jzhu@math.ubc.ca

2.1.2. *Color Quantization with K-Means.* K-Means is an unsupervised machine learning algorithm that groups data into clusters. Given a hyper-parameter K, K-Means can assign rows of data to K clusters, such that each row is assigned to its closest cluster center in Euclidean space, and each cluster center is the mean of data assigned to it.

We applied the K-means algorithm from Scikit-Learn to RGB values of the pixels from all images. To extract the yellow pixels, we only keep the pixels that are in the same cluster as the standard yellow. Compared to the HSV filter, we did not need to specify the range of yellow, which allows the potential that pixels with noise can still be classified as yellow.

2.1.3. *Color Quantization with DBSCAN and HDBSCAN.* DBSCAN (Density-based spatial clustering of applications with noise) and HDBSCAN (Hierarchical density-based spatial clustering of applications with noise) are two other clustering algorithms. Given a hyper-parameter ϵ and *min_samples*, DBSCAN declares two pixels to be in the same cluster if their distance is less than ϵ, and we ignore clusters with less than *min_samples* data points. Compared to K-Means, not every data point needs to be in a cluster, and DBSCAN ensures that pixels with similar values are in the same cluster. HDBSCAN is an improvement of DBSCAN.

However, the run time of these two algorithms is much higher than K-Means. We adapted by resizing the images to 224×224. Also, other pixels are in the same cluster as the standard yellow. Especially with resizing, the RGB values of the pixels space out evenly in the color space, and there is not a large gap between the cluster of yellow pixels and other pixels. Thus, these two algorithms do not perform well. Setting $\epsilon = 5.3$, they excluded almost all pixels. If $\epsilon = 5.4$, they included most of the pixels, so they did not help with extracting the yellow pixels only.

2.1.4. *Color Quantization with Gaussian Mixture Models.* We also applied Gaussian Mixture models to study the color quantization on the images. The Gaussian Mixture model is a probabilistic unsupervised model, and models are learned by using the maximum likelihood estimates (MLE). Given a pixel K-means would classify it as one color only, where the mixture of Gaussian could say with certain probability it is yellow, and with certain probability chance it is green, allowing us the view the color clustering with uncertainty.

We implemented the Gaussian Mixture Models from Scikit-learn with co-variance types "tied" and "diag" and different number of components. Type "tied" showed better results for the image set we have. To identify the yellow pixels from the parking lines, we changed the color modes to RGB for images and follow the same procedures as K-means. We compared the results from K-means and Gaussian Mixture models, and see that Gaussian Mixture models can capture more relevant yellow parking lines pixels from the shadow see Figure 5, which is consistent with our expectations. However, Gaussian mixture was not able to solve the problem with shadow completely.

2.2. **Edge Detection.** Based on the color clustering information, we can now get a masked image with only yellow objects left and then we can move on to do edge detection. To obtain the best edge detection result, we first change the masked image to be grayscale and then apply a Gaussian filtering [8, Section 3.1] to remove noise in the masked image. Following this, Canny edge detection [2] is applied to find the edges for the yellow lines we are interested in. Note that to apply Gaussian filtering, we need to specify the size of the kernel to be used. Moreover, for Canny edge detection, we also need to specify two threshold values, `minVal` and `maxVal`, to classify edge lines and non-edge lines. The value of these parameters used in our experiments will be specified in Section 3. For more information on Canny edge detection along with Gaussian filtering and how to perform it on OpenCV in Python, see [9].

2.3. **Hough Line Detection, Bounding Boxes and Shape Classifier.** Upon obtaining the Canny edge detection result, we can identify objects by creating either line detection based on Hough transform [8, Section 7.4.2] or the bounding boxes based on contours.

Experiments presented in Section 3 apply the probabilistic Hough transform function (`HoughLinesP`) [6] in OpenCV for line detection, and then the detected lines are used as a mask over the original image. Note that, in order to apply the probabilistic Hough transform, we need to specify the values of threshold, the minimal line length, and the maximum line gap. The values used in our experiments will be specified in Section 3. For more information on how to perform Hough transform using OpenCV, see [9].

Bounding boxes are obtained using firstly the function `findCountours` to obtain contours delimiting the detected edges (given by delimiting point), and secondly the function `minAreaRect` to obtain the smallest

rectangle that encloses each contour (given by the corner point of the rectangle). This gives a number of bounding boxes enclosing each of the regions detected in the previous steps.

Note that those rectangles will include lines as well and other identified objects, which are not lines. To distinguish lines from non-lines objects, we classify them based on geometric properties of the bounding boxes using the function `boundingRect`. In particular, we compute the maximum length and aspect ratio (the longest side over the shortest side) of the bounding box since lines will be long and thin (large aspect ration and maximum side). We then select the shapes with aspect ratio largest than 4. Alternatively, we can use clustering methods, such as hierarchical clusters or K-means, to classify long and thin rectangles based on their geometrical properties.

3. Results

In Section 2, we introduced the method we used for automated line recognition. Now we are ready to present the results for some experiments we have done. All experiments presented in this section are performed on the resized 512 by 512 parking lot image shown as Figure 1. Unless specified, we worked on the standard RGB color space with the reference yellow color taken to be $(210, 200, 0)$. If the color quantization is performed, it is always done on all the data images we have. Moreover, a 7 by 7 kernel for the Gaussian filter is applied before Canny edge detection. The minimal and maximal value used in Canny edge detection are 150 and 280, and the threshold, the minimal line length, and maximum line gap for Hough line detection are taken to be 20, 10, and 10 respectively. Finally, it is worth pointing out that all experiments are performed on Python version 3.7.7 with OpenCV version 3.4.2.

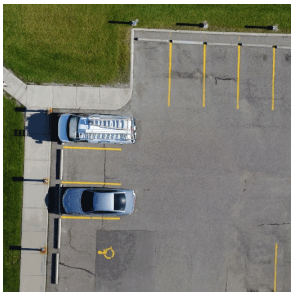
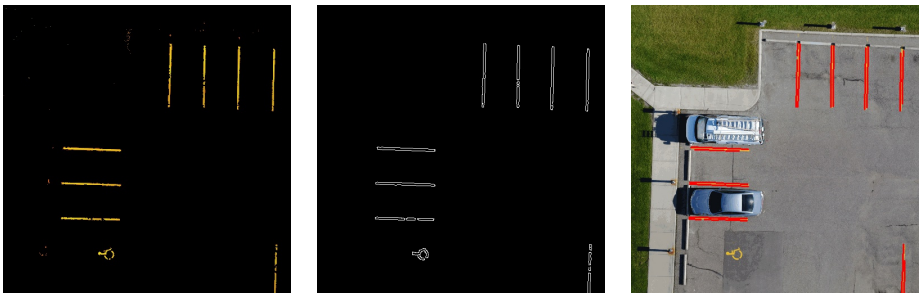


Figure 1. Resized 512 by 512 parking lot image.

In the first experiment, we apply a fixed range yellow color filter defined in the HSV space. Here, we take the lower bound of the yellow filter to be $(22, 60, 140)$ and the upper bound of the yellow filter to be $(60, 255, 255)$. Shown as Figure 2, with applying this color filter, we are able to keep only the yellow objects in the parking lot in the masked image. Following by Canny edge detection and Hough line transform, we can detect the yellow lines in the parking lot accurately (shown as panel (c) of Figure 2). However, a clear drawback for this approach is that we need to predefine a yellow filter in the HSV space, which can be hard to determine and have dependence according to the image. This motivates us to do the next trial, which applies color quantization to filter out the color other than yellow.

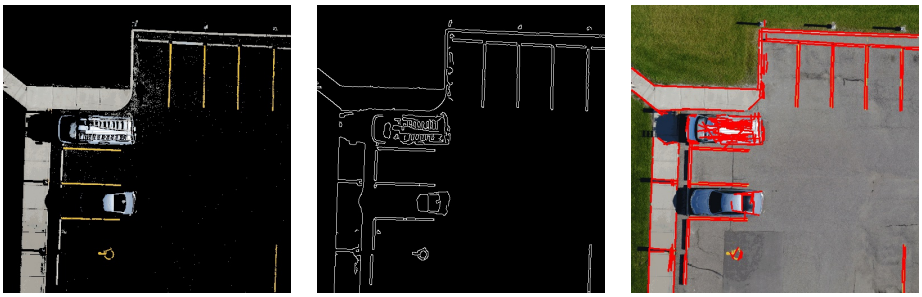
In this experiment, we apply color quantization with K-means clustering of group $K = 5$ in order to filter out the color other than yellow. The corresponding result is shown in Figure 3. In Figure (a), we see that, in addition to the yellow lines in the parking lot, the sidewalk, cars and some other non-yellow objects remain in the masked image. It leads to a noisy Canny edge detection result as shown in panel (b), so that we end up with a very inaccurate detection of the yellow lines of the parking spots. It suggests us that $K = 5$ may be too small for the yellow color filtering, and a bigger number of K should be applied in the clustering to see whether it can improve the result.

Finally, we present the result when either $K = 6$ for K-means or 5 components for Gaussian mixture model is applied for the color quantization in Figure 4. Here, for Gaussian mixture model, we apply the "tied" co-variance. Comparing the K-means with Gaussian mixture model, we see that, by applying either of the clustering method, yellow parking lines can be detected accurately. However, we have used one less cluster group for Gaussian mixture model compared to K-means in this example. After obtaining the edge



(a) (b) (c)

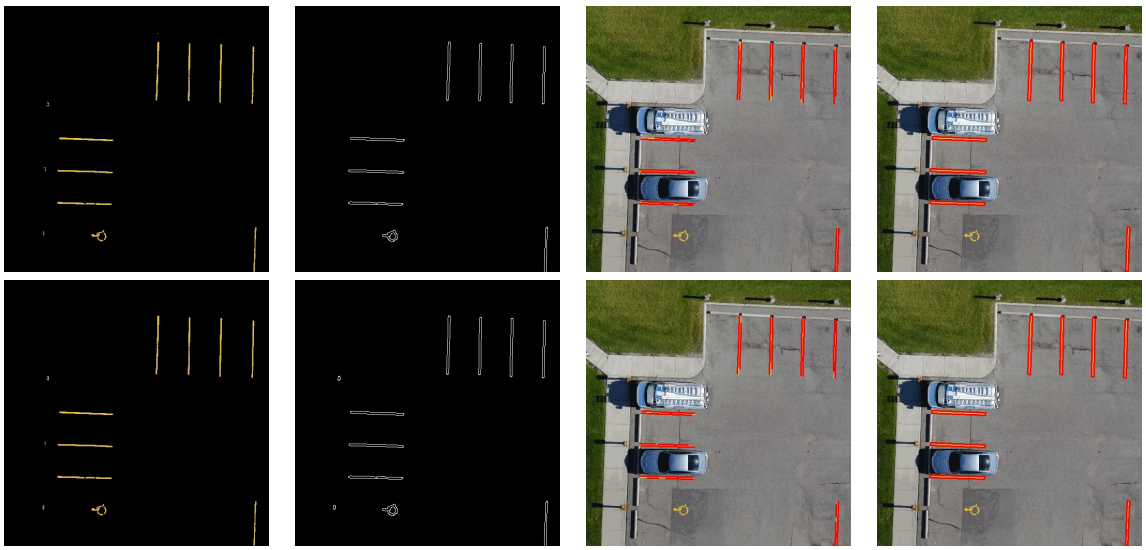
FIGURE 2. Result for a fixed range yellow filter in HSV. (a) Color filtering result, (b) Canny edge detection result, (c) Hough line detection result.



(a) (b) (c)

FIGURE 3. Result for K-means with $K = 5$. (a) Color filtering result, (b) Canny edge detection result, (c) Hough line detection result.

detected image, we can then also apply the shape classification method (described in Section 2.3) to label the yellow lines we are interested in with boxes.



(a) (b) (c) (d)

FIGURE 4. (a) color filtering results, (b) Canny edge detection results, (c) Hough line detection results, (d) box labelling results. Results for K-means are shown on the top and for Gaussian mixture on the bottom.

4. Conclusion and Future Work

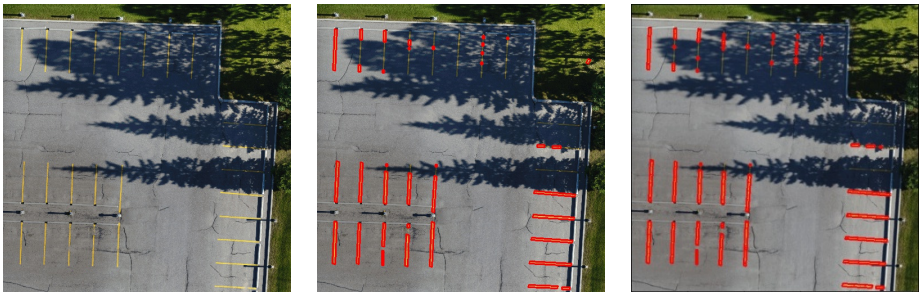
In this work, we use unsupervised machine learning methods and image processing techniques to identify yellow and white lines in aerial images of roads and parking lots. The general pipeline consists of the following sequential steps:

(1) Color quantization. Use unsupervised machine learning (K-means or Gaussian mixture models) to classify colors into clusters. The clustering can be applied to a number of images concatenated together, in which case a larger spectrum of colors is considered and the classifier can be saved for future use.
(2) Color filtering. Keep only yellow (white) pixels based on the color clusters. The output is a copy of the original image (or transformed color model) where all pixels not belonging to the cluster have been removed. This step assumes that all yellow (white) pixels of interest are very similar.
(3) Image smoothing. Apply a filter to eliminate isolated or faint pixels that remain after the color filter.
(4) Edge detection. Apply Canny edge detection to identify all enclosed regions that remain after color and smooth filters. The output is a binary image delimiting all filtered pixels.
(5) Contours. Identify the contour enclosing each of the group of pixels that has been edge detected. The output is a list of contour points delimiting each group of filtered pixels.
(6) Bounding boxes. Determine the minimal rectangle that encloses the contour. The output is a list of four (corner) points for each group of filtered pixels.
(7) Shape recognition. Use unsupervised machine learning (hierarchical clusters or K-means) or heuristic rules (aspect ratio larger than four) to classify and select those rectangles that are thin and long. The output is a subset of the bounding boxes.
(8) Labeling. Identify yellow (white) lines by plotting bounding boxes over the original image.

We believe the pipeline above can be applied outside the data set that we considered. However, more experiments would be required with images including curved or angled road lines, etc. From our experiments with the available data set, we identified a number of issues that we couldn't address due to time constrains. These issues and potential solutions/improvements are explained in the rest of this section.

4.1. **Future Work.**

4.1.1. *Shadow.* Shadow is the main obstacle in this project. For example, from Figure 5 (b) we can see the K-means clustering with Hough line detector cannot find all the yellow lines in the shadow area. This is because K-means can only pick up one yellow color while there is an obvious color difference between the normal yellow color in the bright area and dark yellow color in the shaded area. This can be partially solved by Gaussian Mixture model as shown in Figure 5 (c), in which dark yellow color in shaded area is recognized as the normal yellow color in the bright area with a certain probability. However, we can see some yellow lines in shaded area are still missing from line detection.



(a) (b) (c)

FIGURE 5. (a) resized original image, (b) K-means result, (c) Gaussian Mixture model result.

One possible way to address the shadow issue is to modify the part of the image inside the shadow area while keeping the part of the image outside of the shadow unchanged. We tried two different types of modifications, to increase the brightness of the shadow area by a certain value, or to modify the RGB color values of the shadow area and make them closer to the ones in the bright area. Both methods start with

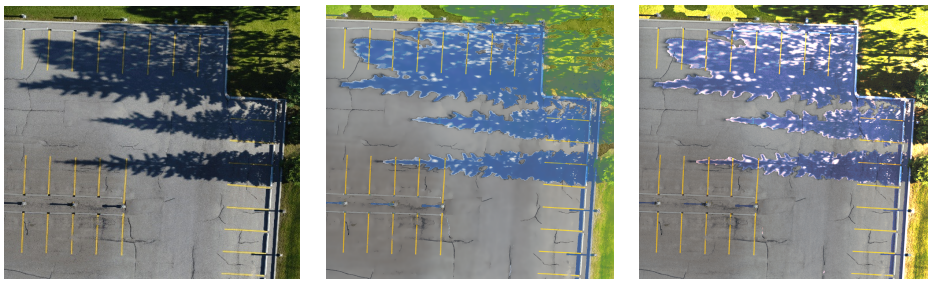
removing the texture of the image by a mean shift filter. Then threshold the image and the shadow area can be picked up. Now change the color space of the shadow area from RGB to HSV so we can increase the brightness of the shaded area. (Or we can multiply the RGB color values of the shadow area to make the shadow area and the bright area have the same means of color values). Finally, we combine the images of the modified shadow area with the normal bright area to obtain the output. We can also use the mean shift filter again or inpainting tool to make the gap between the shadow area and bright area smoother. Figure 6 shows the result of brightening and color modification (the white spots inside shadow in panel (c) come from small components removal after thresholding the original image, which is optional).

Another way to address with shadow parts is to apply the Homomorphic filter [4] to all images in the image process procedures, in order to improve the light exposure. This part is done before the color quantization. To be more specific, we first changed the color modes to HSV. After that, we applied the Homomorphic filter on the low energy frequency on the brightness V channel, because the shadow part is related to the low energy frequency compared to the non-shadow parts. Then, we shifted the mean value of the V channel of the isomorphic filter images back to the original images. We can see some results in Figure 7.

Together with other tools, one of the possible procedures to detect all lines in image with shadow is

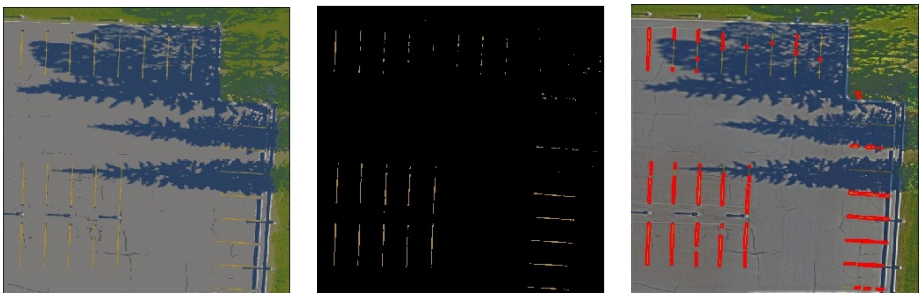
(1) Shadow Modification
(2) Color Quantization
(3) Line Detection

where we can change the order of the first two steps to compare the performance.



(a) (b) (c)

FIGURE 6. (a) original image, (b) brightened shadow, (c) color modified shadow.



(a) (b) (c)

FIGURE 7. (a) Homomorphic filtered image (b) Color filtered image (c) Line detection image with Homomorhpic filter

4.1.2. *Unwanted White Lines.* When detecting white lines on roads, usually curbs are also detected at the same time. This creates problems if we want to distinguish white dash lines or marks in the middle of roads from curbs. As an example, Figure 8 (b) shows curb affecting the detection of white dash lines if we only combine K-means clustering with Hough line detection. Figure 8 (c) shows if we apply one iteration of mean shift filter before K-means clustering, the influence of curbs becomes much smaller. It leaves us future work

to do with further investigation on how to properly apply mean shift filter to only detect the white dash lines in the middle of the road.

(a) (b) (c)

FIGURE 8. (a) original image, (b) direct K-means result, (c) mean shift filtering before K-means result.

4.1.3. *Discontinued Lines.* Some images have discontinued lines due to washed out paint. In this case, two or more objects are identified which may be filtered out in the shape recognition step. We think that such lines can be reconstructed using machine learning based on complete lines present in the same image.

4.1.4. *Shape Recognition.* Currently, the shape recognition uses a fixed number of two clusters to classify lines from non-lines. However, this leads to sub-classifications of lines when there are only lines in the image, and misclassification of non-lines as lines when there are too many non-line objects in the image. We suggest using an unsupervised machine learning method with a variable number of clusters.

4.1.5. *Automated Parameter Selection.* Recall that the goal for this project is to create an automated recognition tool for road lines. When the color filtering step could be done automatically through machine learning techniques, we should not forget the fact that there are many parameters in this method which need to be pre-selected, such as the number of groups in clustering, the minVal, and MaxVal threshold in Canny edge detection, and three parameters in Hough line detection. Moreover, in the newly proposed shadow modification method shown as Figure 6 (b), we also need to manually increase the HSV brightness value. Future work includes using machine learning to automate optimal parameter selection.

ACKNOWLEDGMENTS

We want to thank our industrial mentor Heather Vooys, Brian Bullas, Daniel McReynolds, the rest of the team at AERIUM Analytics, and our academic mentor Matthew Greenberg for their guidance, feedback, and support during the development of this project. We are also very grateful to Kristine Bauer, James Colliander, Ian Allison, and the rest of the PIMS organizing committee and instructors of the Math$^{\text{Industry}}$ 2020 workshop.

REFERENCES

1. G. Bradski, *The OpenCV Library*, Dr. Dobb's Journal of Software Tools, (2000).
2. John Canny, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-8** (1986), no. 6, 679–698.
3. Alex Clark, *Pillow (PIL Fork) Documentation*, 2015.
4. Glasgio, *Homomorphic filter*, `https://github.com/glasgio/homomorphic-filter`.
5. Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, Lawrence Zitnick C., and Poitr Dollár, *Microsoft COCO: Common Objects in Context*, 2014.
6. Jiri Matas, Charles Galambos, and Josef Kittler, *Robust detection of lines using the progressive probabilistic hough transform*, Computer Vision and Image Understanding, **78** (2000), no. 1, 119–137.
7. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.
8. Richard Szeliski, *Computer Vision: Algorithms and Applications, 2nd ed.*, 2020.
9. OpenCV Team, *Image processing in OpenCV*, `https://docs.opencv.org/3.4.2/d2/d96/tutorial_py_table_of_contents_imgproc.html`, 2018.

Math Industry Workshop

HOUSING PRICE PROJECT REPORT

DANIEL DI BENEDETTO, LEIMIN GAO, YIWEI HUANG, NEHA SHARMA AND DONGYING WANG

1. Introduction

The real estate sector is an important industry with many stakeholders ranging from regulatory bodies to private companies and investors. Among these stakeholders, there is a high demand for a better understanding of the industry operational mechanism and driving factors.

Today there is a large amount of data available on relevant statistics as well as on additional contextual factors, and it is natural to try to make use of these in order to improve our understanding of the industry. Notably, this has been done in Zillow's Zestimate [4] and Kaggle's competitions on housing prices [2].

In some cases, non-traditional variables have proved to be useful predictors of real estate trends. For example, in [3] it is observed that Seattle apartments close to specialty food stores such as Whole Foods experienced a higher increase in value than average.

This project can be considered as a further step towards more evidence-based decision making for the benefit of these stakeholders. The project focused on assessment value for residential properties in Calgary between 2017-2020 based on data from [1]. The aim of our project was to build a predictive model for change in house prices in the year 2021 based on certain time and geography dependent variables.

The main steps in our research were the following.

- **Exploratory Data Analysis** (**EDA**).
  By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.

- **Feature Selection**
  In order to avoid overfitting issues, we select 20(according to PCA [12]) variables out of the original 36 by using methods ANOVA [9], LASSO [14], elastic net [15], forward feature selection, backward feature selection.

- **Modeling**
  We apply Decision Tree [7], Random Forest [8] and Xgboost [6] models for prediction of the percentage change of the housing prices.

- **Exploration of reasons for misclassification in model**
  We then go back to the original data to find out why some samples are misclassified by our model.

In this report, we describe our approach to these steps and the results that we obtained.

2. Exploratory Data Analysis

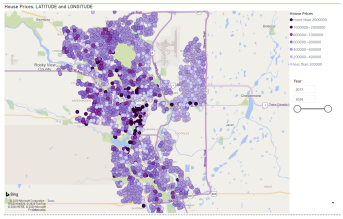


Figure 1. Housing prices for 2017-Calgary

In order to understand our data, we first perform exploratory data analysis. This will provide us with insights that will be useful in building prediction models, as well as insights that may be of interest to stakeholders. As part of the Exploratory Data Analysis we aim to:

- Look into the relationship between each variables and annual house price percentage change, and identify any patterns. For example, between the year of construction of a house and its annual percent price change.
- We will also analyse relationships between the features. This may reveal that certain features are redundant and this would help the subsequent analysis.

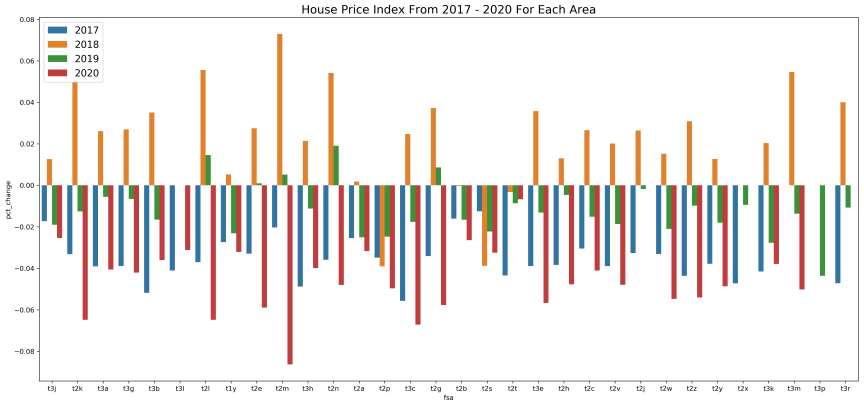


Figure 2. Housing prices index -Calgary

As part of the EDA, we first looked at the mean percent change of the housing prices from 2017-2020 for each FSA whose data is given. Figure 2 suggests that on

an average there was positive change in prices in the Year 2018.
In order to analyse our features more carefully, we also looked at the correlation of various features of the houses.

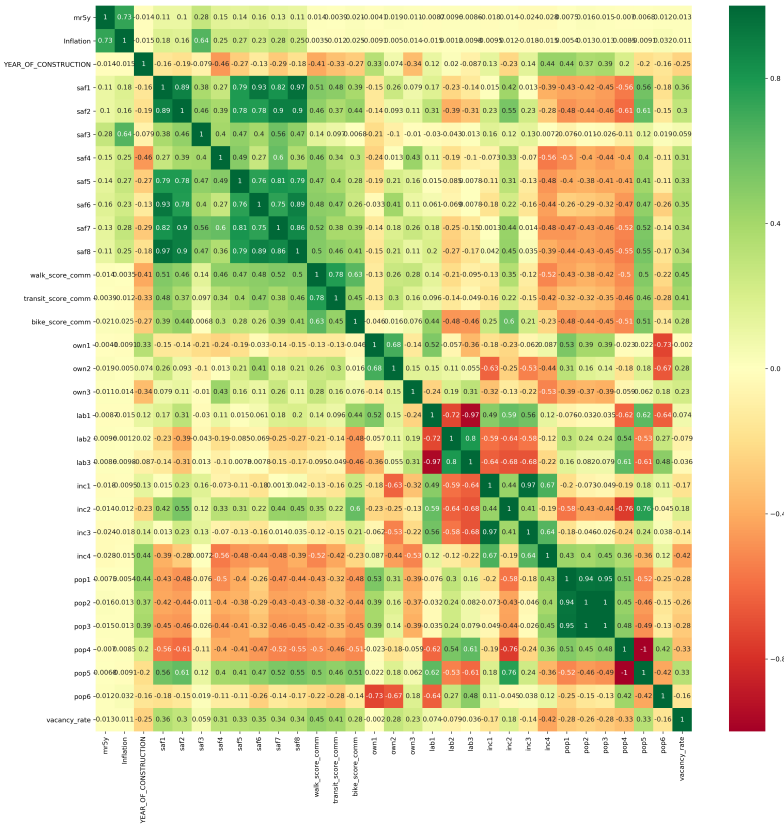


FIGURE 3. Correlation of Features

Figure 2 gives us an insight on how parameters are correlated with each other.

3. METHODOLOGY

3.1. **Feature selection.** Our data has 36 features in total. If we use all of them in our prediction model, the model will have a risk of overfitting. Therefore, we decide to remove some unimportant features. We choose a dimensionality reduction algorithm called Principal Component Analysis (PCA) as the method to estimate how many components are needed to describe the data. The optimal number of features for the prediction can be determined by looking at the cumulative explained variance ratio as a function of the number of components.

This curve quantifies how much of the total, 36-dimensional variance is contained within the first n components. For example, we see that with the digits the first 10 components contain approximately 90% of the variance, while you need around 25 components to describe close to 100% of the variance. Here we see that our

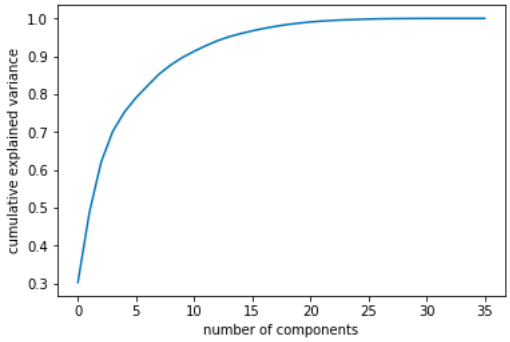


FIGURE 4. PCA Analysis

five-dimensional projection loses a lot of information (as measured by the explained variance) and that we would need about 20 components to retain 98% of the variance.

By using five different feature selection methods: ANOVA, LASSO, ELASTIC NET, FORWARD FEATURE SELECTION, BACKWORD FEATURE SELECTION, we were able to select 20 features out of the initial 36 features as a result of overlapping features that we observed in each feature selection algorithm. Those 20 features are: saf4; saf5; mr5y; Inflation; pop1; pop2; inc3; own3; lab1; walk score comm; Age; saf2; saf3; pop3; pop4; inc1; inc2; own2; lab2; vacancy rate.

3.2. **Percent Change price prediction.** The percent change price can be divided into four different groups: $[-0.12, -0.06)$, $[-0.06, 0)$, $[0, 0.06)$ and $[0.06, 0.12]$. In this section, we are going to consider our problem as a classification problem.

Based on the selected features, we applied three different Machine Learning algorithms: Decision Tree, Random Forest and XGBoost, on the training data and then used the testing data to check the accuracy, which equals to the number of samples that predicted in the right group divides the total sample size of our testing data. Here is the table of the accuracy rate:

Method	Accuracy Rate
Decision Tree	66.8%
Random Forest (with 1000 estimators)	68.1%
XGBoost	69.7%

Since XGBoost model is interpretable and performs best on the accuracy rate, we use it as our prediction model.

The above tree plot gives an example on how does an XGBoost model arrive at its final decision. This plot also shows the conditions on the node that splits the tree.

After getting an XGBoost model, we can examine the importance of each feature within the model by counting the number of times each feature is split on across

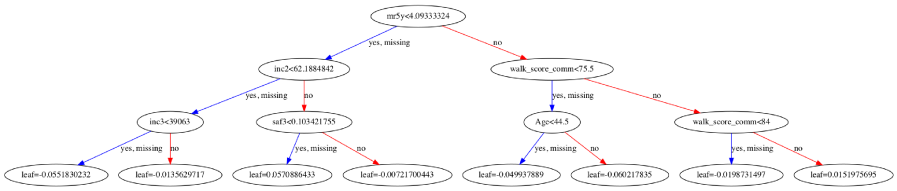


FIGURE 5. XGboost Decision Tree

all boosting trees in the model. The order of the importance of different features is plotted as a bar graph:

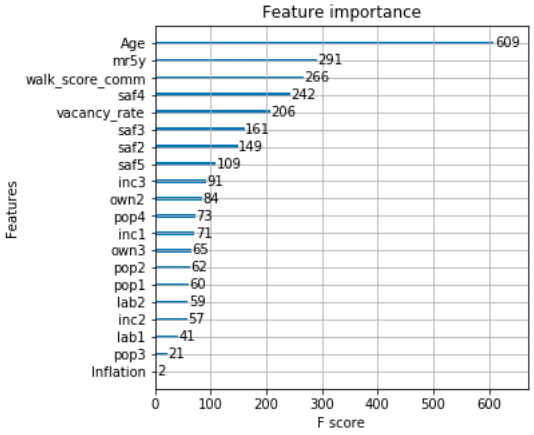


FIGURE 6. Feature Importance Bar Plot

From this plot, we can see that *Age* has the highest importance and *Inflation* has the lowest importance.

Next, we use the confusion matrix to help us visualize the performance of this XGBoost model:

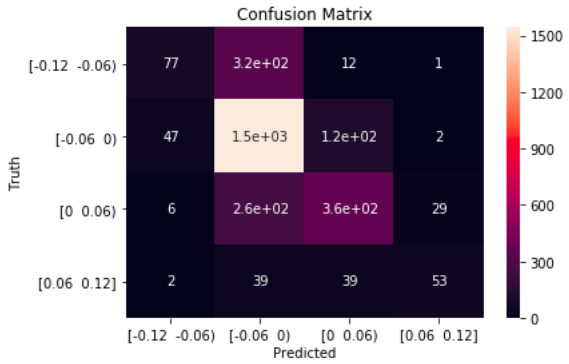


FIGURE 7. Confusion Matrix Results

In the above matrix, each row represents the instances in an actual group while each column represents the instances in a predicted group. It is every easy to see how many samples are mislabeled by this model. Take group $[0.06, 0.12]$ as an example, the actual size of that group is 133, 53 of them are predicted correctly in the group $[0.06, 0.12]$ while 80 of them are mislabeled in the wrong groups: 39 cases are mislabeled in group $[0, 0.06)$; 39 cases are mislabeled in group $[-0.06, 0)$ and 2 cases are mislabeled in group $[-0.12, -0.06)$. The above matrix shows that most cases in group $[-0.06, 0)$ can be predicted correctly by this XGBoost model, but the mislabeling rate for other three groups is not low, especially for group $[-0.12, -0.06)$. Therefore, our next step is to find the main reasons for those mislabeling cases.

4. Exploration of reasons for misclassification in model

We focus on finding the reasons why some houses that are supposed to appear in group $[-0.12, -0.06)$ are in group $[-0.06, 0)$ and houses supposed to appear in $[0, 0.06)$ appear in $[-0.06, 0)$. Thus, we could check datum of significant factors and find out why misclassification occur. Why this is important—we want to find out why the predicted percentage change of price for some houses is exceptionally low or high, which is important to basically every stakeholder. The following is a screen shot of our dashboard and the dots on the map are some selected points from our table.

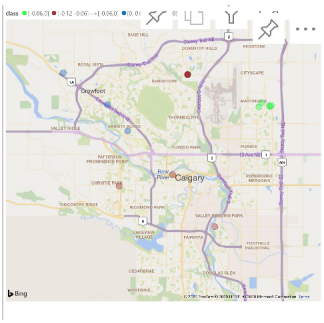


FIGURE 8. Some houses with exceptionally high/low percent change

For further information of other properties, please refer to the links in footnotes[1][2] After that, we count the frequency of each significant features that appears, and we get the following graph

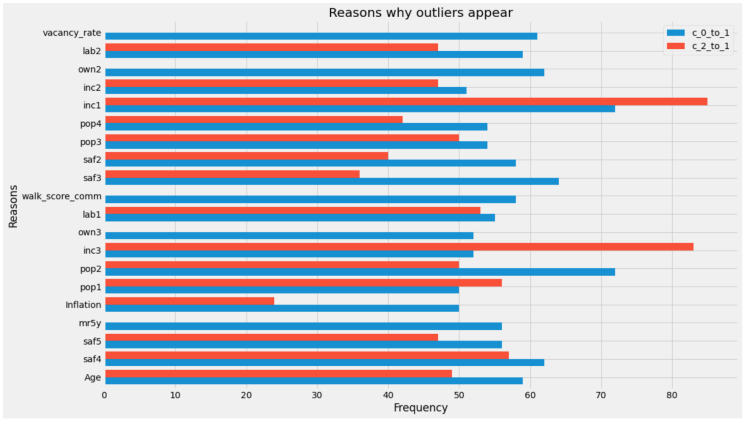


FIGURE 9. Frequency of Significant Features in Outliers

From figure 9,we could see that inc1 and pop1 are the most important factors making some houses that are supposed to appear in group $[-0.12, -0.06)$ are in group $[-0.06, 0)$. Also, in1 and inc3 are the most important factors making houses supposed to appear in $[0, 0.06)$ appear in $[-0.06, 0)$.

5. COMMUNICATING OUR RESULTS

Given that our project was motivated by practical interest to stakeholders, we aim to publicly deploy a dashboard presenting our main results. This part of the project is still work in progress; we are currently working on a prototype in Power

[1] https://github.com/yiwei14/BCFA-yiwei-/blob/master/misclassication0_1.csv
[2] https://github.com/yiwei14/BCFA-yiwei-/blob/master/misclassication2_1.csv

BI (see Figure 10), and we will then build the dashboard using Plotly Dash since this allows for easy public deployment.

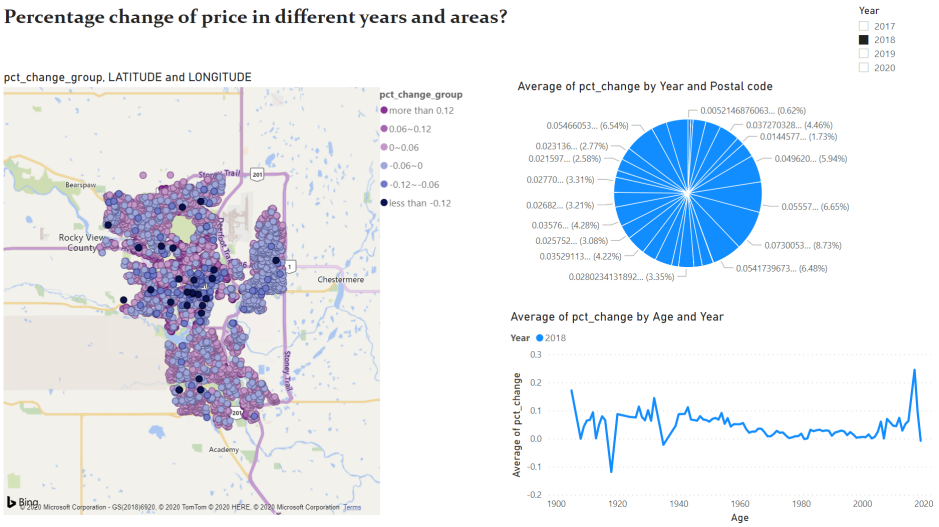


FIGURE 10. Dashboard prototype

Much of our data can be naturally viewed on a map, and by doing this many properties of the data can be seen easily. Specifically we place the set of data points on a map of Calgary using the coordinates of each house. By colouring these points according to the percent change in price of the corresponding house, the user can then visually identify geographical patterns.

The dashboard[3] will be interactive, giving the user the ability to view our results from various perspectives. For example, the user will be able to select a geography-dependent variable of interest to them, and to view the map colour-coded according to this feature with the data points overlaid. Different stakeholders may be interested in different variables, and these interactive features allow each user to choose how they visualise the data.

In a separate section of the dashboard, we will have our predictive model. In this section, the user test our model on new data points and the model will output a prediction for the percent price change of that house in the year 2021. This allows the user to explore how house prices would react in various potential scenarios, and we believe this could be helpful for future decision making.

6. SUMMARY

By analysing historical data for house prices in Calgary along with various relevant features, we established some interesting patterns and trends. Using machine

[3] https://youtu.be/DXC6p8ImGns

learning techniques, we were then able to identify a subset of the original features that are in a sense sufficient to describe our data.

Having selected the most important features, we then trained an XGBoost model for change in house price prediction, which classified samples into one of four categories. This model gave an accuracy rate of 68.7 on a test set that we had kept separate during development. This model can therefore be used to predict, for example, which type of house within Calgary is likely to increase and decrease in price in the year 2021 based on various scenarios.

Acknowledgements

We would like to express our deep and sincere gratitude to PIMS for giving us the opportunity to do this project. As a great bridge between academic and industry, this program educated us how to perform theoretical methodology in real life.

We would like to express our sincere thankfulness to Dr. Firas Moosvi and Dave Dong for the continuous support of our research, for their patience, enthusiasm, motivation and immense knowledge. As our academic mentor, Dr. Firas Moosvi gave a lot active feedback on every step of the research. As industrial mentor, Dave Dong was dedicated to instructing us to get a realistic meaningful model. Both of them kept reminding us of the importance of collaboration, which ensured that the whole project proceeded smoothly.

Additionally, we would also like to thank all our friends who offered us some help, such as Stephen Styles who helped us build the misclassification table.

References

1. *data source*, `https://raw.githubusercontent.com/shughestr/PIMS_2020_Real_Estate_data/master/sample_clean.csv`.
2. *Kaggle competition*, `https://www.kaggle.com/c/house-prices-advanced-regression-techniques`.
3. *Mckinsey report*, `https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/getting-ahead-of-the-market-how-big-data-is-transforming-real-estate`.
4. *Zillow zestimate*, `https://www.zillow.com/blog/zestimate-updates-230614/`.
5. Kam C Chan, Patric H Hendershott, and Anthony B Sanders, *Risk and return on real estate: evidence from equity reits*, Real Estate Economics **18** (1990), no. 4, 431–452.
6. T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system*, In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (2016, August), 785–794.
7. B. de Ville, *Decision trees*, Wiley Interdisciplinary Reviews: Computational Statistics **5** (2013), no. 6, 448–455.
8. T. G. Dietterich, *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization*, Machine learning **40** (2000), no. 2, 139–157.
9. R. A. Fisher, *Statistical methods for research workers*, In Breakthroughs in statistics (1992), 66–70.
10. Franz Fuerst and George Matysiak, *Analysing the performance of nonlisted real estate funds: a panel data analysis*, Applied Economics **45** (2013), no. 14, 1777–1788.
11. Richard J Herring and Susan M Wachter, *Real estate booms and banking busts: An international perspective*, The Wharton School Research Paper (1999), no. 99-27.

12. H. Hotelling, *Analysis of a complex of statistical variables into principal components*, Journal of educational psychology **24** (1933), no. 6, 417.
13. Calvin Schnure and Alexandra Thompson, *Commercial real estate and migration: What can the employment composition of local job markets tell us about future demand?*, Available at SSRN 3544939 (2020).
14. R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society: Series B (Methodological) **58** (1996), no. 1, 267–288.
15. H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, Journal of the royal statistical society: series B (statistical methodology) **67** (2005), no. 2, 301–320.

Appendix

Selected Varialbes	Explanation
Age	Construction Year
inf1	inflation
inc1	Median total income in 2015 among recipients ($)
inc2	Number of employment income recipients aged 15 years and over in private households
inc3	Median employment income in 2015 among recipients ($)
lab1	Labor Participation rate
lab2	Unemployment rate
mr5y	mortgage rate 5 year
own2	Total - Owner households in non-farm, non-reserve private dwellings, % of owner households spending 30% or more of its income on shelter costs
own3	Total - Tenant households in non-farm, non-reserve private dwellings, % of tenant households in subsidized housing
pop1	Population, 2016
pop2	Total private dwellings
pop3	Private dwellings by residents
pop4	Total - Distribution
saf2	Break and Enter Commercial
saf3	Break and Enter - Dwelling
saf4	Break and Enter - Other Premises
saf5	Commercial Robbery
vacancy_rate	Community vacancy rate
walk_score_comm	Community walk score
transit_score_comm	Community transit score

COMPRESSING THE TRANSACTION DATA OF BLOCKCHAIN

IVAN LAU, SHANG LI, EVAN MACNEIL, ALEXANDRA MCSWEEN,
ABHISHEK KUMAR SHUKLA, AND YANHONG XU

Abstract. Since its inception in 2009, the Bitcoin blockchain size has grown in size to more than 295 GB and continues to grow by approximately 50 GB per year. The decentralized, trustless framework of Blockchain requires participating nodes to store the entire blockchain data. This keeps smaller computing devices from participating fully in the network.

In this paper, we present methods to compress the blockchain in a lossless fashion. Our main observations rely on finding redundancies in the Bitcoin transaction data which can be leveraged to decrease the blockchain size. We are able to achieve a compression rate of approximately 20% by applying various compression schemes in concert. Further compression might be possible by using generic compression algorithms on top of our compression scheme.

1. Introduction

Bitcoin, the most famous cryptocurrency, was first introduced in Satoshi Nakamoto's white paper in 2008 [Nak08]. Philosophically, the idea is to have a peer-to-peer network of electronic cash without a centralized financial institution. In this decentralized system, we can operate (mostly) anonymously but we cannot trust anybody.

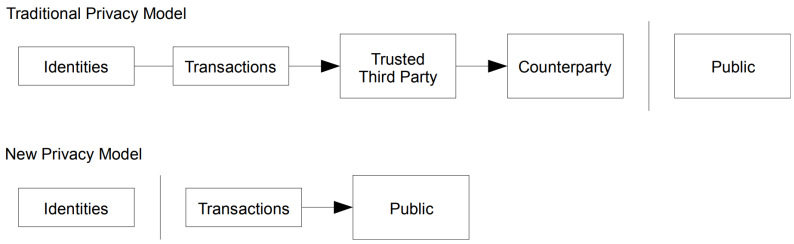


Figure 1. From [Nak08]

However, participating fully in the Bitcoin network remains inaccessible to most people. To verify the history of transactions, a node needs almost 300 gigabytes (GB) to store the blockchain [Blo]. To help "mine" the next block in the blockchain, thereby confirming new transactions, substantial computing power is needed. Many "wallet" apps allow users to trade in Bitcoin on the network without requiring large amounts of storage space and CPU power, but they require the user to trust in third parties, violating Bitcoin's trustless model.

The Divi Project [Div] is a blockchain-based cryptocurrency and smart wallet that seeks to make the cryptocurrency market accessible to all. It offers the first and only genuinely one-click masternode deployment and five tiers of affordability.

The motivation behind our project is help improve this inaccessibility. The original problem was posed by Germàn Luna from the Divi Project. We hoped to answer the following questions:

- Determine to what extent a transaction graph can be compressed (for later decompression) or what obstructions exist to its compression.
- What compression ratio can we achieve for an ordered sequence of cryptographic hashes?

These questions were intentionally left vague to allow us the freedom to explore and develop the project ourselves. We explored many options but in the end decided to look at techniques and places to compress transaction data since they carry the bulk of the storage strain.

For the purposes of our project, we worked on Bitcoin transactions, though we expect our methods to be applicable to other cryptocurrencies, including the Divi Project, as well.

1.1. **Anatomy of the Bitcoin Blockchain.** We can think of the blockchain as the ledger. It consists of a linked list of blocks, chronologically ordered, and each block contains a list of transactions. Besides transaction data, a block also contains some metadata regarding the block itself. In detail, the components of the block are as follows.

- The *size* of the block in bytes.
- The *block header*, which consists of:
  - The *version number* of the block.
  - The *hash* of the header of the previous block in the chain.
  - The *Merkle root* of the transactions in this block. This is a combined hash of all of the transactions.
  - A *timestamp* indicating approximately when the block was mined.
  - The *target* (also called "bits"). The smaller the target, the more difficult the block is to mine
  - A *nonce*. A value placed in the header so that the header's hash is smaller than the target.
- The *transaction count*, i.e. the number of transactions in the block.

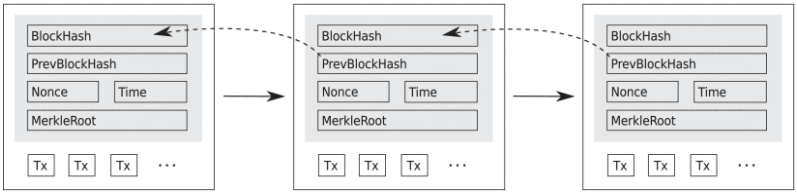


FIGURE 2. From [TS16]

- A list of *transactions*.

To mine a new block, a miner collects a number of transactions into a list and computes their Merkle root [Mer80]. The Bitcoin protocol dictates what the target value should be. The miner then iterates over several nonce values until one is found that causes the block header's hash to be lesser than the target value. It takes the entire network of miners working together about 10 minutes to mine a block; if miners work too fast or too slow, the Bitcoin protocol adjusts the target to keep this time around 10 minutes.

Note that a block header depends on the hashes of the blocks before it, which in turn depend on the transactions in those blocks. Therefore editing a previous transaction requires recomputing all subsequent block headers. Meanwhile, the mining network will be adding new blocks. Someone maliciously editing transactions will not be able to catch up unless they have more CPU power than the rest of the network combined.

1.2. **Anatomy of a Transaction.** Each block consists of one or more transactions. The first transaction in a block is the coinbase transaction, which rewards miners with BTC when adding a new block to the chain. Subsequent transactions reflect people sending money to one another. A transaction contains the following data in this order:

- The *version number*, either 1 or 2. Version 2 reinterprets the sequence numbers, below.
- A *flag* that, if present, is always equal to 1 and indicates that input script signatures are to be found in the witness section below[1].
- The *input count*, or number of inputs in the transaction.
- Several inputs, the number of which must match the count above. Each input in turn contains the following data:
  - The *previous transaction hash*, the hash of the transaction whose output this input will spend.
  - The *index* of the output within that transaction, since a transaction may have multiple outputs.
  - The *length* of the script signature to follow.
  - The *script signature*, a script that, when combined the output's script public key, must evaluate to true in order to spend the output.
  - The *sequence number*, related to when a transaction becomes final.
- The *output count*, the number of outputs in the transaction.
- Several *outputs*, each of which contain the following data:
  - A *value*, the amount of bitcoin to be transferred.
  - The *length* of the script public key to follow.
  - The *script public key*, a script that accepts an input and evaluate to true or false. Only the intended recipient should be able to produce the input that causes it to evaluate to true.

[1] "Segregated witness" (or "SegWit") is an upgrade to Bitcoin to address transaction malleability and block size limits. See [LLW]

- A list of *witnesses* (if the flag above was present) equal in number to the number of inputs. For each witness we have the number of items to push onto the stack (when evaluating scripts) followed by those items preceded by their lengths.
- A *lock time*, related to when a transaction becomes final.

1.3. **A Raw Transaction.** A transaction appears on the Bitcoin network as a serialized sequence of bytes. Here, is an example of one such serialized transaction, written in hexadecimal:

```
020000000001020df23cb58292fa49a1c14083e74b8f79d5dee5e32f75a96798438e
3be32ab68b0100000017160014b3026ad925e5c05d901493a733edfac535413f97fe
ffffff11e522fdf84b31800d1504d88b0bcd2fa36dcf83d3304222cc1ada77bd9b9d
1b000000000171600141ab4829400dd414ef49069b984d542847fb06f65feffffff02
808417000000000017a914d084a31c88c447cf6600b3cef10c63514bc0a91c878020
13000000000017a9148becb7c6ba1a3cde90cfa91dd28d3143c0c412428702473044
022054b3e206d43deb741f2e581d312bec9739d55c1fe8340a53be631a8fc818a269
022060fdec6eb9ef9bd9f0aadd0175018853670ad5f3cc7b33dd42668d31c82b9fbd
01210285687bd88db3039d5878e120a28b4f62e4726a2c08638d9181fa9233a20acb
4a0247304402203cea8cf98b019bc55c6d92c571b86c9a2bf9430cd340a487a7ffea
75e14b3ff602204c16854d229b883ee1009d961727d03479e917e675efa4b6331583
d2b3e66b7c01210268aded79b39ba3fd5dae2c335cfcaa676c73c7679ab1fd424fe6
748a079fe427c2dd0900
```

A transaction in this format is called a raw transaction. Reading left-to-right, all of the data from § 1.2 appear in order. If one knows the size in bytes of each piece of data, one may extract the meaning from the raw bytes. For instance, the first 4 bytes always correspond to the transaction version number. If the next byte is `00`, then it is followed by `01`, indicating the flag `0001` is present. Otherwise, if the next byte is non-zero, it belongs to the input count (which is necessarily non-zero). If the flag is present, it is then followed by the input count. The raw transaction above is parsed in full in Table 1 on next page.

Table 1: A parsed transaction

Version			02000000
Flag			0001
Input count			02
Input 0		Prev. hash	0df23cb58292fa49a1c14083e74b8f79d5dee5e32f75a96798438e3be32ab68b
		Prev. index	01000000
		Script length	17
		Script sig.	160014b3026ad925e5c05d901493a733edfac535413f97
		Sequence no.	fefffff
Input 1		Prev. hash	11e522fdf84b31800d1504d88b0bcd2fa36dcf83d3304222cc1ada77bd9b9d1b
		Prev. index	00000000
		Script length	17
		Script sig.	1600141ab4829400dd414ef49069b984d542847fb06f65
		Sequence no.	fefffff
Output count			02
Output 0		Value	8084170000000000
		Script length	17
		Script pub. key	a914d084a31c88c447cf6600b3cef10c63514bc0a91c87
Output 1		Value	8020130000000000
		Script length	17
		Script pub. key	a9148becb7c6ba1a3cde90cfa91dd28d3143c0c4124287
Witness 0	Item count		02
	Item 0	Length	47
		Data	3044022054b3e206d43deb741f2e581d312bec9739d55c1fe8340a53be631a8fc818a269022060fdec6eb9ef9bd9f0aadd0175018853670ad5f3cc7b33dd42668d31c82b9fbd01
	Item 1	Length	21
		Data	0285687bd88db3039d5878e120a28b4f62e4726a2c08638d9181fa9233a20acb4a
Witness 1	Item count		02
	Item 0	Length	47
		Data	304402203cea8cf98b019bc55c6d92c571b86c9a2bf9430cd340a487a7ffea75e14b3ff602204c16854d229b883ee1009d961727d03479e917e675efa4b6331583d2b3e66b7c01
	Item 1	Length	21
		Data	0268aded79b39ba3fd5dae2c335cfcaa676c73c7679ab1fd424fe6748a079fe427

Lock Time	c2dd0900

We note that many of the integer values in Table 1 (namely the version number, sequence numbers, indices, values, and lock time) are stored in little-endian format, meaning the least significant byte appear first. As such, 8084170000000000 would be read as the hexadecimal number 0x178480, or the decimal number 1541248, indicating a transfer of 0.01541248 BTC.

2. Compression

2.1. **Addresses.** Addresses in Bitcoin take the form of ECDSA (Elliptic Curve Digital Signing Algorithm) private key-public key pairs, on the elliptic curve Secp256k1. A private key is a randomly chosen 256-bit integer. The associated public key is obtained by multiplying the curve's standardized base point by the private key. Thus the public key consists of the x and y coordinates, each 32 bytes, of a point on the elliptic curve and can be used as a public address to which Bitcoins may be sent. However, advancements in quantum computing may lead to private keys being recoverable from public addresses using Shor's Algorithm. Bitcoin users now use 20-byte hashes of public keys, rather than the public keys themselves, as addresses.

Addresses appear in transaction scripts. For maximum privacy, a new address should be used for each transaction. Bitcoin software can generate and manage a multitude of addresses for the user. However, through our analysis we have found that some addresses have been reused in more than 50,000 transactions. Moreover, fewer than 2^{32} addresses appear in the Bitcoin blockchain, so that we can store addresses in a table and refer to them instead by a 4-byte index into this table. This saves us space as long as addresses are reused sufficiently often. Indeed, we find this to be the case.

We estimated the number of times an address is reused in the blockchain based on a sample of addresses. We queried https://sochain.com for the number of transactions involving a list of 74,591 addresses. Table 2 shows part of the data. The addresses are sorted lexicographically. We can see the first address was used 55,417 times in the sample, while the next two addresses were used 313 and 136 times, respectively.

Address	Frequency
1111111111111111111114oLvT2	55,417
11112BvbV6fY4Y5rghDB1vnJtLGSjoB2n	313
1111VHuXEzHaRCgXbVwojtaP7Co3QABb	136
1111vP5eq5RCmRBeMwJGFW65owVtb3nM	67
11121FrRst9KCVrdM8SqLRzAFw3b1woSno	1
...	...
12pPUDiYU7JWejK6gT2Rr9NxS5QsGcF78f	6
12pPup7Pe1XGhpCLWHeGm9D9KBjLG4mvQv	1

Table 2. Address reuse frequency

Table 3 groups addresses by the number of times there were reused. We can see that there were 53,967 addresses that were used exactly once each; 2,277 addresses that were used exactly twice each; 1,117 addresses used exactly three times teach; and so on. We found that the most frequently appearing address was reused 59,887 times in the sample. Furthermore, 72% of addresses were used only once while 28% of addresses were reused.

# of uses	# of addresses	% of sample
1	53,967	72.0021%
2	2,277	3.0379%
3	1,117	1.4903%
4	892	1.1901%
5	839	1.1194%
...	...	...
20,931	1	0.0013%
26,484	1	0.0013%
32,269	1	0.0013%
55,417	1	0.0013%
59,887	1	0.0013%

TABLE 3. Addresses by reuse frequency

If this sample is representative of the whole blockchain, then given that there are approximately 1.5×10^9 outputs [Bit], we can estimate the number of addresses as follows. Let x be the number of addresses in the Bitcoin blockchain. Then out of all the outputs $0.720021x$ are the number of addresses appearing exactly once, $0.030379x$ are the number of addresses appearing exactly twice, and so on. We solve for x in

$$x\,(1 \times 0.720021 + 2 \times 0.030379 + 3 \times 0.014903 + \ldots + 59887 \times 0.000013) = 1.5{\times}10^9$$

and find that there are approximately $x = 42,943,936$ distinct addresses appearing in the blockchain. This is much fewer than 2^{32} addresses, so we can store these addresses in a table and refer to them by a 4-byte index instead. Implementing such a table would cost $4x$ bytes, about 819 MB. However, replacing 20-byte addresses by 4-byte indices saves us 16 bytes per output, approximately 22.4 GB. The net savings are about 21.6 GB.

2.2. **Transaction Hashes.** Each transaction input refers to an unused output of an earlier transaction through the corresponding 32 bytes transaction hash (TXID). Therefore, transaction hashes will appear multiple times in the blockchain. Using data from [Bit] and [Tot], we find that there are approximately 1.4×10^9 inputs (or spent outputs), but only about 6.0×10^8 unique TXIDs. Therefore the average TXID is used 2.3 times.

Similar to § 2.1, we can store TXIDs in a table and refer to them instead by a 4-byte index. Implementing this table would cost 32 bytes per unique TXID, about

17.9 GB. However, we save 28 bytes per input, about 36.5 GB in total. The net savings are about 18.6 GB.

It is also worth mentioning that transaction hashes have a kind of "recursive structure" that can be exploited for further compression later on. Namely, since hashing is a deterministic operation and transactions depend on previous transactions' hashes, a different kind of lookup table might be possible where some hashes are stored and others are derived at decompression time by hashing decompressed data.

2.3. **Scripts.** A script is a list of instructions (also called script words, opcodes, or commands). It is a stack-based language processed from left to right. Every transaction input and output contains a script. In order for an input to spend an output, the input and output scripts are combined into one script, then evaluated. The spending is permitted if the script executes without error and leaves the value "true" on the stack.

Most scripts follow one of a few standardized forms, the most common of which is Pay to Public Key Hash (P2PKH). In P2PKH, the input script pushes a digital signature and public key onto the stack while the output script hashes the public key, compares it to an expected values, then verifies the signature.

In a script, data to be pushed to the stack is generally enclosed in angle brackets `<>` and data push commands are omitted while non-bracketed words are opcodes. For the sake of clarity, we include length of the script (`lenScript`) as well as push commands (`PUSHBYTES`) here. We now list five of the most common standardized scripts appearing in Bitcoin transactions, as well as how they appear in encoded in a raw transaction. Besides these five types, there are also multi-signature scripts (MULTISIG), unspendable outputs (OP_RETURN), and non-standard scripts.

PUBKEY	`lenScript PUSHBYTES[65] <pubKey> OP_CHECKSIG`
	`4341<pubKey>ac`
P2PKH	`lenScript OP_DUP OP_HASH160 PUSHBYTES[20] <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`
	`1976a914<pubKeyHash>88ac`
P2SH	`lenScript OP_HASH160 PUSHBYTES[20] <scriptHash> OP_EQUAL`
	`17a914<scriptHash>87`
P2WPKH	`lenScript OP_0 PUSHBYTES[20] <pubKeyHash>`
	`160014<pubKeyHash>`
P2WSH	`lenScript OP_0 PUSHBYTES[32] <witScriptHash>`
	`220020<witScriptHash>`

TABLE 4. Standard scripts

From Table 4, we observe that these five standard scripts have fixed formats except where the public keys may vary. For instance, a P2PKH script is always 26 bytes, starts with the bytes `1976a9a4`, and ends with the bytes `88ac`. One can replace this with a single byte that indicating it is a P2PKH script, followed by

Script type	Compressed script	Bytes saved
PUBKEY	`00<pubKey>`	2
P2PKH	`01<pubKeyHash>`	5
P2SH	`02<scriptHash>`	3
P2WPKH	`03<pubKeyHash>`	2
P2WSH	`04<scriptHash>`	2
Other	`05<script>`	-1

TABLE 5. Compressed scripts

the 20-byte hash, which may vary from script to script. We can do likewise for the other standard script types. Table 5 summarizes how we may compress these scripts. There is a drawback in that we would be required to reserve a byte to indicate if a script is not of one of these five handled types, followed by the uncompressed script. This costs us one byte per unhandled script, but this cost is offset by the fact that these unhandled types are far outnumbered by the others.

The savings in Table 5 are per output with a script of the given type. Using data from [Bit], we compute the expected total savings by multiplying the per-output savings by the number of outputs with the given script type. For instance, at the time of writing, there are 1,044,567,464 P2PKH scripts, so that we can save 4.86 GB compressing P2PKH scripts alone. There are 46,788,036 scripts not of the above five types, costing us about 44.6 MB with our compression scheme. Putting it all together, we expect a net savings of 5.91 GB.

2.4. **Version Number, Flag, Lock Time, and Sequence Numbers.** Each transaction has a version number, stored in 4 bytes. However, there are presently only two transaction versions: version 1 and version 2. The transaction version can therefore be represented by a single bit until such a time as a new version is created. Compressed data must be written in whole bytes at a time to file, but we observe that there is other information in a transaction that can also be represented by a single bit — namely the flag and information pertaining to the lock time and sequence numbers — and we collect these all into one byte together.

A transaction sometimes includes a flag that, if present, takes 2 bytes and is always equal to `0x0001`. This flag is used to indicate whether the transaction includes any witness data (used in scripts such as P2WPKH and P2WSH). As this can be represented by one bit, we store this in the same byte as the transaction version number.

Each transaction has a 4-byte "lock time" and each input has a 4-byte "sequence number". The lock time and sequence number relate to when the transaction becomes final. The lock time is usually assigned the minimum value `0x00000000` while the sequence numbers are usually the maximum value `0xffffffff`. We use one bit to represent whether the lock time is zero or not and store this bit with the version number and flag. We write the lock time to the compressed file only when it is non-zero. Similarly, we use one bit to represent whether all sequence numbers are maximal or if one sequence number is non-maximal. If they are all maximal, then we

do not write them to the compressed file. Moreover, if the sequence number is not maximal, then we also observe that it is usually some number close to `0xffffffff`. Indeed, we notice that the sequence number is usually between `0xffffff00` and `0xffffffff`. These we can represent with a single byte by storing instead their distance from `0xffffffff`.

In total, we save

- 3 bytes per transaction by compressing the version number down to 1 byte.
- 2 bytes per transaction with the flag present.
- 4 bytes per transaction with a lock time of `0x00000000`.
- 4 bytes per input in a transaction in which all sequence numbers are `0xffffffff`.
- 3 bytes per input in a transaction in which all sequence numbers are between `0xffffff00` and `0xffffffff`, inclusive, but where some sequence numbers is strictly below `0xffffffff`.

Based on a sample of $261,003$ transactions between August 17 and 18, 2020, we find that

- 20% of transactions had the flag present.
- 76% of transaction had a lock time of `0x00000000`.
- 68% of inputs were in transactions where all sequence numbers are `0xffffffff`.
- 31% of inputs were in transactions with a sequence number below `0xffffffff`, but where all sequence numbers are no less than `0xffffff00`.

If this sample is representative of the transactions in the blockchain, then given that there are now more than 560 million transactions and 1,380 million inputs (or spent outputs) [Bit], we can expect to save at least

$$560\,(3 + 2 \times 0.20 + 4 \times 0.76) + 1,380\,(4 \times 0.68 + 3 \times 0.31) = 8,643.4$$

8,643.4 million bytes, or about 8.05 GB.

2.5. **Values.** Each output in a transaction includes a "value" that represents the number of satoshis (100 millionths of a bitcoin) being transferred to the output address. This value is stored as an 8-byte signed integer, allowing up more than 9×10^{18} satoshis to be sent. However, Bitcoins protocols dictate how new coins enter the system and place a hard limit of 2.1×10^{15} satoshis. An 8-byte value allows one to send more satoshis than can ever exist. Of course, we also find that most of the values being sent are much smaller than even 2.1×10^{15} and the vast majority fit in a 4-byte unsigned integer. Hence, one approach to compressing the blockchain is to put these values in a data structure whose size is adaptable as needed.

We queried $54,947,133$ outputs from the Bitcoin blockchain to gather statistics on the sizes of values being sent. We found the following.

(1) 45,402,811 values (82.6%) are larger than 2 bytes.
(2) 23,631,735 values (43.0%) are larger than 3 bytes.
(3) 2,600,279 values (4.7%) are larger than 4 bytes.
(4) 18,532 values (0.03%) are larger than 5 bytes.
(5) No values are larger than 6 bytes.

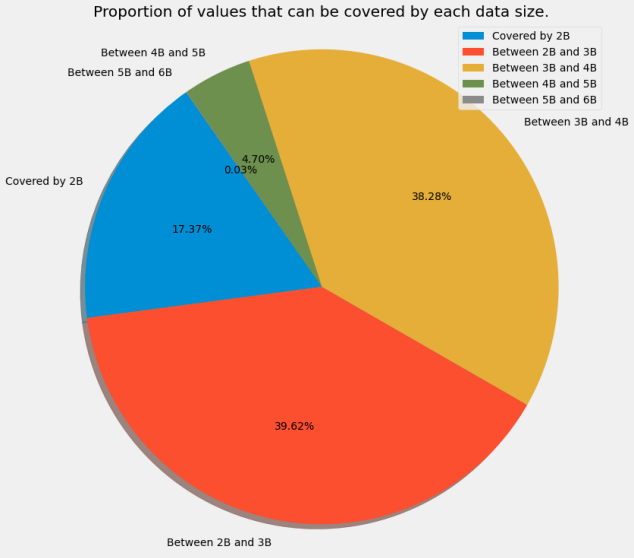


FIGURE 3. Proportion of values that can be covered by each data size.

We can see that over 95% of the values can be covered by 4-byte unsigned integers. Even for 3 bytes, 57% of the data can be covered. We propose the following strategies:

(1) Encode the values with 2 kinds of data types: 4 bytes and 8 bytes unsigned integers. In this case, we need one extra bit as flags for decompression indication.
(2) Encode the values with 4 kinds of data types: 3 bytes, 4 bytes, 6 bytes and 8 bytes unsigned integers. In this case, we need two extra bits as flags for decompression indication.
(3) Encode the values with 4 kinds of data types: 2 bytes, 3 bytes, 4 bytes and 8 bytes unsigned integers. In this case, we need two extra bits as flags for decompression indication.
(4) Encode the values with 8 kinds of data types: 1 byte, 2 bytes, ..., 8 bytes unsigned integers. In this case, we need three extra bits as flags for decompression indication.

Assuming this sample is representative of the transactions in the blockchain, then given that there are approximately 1.5×10^9 total outputs [Bit], we can expect to save

- about 5.5 GB of data if we use method 1;
- about 6.5 GB of data if we use method 2;
- about 6.4 GB of data if we use method 3; or
- about 6.4 GB of data if we use method 4.

It seems that method 2 is the best among these approaches, which could give us an approximate 2% compression rate, based on the fact that the size of the current Bitcoin blockchain is about 300 GB.

3. Implementation

We have created a toy implementation of some of the compression schemes descrived in § 2, available at `http://github.com/emmacneil/btcompress`. The blockchain, when downloaded, is split up into multiple .dat files, approximately 128 MB each. Our toy implementation compresses and decompresses a single .dat file at a time. It implements compression of transaction hashes, version numbers, flags, lock times, sequence numbers, and values as outlined in § 2.2, § 2.4, and § 2.5. Due to time constraints, we did not implement compression of addresses and scripts as in § 2.1 and § 2.3.

We were able to compress the 128 MB file down to 117 MB, 91.4% of its original size. Of course, by implementing the rest of the methods described in this paper, we would bring that figure lower. We note also that the results are dependent on the choice of .dat file. Files corresponding to early blocks in the blockchain have proportionately fewer transactions and would yield worse results. Conversely, by compressing a single file at a time, we cannot take full advantage of the repetition of hashed values. By compressing multiple files at a time, we could achieve better results.

4. Conclusion

In § 2, we analyzed the repeated or redundant patterns in the Bitcoin blockchain. We then discussed approaches to compression based on those patterns. The compression rates we can get from each of those approaches are summarized in Table 6. If applied to to the whole blockchain, we expect a savings of about 58.1 GB, a compression rate of approximately 19.6%. In section § 3, we implement a few of the methods from § 2 to a portion of the blockchain and achieve a compression rate of 8.6%.

Patterns	Sections	Compression Amount	Compression Rate*
Indexing Repeated Addresses	2.1	21.6 GB	7.3%
Indexing Repeated Transaction Hashes	2.2	16.0 GB	5.4%
Indexing Scripts	2.3	5.9 GB	2.0%
Version Number, Flag, Lock Time, and Sequence Numbers	2.4	8.1 GB	2.7%
Compressing Redundant Values Data Type	2.5	6.5 GB	2.2%
Overall		58.1 **GB**	19.6%

* Compression rate is obtained as dividing the compression amount by the total block size 295 GB by the time of Aug 25, 2020 [Blo].

Table 6. Separated Compression Efficiency Achieved by Each Approach

Our methods of compression are applicable not only to the Bitcoin blockchain, but also other cryptocurrencies such as the Divi Project where transaction data includes repeated values and values represented by more space than is needed. Our methods

may also be applied to data analyses that do not require the entire blockchain data. We consider the case of an analyst who wishes to study the graph structure of the blockchain's transactions and may be interested in storing transaction hashes and addresses of senders and receivers, but not scripts and version numbers. Applying some of our compression methods may mean the difference between their data fitting in RAM or not, which would greatly speed up the analysis.

5. Future Work

There are still more simple analyses that can be done. The witness section of a transaction may contain compressible data, but we have not explored this, due in part to not understanding what data was contained within until late in the project. Indeed, the witness section contains addresses and hashes much like input script signatures.

An output script typically contains a hashed value. An input that spends it has in its script the unhashed value. The former can be derived from the latter, meaning that we do not need to store both. A full analysis of how much can be saved has not been done.

In § 2.3, we focused on compressing output scripts. A P2SH output script may correspond to an input whose script, in turn, contains a P2PKH script. That is, for some input scripts, the methods of § 2.3 may be applicable.

Some output scripts may be classified as OP_RETURN scripts. These outputs are unspendable. It is possible to mark some bitcoins as permanently unspendable, a way of burning digital money. However, an OP_RETURN script usually comes with a value of 0 BTC, while the script itself contains metadata written in plain English, Chinese, or some other language. Natural plain text can be compressed effectively using variable-length encoding algorithms such as Huffman Coding, another avenue for compressing the blockchain. However, OP_RETURN scripts are only a small minority of outputs (about 3%), and we expect such compression to save hundreds of megabytes out of 295 GB.

We have focused primarily on exploiting the particular structure of Bitcoin's file format to achieve compression. We have proposed, among other things, to place frequently reused values such as transaction hashes and addresses into lookup tables. These tables, in turn, may be amenable to further compression by more general compression algorithms. This returns us to one of the original questions posed to us: "What compression ratio can we achieve for an ordered sequence of cryptographic hashes?" One could analyze how different generic compression algorithms perform on a sequence of hashes and whether reordering the hashes impacts the results.

Finally, even if we could achieve a compression ratio of 50%, the compressed block chain would still be over 100 GB in size, too large for small devices with storage space constraints. One might ask whether a cryptocurrency can be designed that allows even small devices to participate as full nodes on the network, requiring a blockchain on the order of at most a few gigabytes. We wonder whether a new cryptocurrency model can be defined whereby individual nodes can independently verify the validity of transactions without storing the entire history of transactions or depending on

third parties. Perhaps an imaginative application of concepts such as zero-knowledge proofs, cryptographic set membership testing, homomorphic encryption, and Bloom filters would permit such a scheme.

Acknowledgments

The authors would like to thank our industry mentor, Germàn Luna, from the Divi Project for his guidance, support, and expertise. We'd also like to thank our academic mentor, Cuneyt Akcora, from the University of Manitoba for his helpful insight and advice. Finally, we'd like to thank Mitacs and the M2PI team and especially Kristine Bauer for organizing this amazing opportunity.

References

[Bit] Bitcoin transaction output statistics. `https://bitaps.com/statistic/outputs`. Accessed: 2020-08-25.

[Blo] Blockchain size. `https://www.blockchain.com/charts/blocks-size`. Accessed: 2020-08-23.

[Div] Divi project. `https://diviproject.org`.

[LLW] Eric Lombrozo, Johnson Lau, and Pieter Wuille. Segregated witness (consensus layer). `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`. Accessed: 2020-09-03.

[Mer80] Ralph C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy,1980*, pages 122–134. IEEE Computer Society, 1980.

[Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[Tot] Total number of transactions. `https://www.blockchain.com/charts/n-transactions-total`. Accessed: 2020-08-23.

[TS16] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys Tutorials*, 18(3):2084–2123, 2016.

E-mail address: `iplau@sfu.ca`

E-mail address: `shang.li1@ucalgary.ca`

E-mail address: `macneil.evan@ucalgary.ca`

E-mail address: `amcsw087@uottawa.ca`

E-mail address: `shukla2@ualberta.ca`

E-mail address: `yanhong.xu1@ucalgary.ca`

Math Industry Workshop

THORON DETECTION IN RADON SOURCES

EDWARD TIMKO, STEPHEN STYLES, AND LIAM WRUBLESKI

Abstract. The detection of radon levels is important for reducing home and workplace exposure to ionizing radiation. For the detector under consideration, this detection is done by counting the number of alpha decays in any given time period, and a single measurement cannot distinguish between the different isotopes of radon that may be present. In this paper, we present a method to approximate the relative amounts of radon-222 and radon-220 in a particular sampling sequence by performing a linear regression to the theoretical expected count rate from each of these isotopes.

1. Problem Statement

Residential radon (Rn) progeny exposure is "the leading cause of lung cancer in non-smokers, and the second leading cause of lung cancer in smokers" [1]. Uranium (U) and thorium (Th) in the soil eventually decay into radon, which can then seep into basements and low-lying areas of the house. The two main radon isotopes are Rn-222, which is part of the U-238 decay chain, and Rn-220, also called thoron, which is part of the Th-232 decay chain. There is currently much interest in the Rn-220 contribution to radon progeny exposure, which has so far been largely ignored. Though Rn-220 has a relatively short half life and usually decays before it reaches the living areas in a house, its radioactive progeny can still pose a problem.

Radon is chemically inert, and is most often detected by its decays. Environmental Instruments Canada (EIC) produces a Radon Sniffer [2], which is used by radon mitigators and building scientists to find radon entry points. The sniffer works by pumping air through a filter that removes all radon progeny, after which it is passed into a detector that counts alpha particle emissions from the decaying particles. The detector only counts the total number of alpha decays in a given period, so it cannot distinguish between Rn-222, Rn-220, or their progeny without further processing. Currently, these sniffers assume the only radon species present is Rn-222.

The problem we were presented with was to determine a sampling scheme and algorithm that can reliably determine the approximate amounts of Rn-220 and Rn-222 in any particular radon-containing sample of air, given the existing capabilities of the sniffer.

2. Method

2.1. **Primer on Radioactive Decay.**

The nuclei of some atoms can randomly and spontaneously decay. Such atoms, and the substances they comprise, are said to be *unstable*, or *radioactive*. Nuclear decay is accompanied by the emission of a particle.

There is a variety of particles that can be emitted on decay, but here we are only concerned with two : alpha (α) particles and beta (β) particles. The type of particle emitted in the decay is called the *mode* of that decay. Alpha or beta decay change the *nuclear species* (i.e. the number of protons and neutrons) of the given nucleus. Another common mode of decay is gamma decay, but this does not alter the nuclear species, and for this and other reasons it does not play a part in the problem at hand. The nuclear species of an atom is also known as that atom's *nuclide*.

When an atom decays, the result may also be unstable. A sequence of such decays form what is known as a *decay chain*, where an unstable substance A decays into another unstable substance B, which itself decays into substance C, etc. These decay chains continue until a stable nuclide is reached. An important quantity describing any radioactive substance is its half-life $t_{1/2}$, which is the amount of time over which a given particle of the substance has a 50% probability of having decayed. Equivalently, it is the time by which 50% of the substance is expected to have decayed.

The last point to make is that the probability of decay for each atom in any interval of time depends only on the length of that interval (i.e. the decay process is memoryless). It does not depend on how long that atom has existed overall, nor does it depend on the atoms around it. As such, mixing different radioactive substances does not change their individual behaviour. From the memoryless property of the process, it follows that the time it takes for a given atom to decay follows an exponential distribution, while the number of decays in a given period from a large amount of a pure radioactive substance approximately follows a Poisson distribution.

2.2. **Modelling Expected Values.** Suppose at time 0 you have a collection of $N(0)$ radioactive atoms, all of a single nuclide. On average, the number $N(t)$ of atoms you expect to find remaining at time t is given by

$$N(t) = N(0)e^{-\lambda t},$$

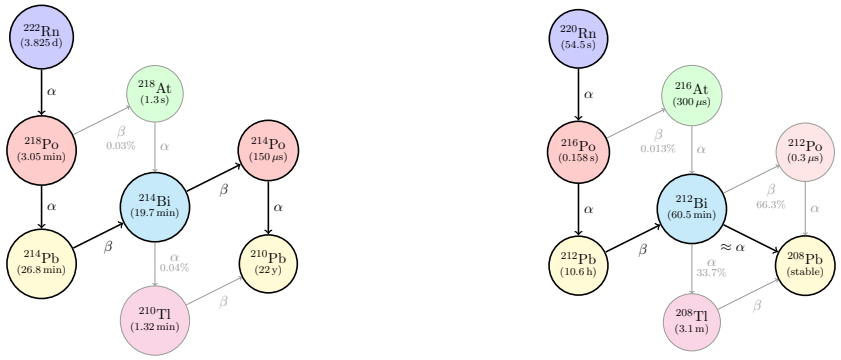
where λ, called the *decay constant* of the nuclide, is related to the half-life $t_{1/2}$ by $\lambda = \ln(2)/t_{1/2}$. The decay rate of this collection of atoms at time t is $\lambda N(t)$.

Before we proceed, a word about units. Radioactive quantities are frequently described not by mass or number of particles but in terms of *activity*, which in the equation above is the quantity $\lambda N(t)$. Activity carries units of decays per unit time. The most common units of activity in the applications being considered are becquerels (Bq) and picocuries (pCi). By definition, 1 Bq is 1 decay per second, and 1 pCi is equivalent to 0.037 Bq. In the context of radon mitigation, quantities are usually described in activity per volume, and the units are becquerels per cubic metre ($\mathrm{Bq/m^3}$) and picocuries per litre (pCi/L). One finds that 1 pCi/L is equal to 37 $\mathrm{Bq/m^3}$.

As mentioned before, the progeny of a radionuclide can also be radioactive, resulting in so-called decay chains. The word "chain" may be somewhat misleading, as

some radionuclides can decay in more than one way. For example, Bi-212 decays by alpha emission into Tl-208 with a probability of 33.7%, and decays by beta emission into Po-212 with a probability of 66.3%. The decay series which are relevant to this project are those of U-238 and Th-232, the relevant portions of which are illustrated in Figure 1.

As can be seen from Figure 1, many of the alternative decay branches occur only with small probability. Because these routes are improbable in most cases and so will contribute little to the decays seen by the detector, we neglect them in the model. Additionally, as Po-212 has a half-life of only 0.3 μs and the beta decay from Tl-208 is not seen, we neglect the different decay paths from Bi-212 and assume that Bi-212 decays directly to Pb-208 by alpha emission. Normally, we end a chain at a stable nuclide where it will no longer decay. However for Rn-222, we stop the decay at Pb-210 as it's half-life is 22 years, which can be considered stable for our model. More precise values for these decay chains can be obtained from NuDat[3], but does not significantly change the model. As a consequence, the decay chains we consider in our model are linearly ordered.



(a) Rn-222 Decay Chain **(b)** Rn-220 Decay Chain

Figure 1. Decay series for Rn-222 and Rn-220, respectively. These are based on [4, Chapter 15]. The effective linear chains are indicated in bold.

Consider a (linearly ordered) decay chain $X_0 \to X_1 \to \ldots \to X_\ell$ in which the decay constant of X_j is λ_j for $j = 0, 1, \ldots, \ell - 1$, and X_ℓ is stable. We assume throughout that $\lambda_0, \lambda_1, \ldots, \lambda_{\ell-1}$ are distinct and positive. Starting at time 0 with a collection $N_0(0)$ atoms of X_0, $N_1(0)$ atoms of X_1, etc., let $N_0(t), N_1(t), \ldots, N_{\ell-1}(t)$ denote the expected number of atoms of the respective type in the given decay chain remaining at time t. When $0 < j < \ell$, the number $N_j(t)$ can change over time either by the decay of an atom of type X_j into one of type X_{j+1} (decreasing $N_j(t)$ in the process) or by an atom of X_{j-1} decaying into an atom of X_j (thereby increasing $N_j(t)$). The time evolution of the ensemble is thus described by the following system of ordinary differential equations:

$$\frac{dN_0}{dt} = -\lambda_0 N_0, \quad \frac{dN_j}{dt} = -\lambda_j N_j + \lambda_{j-1} N_{j-1} \quad (0 < j < \ell).$$

Compare with Equation (5) in [4, Chapter 15].

Recall that the detector does not measure the number of atoms remaining $N_j(t)$ but instead decay counts, the expected number of which in an interval $(s_1, s_2]$ is given by a sum of expressions of the form $\int_{s_1}^{s_2} \lambda_j N_j(s)ds$. Nevertheless, it will be in our interest to solve for N_j first. The solutions of the system are provided by the *Bateman Equation* [5] when $N_1(0) = \cdots = N_{\ell-1}(0) = 0$:

$$N_j(t) = \frac{N_0(0)}{\lambda_j} \sum_{r=0}^{j} \left(\prod_{q=0, q\neq r}^{j} \frac{\lambda_q}{\lambda_q - \lambda_r} \right) \lambda_r e^{-\lambda_r t}.$$

Thus the number of decays by atoms of the j-th type in the time interval $(s_1, s_2]$ is

$$\int_{s_1}^{s_2} \lambda_j N_j(s)ds = N_0(0) \sum_{r=0}^{j} \left(\prod_{q=0, q\neq r}^{j} \frac{\lambda_q}{\lambda_q - \lambda_r} \right) (e^{-\lambda_r s_1} - e^{-\lambda_r s_2}) \tag{2.1}$$

Since our detector only detects alpha decays, the number of counts we expect to see in the time interval $(s_1, s_2]$ is given by

$$\sum_{j \ \alpha\text{-decays}} \int_{s_1}^{s_2} \lambda_j N_j(s)ds.$$

As each of these expressions shares the same independent variable $N_0(0)$, we may determine the total counts expected from this decay chain in any fixed interval as some function of only $N_0(0)$, provided we are working with a single decay chain. For the problem under consideration, we must work with two. The decay chains for Rn-222 and Rn-220 do not intersect, and thus the total counts from a collection of both of these isotopes of radon is the sum of the counts from the individual decay chains. In particular, the expected total number of counts in a time interval $(s_1, s_2]$ from an initial mixture of $N_{222}(0)$ atoms of Rn-222 and $N_{220}(0)$ atoms of Rn-220 is given by an expression of the form

$$N_{222}(0) \int_{s_1}^{s_2} f_{222}(t)dt + N_{220}(0) \int_{s_1}^{s_2} f_{220}(t)dt,$$

where f_{222} and f_{220} are functions derived from (2.1). This indicates that linear regression will provide an effective means of extracting the coefficients to determine the quantities of Rn-222 and Rn-220 present in a given sample.

Using the expected number of decays and the observed counts, we use a linear regression model to estimate our initial amount of each isotope. Letting P be the total number of sampling periods, we fit the model:

$$y_i = N_{222}(0)x_{222i} + N_{220}(0)x_{220i} + \epsilon_i$$

where y_i is the number of alpha particle decays observed in the time interval $(s_{i-1}, s_i]$, $x_{222i} = \int_{s_{i-1}}^{s_i} f_{222}(t)dt$, $x_{220i} = \int_{s_{i-1}}^{s_i} f_{220}(t)dt$, s_i denotes the end of sample period i, for $1 \leq i \leq P$, and s_0 denotes the beginning of sample period 1. Note that

the errors $\epsilon_1, \ldots, \epsilon_P$ are random but not independent of one another.

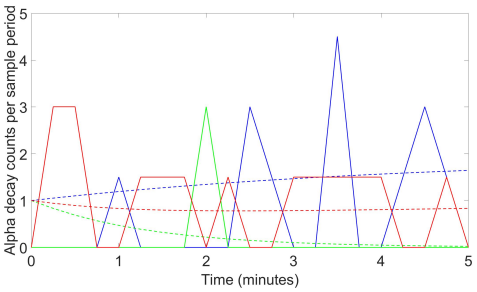
2.3. **Simulating Experimental Values.** In order to design the optimal sampling scheme, we must be able to evaluate the effectiveness of the linear regression model for any particular scheme. Determining this analytically is not straightforward, as the counts in any particular period are random and depend on the state underlying the counts in the previous period, as well as the length of each period. Even for a single particle, the time at which it decays into any other particle in the chain is the sum of random variables from several exponential distributions with different parameters. Therefore, we elected to evaluate the effectiveness of a scheme over test data, which we generated. We also narrowed the scope of sampling schemes under consideration to schemes where counts are reported at fixed times, over a fixed duration of time, where the contents of the cell are held constant over that duration. This sampling scheme does also require that we consider a time offset in order to account for the time it takes to fill the chamber, or else we would underestimate the Rn-220. Phrased in terms of the previous section, $s_i = T_s \cdot i + T_f$ for $0 \leq i \leq P$, where T_s is time of the sampling period and T_f is the time taken to fill the chamber.

This differs from the typical method used by radon mitigators in the field, which is continuous sampling of some number of environments, interspersed with flushing periods. However, based on our early results, we do not believe that this is a viable scheme to determine the relative amounts of Rn-220 and Rn-222, as the half-lives are too different and variations during pumping are not well understood. This change in procedure is a reasonable compromise, as in most circumstances knowing the Rn-220 concentration isn't necessary, and in those situations where it is the sampling scheme we propose is acceptable.

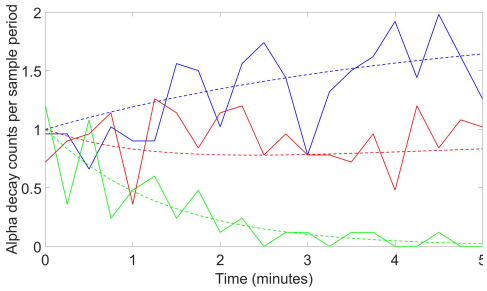
Generating test data for a sampling scheme of this type is relatively straightforward, but as the random processes involved are complicated, the best method we were able to come up with is not particularly efficient, having time complexity of $O(N \cdot \ell)$, where N is the number of particles in the chain at time $t = 0$ and ℓ is the length of the chain. We assume that at time $t = 0$, for a given decay chain, that only the substance at the head of the chain is present. Then, for each particle of that substance, we sample the *particle lifetimes*, which is the time that particle will take to decay from each substance in the decay chain to the next substance. Then the time at which that particle decays into each substance is the cumulative sum of the particle lifetimes up to that point. Using these decay times for the particles, we can then count how many alpha decays occur in each sampling period, which gives us the observed count for that sampling period.

2.4. **Model Visualization.** To demonstrate the relationship between the expected counts and the generated count data, we graph the expected count rate and a corresponding sample run for a series of concentration levels in Figure 2. Homeowners are advised to take action if the radon activity is greater than 4 pCi/L in air [6],

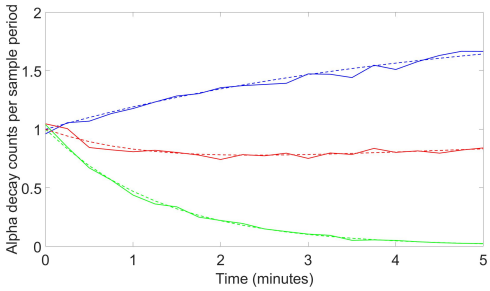
sub-slab measurements can be on the order of 100 pCi/L, and the largest value that Environmental Instruments Canada has reported to us was around 8000 pCi/L. By showing these values, we can see the amount of variation implicit to this problem, and note that higher activities make it much easier to determine the amounts of Rn-222 and Rn-220.



(a) Activity level of 4 pCi/L



(b) Activity level of 100 pCi/L



(c) Activity level of 8000 pCi/L

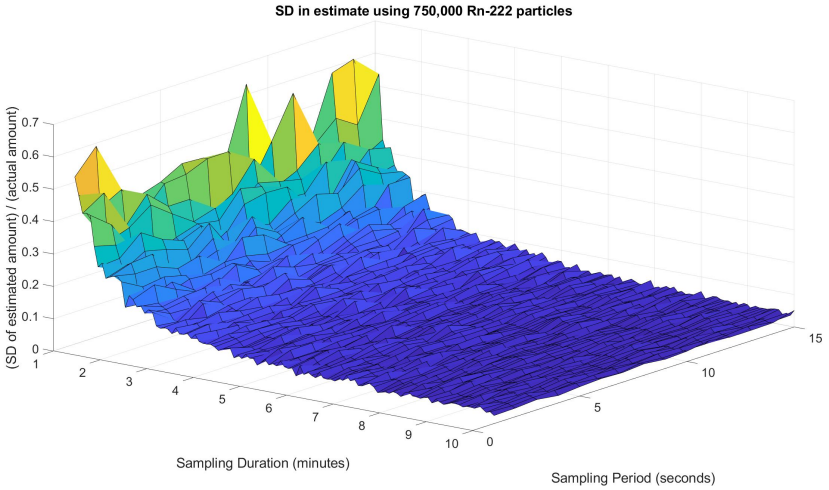
Figure 2. Sample runs and expected values for various mixtures of Radon isotopes (blue: pure Rn-222, red: Rn-222/Rn-220 mixture, green: pure Rn-220; dashed: expected counts, continuous: observed counts)

3. Parameter Estimation

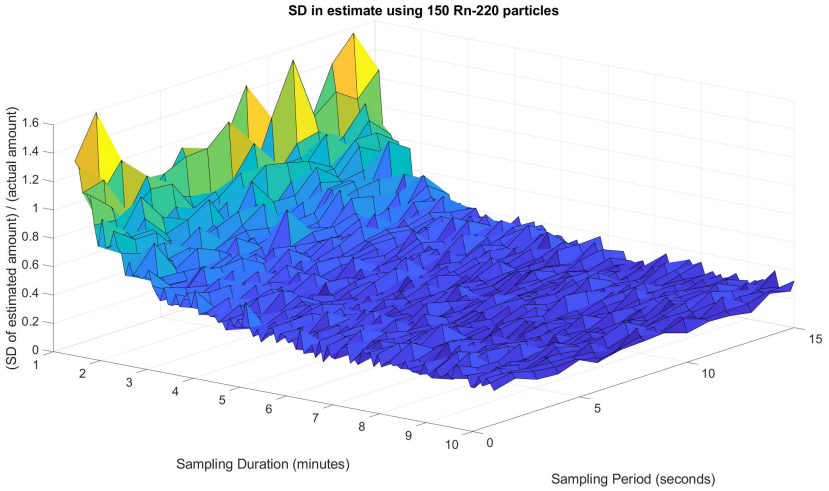
3.1. **Grid Search.** To find the sampling time and sampling period which minimize the variance in the estimates, we run a grid search that calculates multiple regressions on a fixed set of concentrations. Running a new sample set of decays for each regression is done to avoid over-fitting the model to a training set of data. From here, we calculate the mean and standard deviation of the estimated coefficients for various combinations of the sampling time and period to find appropriate values, attempting to minimize both the error relative to the true concentration, and the standard deviation in the estimates. With any sampling time below 90 seconds, we were not able to see a separation between the Rn-222 and Rn-220. When we sample for longer, our estimates start to level off in accuracy and increased time is no longer needed. From figure 3 we see that the standard deviations level off at approximately 5 minutes duration and are mostly independent of sampling period.

For each potential sampling time and period, the actual model with which estimations can be made takes the form of the $2 \times P$ matrix described by the left pseudo-inverse of X, $(X^T X)^{-1} X^T$, where the $P \times 2$ matrix $X = \begin{bmatrix} x_{222} & x_{220} \end{bmatrix}$ is formed using the values for x_{222} and x_{220} determined using the grid search, and when given a sample vector y, the estimated amounts $\hat{N}_{222}$ and $\hat{N}_{220}$ are determined by

$$\begin{bmatrix} \hat{N}_{222} \\ \hat{N}_{220} \end{bmatrix} = (X^T X)^{-1} X^T y$$



(a) Standard deviation of 25 different linear regression estimates for radon

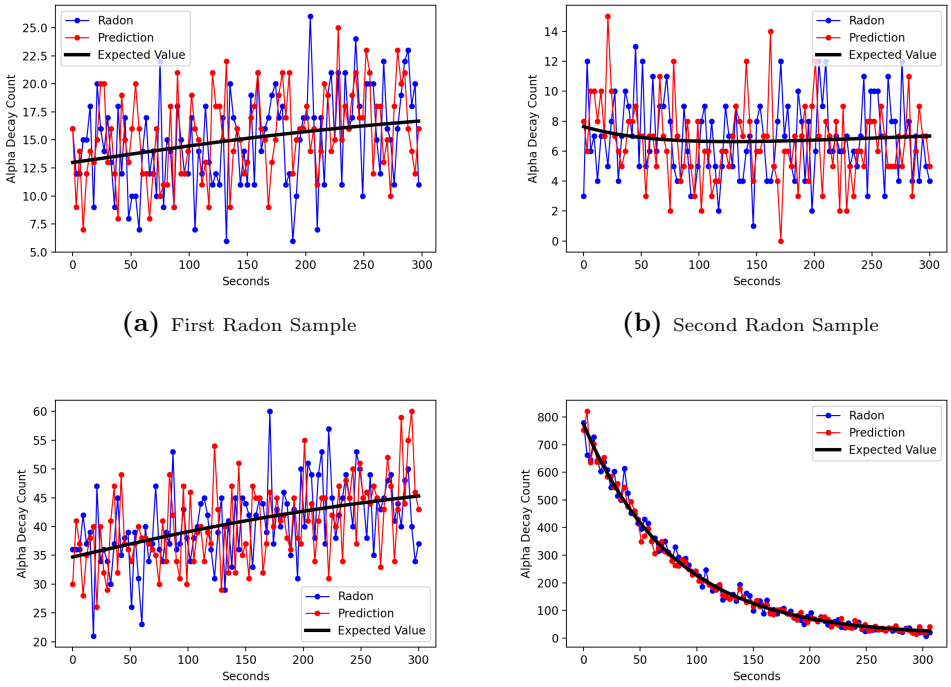


(b) Standard deviation of 25 different linear regression estimates for thoron

Figure 3. Standard deviation in estimated values over varied sampling parameters

3.2. **Results.** After tuning our sampling parameters, we requested and received data on different radon and thoron sources from EIC. Using this data, we fit our linear regression model and solve for our initial concentrations of Rn-222 and Rn-220. Using those estimates rounded up to the next whole number, we then solve for what the expected number of alpha decays would be over our 5 minute interval and generate test data for the same amount of each nuclide. The following graphs highlight the accuracy of our model.

From figures 4 (a)-(d), we see that our model can predict the radon levels for a typical basement as well as the extreme cases where we are measuring high concentrations of Rn-220. These measurements where done all in the same day so the progeny of the previous test would be included in each subsequent test, therefore causing more error into our model. Although our predictions appear to be fairly accurate, they rely on the assumption that our radon sniffing chamber is empty. This progeny remaining in the sniffer will cause our model to overestimate the true values and a correction to the alpha decay count could increase the effectiveness of our model.



(a) First Radon Sample

(b) Second Radon Sample

(c) Third Radon Sample

(d) Thoron Sample

Figure 4. Observed and Predicted Values of Radon Samples

3.3. **How to use the results.** Overall, based on the efficacy of our model on these sample data, and in consultation with Environmental Instruments Canada, we have

selected a sampling period of 3 seconds, over a total duration of 5 minutes. The particular sampling schedule is as follows:

(1) Flush the cell for 90 seconds (this is probably an overestimation of the time required).
(2) Pump air from the spot to be sampled into the cell for 90 seconds (this is also probably an overestimation of the time required).
(3) Stop the pump, to let the sample be measured. Counting for this sequence should begin here.
(4) Let the sample sit for 5 minutes, reporting counts every 3 seconds.
(5) Take the reported counts as a 100 element column vector.
(6) Multiply this vector by the 2×100 matrix $(X^TX)^{-1}X^T$, where X is the 100×2 matrix $\begin{bmatrix} x_{222} & x_{220} \end{bmatrix}$, using x_{222} and x_{220} determined from the solutions to the Bateman equations over the intervals (Python code to generate each of x_{222}, x_{220}, X, and $(X^TX)^{-1}X^T$ is available on the project's GitHub repository)[7]. Note also that:
    - the values x_{222} and x_{220} also need to account for the delay of pumping (in this particular setup, this can be done simply by computing the solutions for the first 130 intervals, and ignoring the first 30); and
    - this matrix is constant for a given sampling schedule, and does not need to be recomputed each time.
(7) This gives the 2-element column vector $\begin{bmatrix} \hat{N}_{220} & \hat{N}_{222} \end{bmatrix}^T$, which estimates the number of particles of each isotope that were in the chamber at the beginning of pumping (under the various assumptions of the underlying physics within the chamber).
(8) Assuming that λ is given in s^{-1}, the absolute activity of each can then be determined in Bq as $\lambda\hat{N}$, and the volumetric activity can then be determined using the volume of the scintillation cell.

4. Further Research

Further analysis could be done on the model to consider various constraints. A non-negative least squares regression would constrain the estimated amounts of each isotope to be positive. In our experimentation with this model, it tended to remove one of the covariates and predict either pure Rn-222 or pure Rn-220. Other regression methods such as Poisson regression or negative binomial regression could be considered to account for the discrete output, however when we compared our results to a normal Q-Q plot, the model fairly accurately followed a normal distribution, so these other methods may be unnecessary. Adding a correction term to the alpha decay count to account for the progeny left in the chamber could also increase the accuracy of the model.

To speed up the optimization, instead of considering a new dataset for each regression we could consider a fixed number of data sets for training. This would allow us to reduce the computation time, allowing us to check a finer grid of parameters.

Grid search is not a particularly effective method of finding the optimal configuration, so another optimization algorithm may be useful. The reason this was not done here is that the optimization variables include the sampling period and duration, and as these determine not only the distributions of the random variables under consideration, but also the constants used in any optimization model, this makes creating an analytic model which can be given to a solver very difficult. Stochastic methods using randomly generated data are better suited to this task, but still encounter issues.

Developing a model which determines the amount of each nuclide over time when the chamber is being filled from a source of particular concentration may allow this model to use additional data from the fill period or the purge period, and may also allow it to be extended to continuous sampling.

The relative error in the amount of Rn-220 is quite high. To achieve activity commensurate with that of Rn-222, one requires much lower concentrations of Rn-220, so the relative variability in counts per period is much higher. There may be heuristic methods of determining whether there is Rn-220 that could supplement this model, which may reduce the error caused by this, but we do not consider them here.

Due to the aforementioned inefficiency, the method we used for generating random data is only viable for relatively small numbers of particles ($N \ll 10,000,000$). For numbers of particles greater than this, another method should be developed. We sidestepped this issue by running trials in parallel, but this solution will not work for significantly larger numbers of particles.

5. Conclusion

Using the 90 second procedure of continuous sniffing, we were able to measure how much Rn-222 is in our measurements. However, using this same sample scheme, we were unable to consistently differentiate between data obtained by measuring pure Rn-222 and that obtained from a mixture of Rn-222 and Rn-220. As such, we evaluated different sampling times and periods in which to most accurately estimate the quantity of each radon isotope. With this new sampling scheme of 3 second samples counted over a time of 5 minutes, we developed an algorithm by which the amount of Rn-220 in a sub-slab measurement of radon isotopes can be estimated through a linear regression model. In tandem with existing methods, this can be used to assist radon mitigators in making mitigation decisions by ensuring they have as much information as possible about the situation, without drastically increasing the time for each test.

Acknowledgements

We wish to thank Kai Kaletsch from Environmental Instruments Canada for assisting us in his role as an industry mentor, and for providing an interesting challenge to work on. We also wish to thank the Pacific Institute for Mathematical

Sciences (PIMS) for running the $Math^{Industry}$ workshop, and the organizers within PIMS who put in a great deal of effort to make it happen.

References

[1] Lung Cancer Canada, "Radon", https://www.lungcancercanada.ca/Lung-Cancer/Radon.aspx#:~:text=Health%20Canada%20recently%20increased%20its,deaths%20globally%20are%20radon%20induced.

[2] Environmental Instruments Canada, "CT007-R Radon Sniffer", https://radonsniffer.com/

[3] Brookhaven National Laboratory, "NuDat 2", https://www.nndc.bnl.gov/nudat2/

[4] R. B. Leighton, *Principles of Modern Physics*, McGraw-Hill, New York, 1959.

[5] H. Bateman, *The solution of a system of differential equations occurring in the theory of radioactive transformations*. Proc. Cambridge Philos. Soc, Vol. 15, No. pt V (1910) pp. 423427.

[6] US Environmental Protection Agency, "Health Risk of Radon", https://www.epa.gov/radon/health-risk-radon

[7] https://github.com/StephenStyles/Radon