HopScotch Final Report

Song Bai, Wei Chen, Rohit Chugh, David Feng, Anjali Ramchandani
Department of Computer Science
University of California, Riverside
Riverside, CA

1. INTRODUCTION

In the last decade, there has been exponential growth in the food, hospitality, and tourism industry. With exposure to social media and increasingly affordable travel options, more people are traveling to new destinations than ever before. However, for people to get the most out of their trip, they need to do extensive study and research to find the best places to visit, meet new people, and make friends.

One of the numerous methods to do so is the pub crawl. Pub crawls are predominantly popular in European cities. Bar hopping serves as a social gathering for local tourists and enables participants to meet new friends and become acquainted with new bars in a strange city. Often this requires prior research and knowledge of a new area, difficult for someone in a foreign place.

2. RELATED WORK

2.1. TRAVELING SALESMAN PROBLEM

From a high-level perspective, we can reduce bar crawl pathing down to the Traveling Sales Problem, where each bar is a node and given a starting point, trying to find the shortest route between all bars and return to our origin. This problem dates back to the 19th century, and one of the most intensively studied problems in optimization. Unfortunately, the problem is also NP-Hard, with no known polynomial-time solution for this problem [1]. TSP poses a problem for us as any decent-sized metropolitan area will have hundreds of bars, and iterating through each is a computationally unrealistic process, even within a cluster. Combine this with the notion that we are attempting to path based on not only distance but also star ratings and reviews, a unique variant of TSP is needed to accomplish this.

2.2. HEURISTIC

To combat this, we require the use of comparable fast run times that still yield near-optimal solutions. Two of the most popular heuristics we are considering include Branch and Bound, as well as for Simulated Annealing [2]. We can measure the optimality of these algorithms using the Held-Karp lower bound, which produces a lower bound to the optimal solution.

[3] The Branch and Bound strategy divides a problem to be solved into several sub-problems, and if the best solution found so far costs less than the lower bound for this subset, we need not explore this subset at all [4]. Simulated Annealing works by randomly selecting tours, and updating if we find a more optimal tour. [5] The trick is that we sometimes accept a worse tour temporarily as a potential stepping stone to leave a local minimum, which significantly improves over the naive hill-climbing problem.

2.3. CLUSTERING AND COST BASED ANALYSIS

We will use DBSCAN (Density-based spatial clustering of applications with noise) for clusterings of bar locations into natural districts or neighborhoods before applying TSP. DBSCAN[6] is a density-based clustering method that can identify noises and does not require a pre-set number of clusters. In addition, Yelp businesses are distributed based on the population density. Therefore, we are using this method for clustering the filtered data.

[8] introduced Traveling Salesman Problems with Profits (TSPs with Profits), a generalization of the Traveling Salesman Problem (TSP), and solutions to the TSPs with Profits. In TSPs with Profits, the objective is not necessary to visit all vertices. Each vertex is associated with a profit. The objective is to find a route with a satisfying collected profit (maximized), and travel cost (minimized), implying TSPs with Profits can help us optimize the pathfinding method in our application. [9] Similarly, other work has been done with limited cost constraints, which is invaluable because we are not only looking at the distances between bars but also if they are popular and well-received. An additional dimension to our graph may be necessary.

2.4. DATA CLEANING

Our project needs many dimensions of data, while some of the dimensions of data from the Yelp API might be lost. Some of them can just be fixed using function dependencies, such as some constraints of longitude and latitude are related to specific zip codes. Some of the bar's opening times could be written wrong for not being written in 24h format.

2.5. GOOGLE MAPS

Many other projects using TSP only have a fixed weight on their map. But for our project, we have a flexible weight for each analysis. This can include the time of travel using different means of transportation. We also consider the bound between traveling time and business hours. So the distance of each point can vary a lot from different dimensions.

The Google Maps direction service can search for the best traveling route for our project. It can show the fastest route by setting the method of traveling, the location. So it can be used to calculate the best distance from one place to another. It can be used in both the TSP problem for distance calculation and visualizations.

3. PROJECT PIPELINE

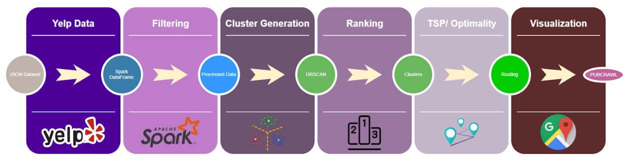


Figure 1: Complete Data Pipeline

3.1. DATASET SCRAPING & PROCESSING

Yelp is the largest active business directory, which made it our obvious choice for gathering data. In our early trials, we used the provided Yelp challenge dataset, which helped form an approach for filtering. Unfortunately, our trials always failed due to incomplete data. Per the dataset, there were only 19 bars in the entire state of California. We wanted something more concrete and to tie together our results to the real world. This required scraping and querying from Yelp in real-time, we selected Yelp API with GraphQL.

The advantage of GraphQL over REST is that we can dynamically request the required fields as a JSON response. Unfortunately, Yelp does not allow constrained search parameters, meaning it will show business listings from outside a location search regardless of the actual query address. For instance, looking for only business listings in the city of Riverside might append another hundred from the surrounding cities.

Our preprocessing step resulted in constraining our example set to only Los Angeles County zip codes, removing the many redundant listings in each query, and filtering only US listings. Each query included the following schema for each business: the address, GPS coordinates, rating, review, URL, and it's corresponding subcategories.

Using this information, we have enough to proceed with clustering.

3.2. CLUSTERING

Using Python's scikit-learn library we can call DBSCAN to generate clusters based on a list of GPS coordinates. For DBSCAN, we have two parameters, epsilon and minPoints. Epsilon controls the maximum distance between the two points to be considered in a cluster. minPoints constraints the minimum number of points to be a cluster. In our results, we found the best values to be .028 and 6, respectively.

A standard way to analyze our data set for epsilon is to calculate 1-NN, and find the distance at which most points are separated by. A figure above shows a plot of points to their nearest neighbor. Take note of the point of maximum curvature. We also choose the 'ball_tree' algorithm and 'haversine' metric as they are explicitly tuned for geo-spatial data and account for the curvature of the earth.

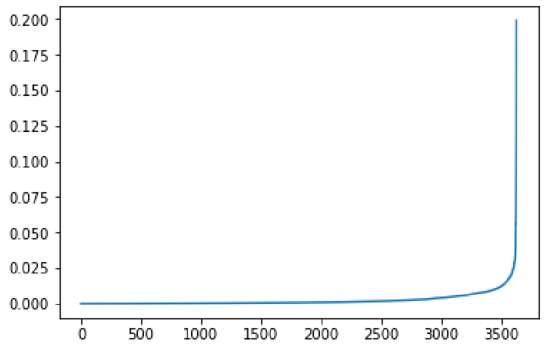


Figure 2: 1-NN distances for each node

3.3. RANKING

The next step is to determine the most valuable listings in each cluster of businesses. This is crucial in finding results that not only account for distances but also value to the user. To do so, we assign a rank to each node in a cluster. To calculate the rank, we examine multiple input signals. The input signals include(but not limited to) the rating of a business and the number of reviews. The normalized score primary values those with a high overall review and then businesses with a large number of reviews.

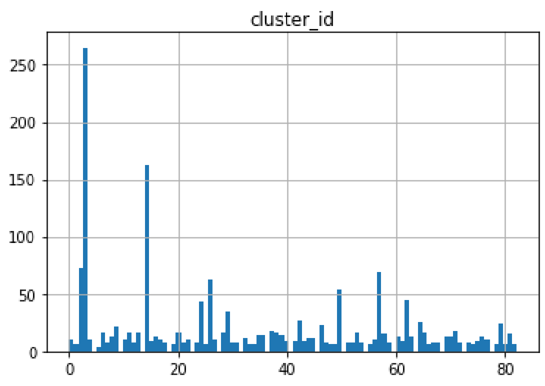


Figure 3: Histogram of # of nodes per cluster

3.4. TRAVELING SALESMAN PROBLEM

Next, we perform our most expensive step of calculating and generating a route for every cluster using TSP:

1. First, we generate a distance matrix of every node in each cluster using the Google Distance Matrix tool for actual road-driving/walking distances rather than the Euclidean distances between each point for higher accuracy.
2. Second, we solve TSP using Google OR-Tools. A specific variant is chosen called Capacitated Vehicle Routing Problem (CVRP). This represents a VRP in which vehicles with limited carrying capacity need to pick up or deliver items at various locations to simulate TSP with Profits. Each cluster is assigned

an average cost, which means that it not only generates the best route but also drops nodes that it finds to be less valuable. We set the capacity of each route accordingly to make sure it does not always go through all the nodes.

3. Finally, we generate the higher scored route for each cluster and pass it to forward for visualization.

```
Dropped nodes: 4 5
Route for vehicle 0:
 0 Load([0.]) ->  6 Load([3.45424251]) ->  3 Load([10.57229061]) ->  2
Load([12.3733206]) ->  1 Load([17.3733206]) ->  0 Load([17.3733206])
Distance of the route: 203m
Load of the route: [17.3733206]

Total Distance of all routes: 203m
Total Load of all routes: [17.3733206]
```

Figure 4: Output from Google's CVRP

3.5. VISUALIZATION

We choose to show it by using the Google Maps API. But there are also several choices. One is to use python for obtaining data from Google Map API and then using the request to send the information to the web interface. Another way is to use the JavaScript to request for the route query from Google Map API And then do the showing stuff just in the web interface. We choose the latter for future compatibility and the ability to adjust per the user's needs.

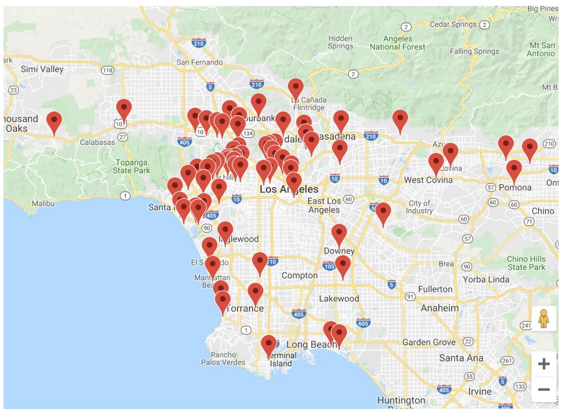


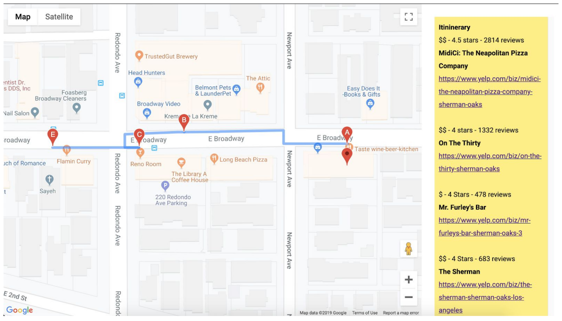
Figure 5: All clusters displayed in the LA County

For each area, we can have hundreds of thousands of routes. But the problem is how should we represent it to the user. There are some solutions to deal with it. For one of Google's own projects, it can also show routes in a list along the left side of the web interface. Those routes are chosen by humans, which shows some unique ways to have a trip in that area.

We use markers on the map to represent the starting point. Users can choose the starting points nearby and elect their preferred bar crawling route. Each time you click a marker, it will show that bar crawling route. We also show the name and location of the bar on the right panel alongside the map.

3.6. DATA TRANSMISSION

Lastly, in building our front-end interface, we need to translate our data from Python to Javascript. There are several ways we can choose to do the transmission. The first way is to pass the JSON to Javascript directly. Unfortunately, this method takes quite a lot of time because the file transmission and parsing both need to be done in the javascript. The second way is to parse the data in Python and then send the data directly to the Javascript using WebSocket. This method is super fast because the data is processed as it is being concatenated. The calculation of the parsing step is moved from javascript to Python. We also elect WebSocket because it is about 22% faster than the equivalent HTTP request.



Itininerary
$$ - 4.5 stars - 2814 reviews
MidiCi: The Neapolitan Pizza Company
https://www.yelp.com/biz/midici-the-neapolitan-pizza-company-sherman-oaks

$$ - 4 stars - 1332 reviews
On The Thirty
https://www.yelp.com/biz/on-the-thirty-sherman-oaks

$ - 4 Stars - 478 reviews
Mr. Furley's Bar
https://www.yelp.com/biz/mr-furleys-bar-sherman-oaks-3

$$ - 4 Stars - 683 reviews
The Sherman
https://www.yelp.com/biz/the-sherman-sherman-oaks-los-angeles

Figure 6: Example A - Route with sidebar information

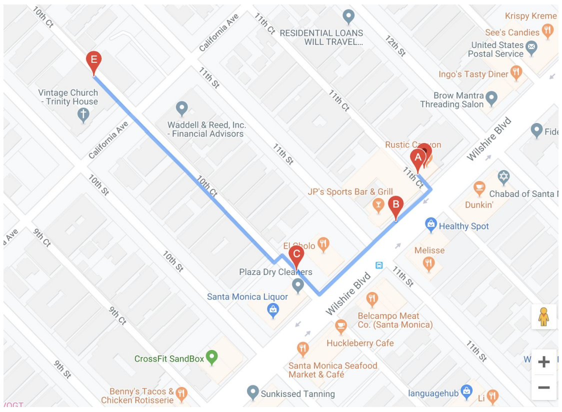


Figure 7: Example B - TSP Routing

4. CONCLUSION AND FUTURE WORK

In conclusion, we have a working demo that shows the power of combining such tools together. From public Yelp data, we can generate efficient pathing routes that work and feel hand-curated. A clean front-end allows a user to capture and use our product to achieve our desired result. Each cluster accurately displays a viable route that accounts for proper pathing, as well as the shortest path.

Given additional runway, there are future features that could extend our project. One of the most significant upgrades would be a better front-end system that allows the user to query and filter by more specific preferences. An example might be the ability to filter by specific sub-categories, like selecting only karaoke and dive bars, versus everything from clubs to pubs.

Another feature would be to limit automatically selected by the user's location and display a dashboard of those nearest to them. From the back-end, we could also enlist the use of each bar's hours, and based on a sliding window, limit real-time results to only those open during certain hours and days of the week. A richer sidebar that provides additional insight into each business along a route, as well as alternative options, would be a great feature as well.

Overall, this project was an exciting deep dive into the world of big data and the application of driving results from a more substantial data source. It was a tremendous team effort to construct and design the proper tools necessary for such a complicated endeavor.

REFERENCES

[1]"Reusing the NP-Hard Traveling-Salesman Problem to Demonstrate That P~NP (Invited Paper) - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7785793. [Accessed: 10- Nov- 2019].

[2]C. Nilsson, "Heuristics for the Traveling Salesman Problem", 2003. [Online]. Available: https://pdfs.semanticscholar.org/7b80/bfc1c5dd4e10ec807c6f56d0f31f8bf86bc6.pdf. [Accessed: 10- Nov- 2019].

[3]C. Pothineni, "Travelling Salesman Problem using Branch and Bound Approach", 2013. [Online]. Available: http://cs.indstate.edu/cpothineni/alg.pdf. [Accessed: 10- Nov- 2019].

[4]J. Walker, "Simulated Annealing: The Travelling Salesman Problem", *Fourmilab.ch*, 2018. [Online]. Available: https://www.fourmilab.ch/documents/travelling/anneal/. [Accessed: 10- Nov- 2019].

[5]S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing", 1983. .

[6] M. Ester, H. Kriegel, J. Sander, X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise” ,1996. [Online]. Available: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

[7]M. Jyoti Yadav, "IJEET - A Review of K-mean Algorithm", Ijettjournal.org, 2013. [Online]. Available: http://ijettjournal.org/archive/ijett-v4i7p139. [Accessed: 10- Nov- 2019].

[8]D. Feillet, P. Dejax and M. Gendreau, "Traveling Salesman Problems with Profits", 2005. .

[9]P. Sokkappa, "The cost-constrained traveling salesman problem", 1990. .

[10]N. Tang, "Big Data Cleaning", 2014. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-11116-2_2. [Accessed: 10- Nov- 2019].

[11]D. Guttentag, "Why tourists choose Airbnb: A motivation-based segmentation study underpinned by innovation concepts", *Uwspace.uwaterloo.ca*, 2016. [Online]. Available: https://uwspace.uwaterloo.ca/handle/10012/10684. [Accessed: 10- Nov- 2019].

[12]T. Jindal, "Finding local experts from Yelp dataset", *Hdl.handle.net*, 2015. [Online]. Available: http://hdl.handle.net/2142/78499. [Accessed: 10- Nov- 2019].

[13]N. Azi, M. Gendreau and J. Potvin, "An exact algorithm for a single-vehicle routing problem with time windows and multiple routes", 2007. .

[14]Z. Hashim and W. Ismail, "Applications of Travelling Salesman Problem in Optimizing Tourist Destinations Visit in Langkawi", 2016. .

[15]C. Bailey, T. McLain and R. Beard, "Fuel saving strategies for separated spacecraft interferometry | Guidance, Navigation, and Control and Co-located Conferences", Arc.aiaa.org, 2012. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2000-4441. [Accessed: 11- Nov- 2019].