Final Year Project

Learning shortest path tours to all pubs in UK

Sabeer Bakir

Student ID: 16333886

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Deepak Ajwani

UCD School of Computer Science

University College Dublin

October 26, 2019

Table of Contents

Abstract

Designing algorithms for NP-hard combinatorial optimization problems over graphs is a complex task. It has been shown with use of good heuristics and approximation algorithms that these problems can be solved in polynomial time. However, these solutions require problem-specific knowledge and trial-and-error. Is it possible to automate this learning process with the use of machine learning classification? In real world applications, the data is very rarely similar however the structures within may not differ as much. In this paper, we will explore a method to solve the Travelling Salesman Problem (TSP) using classification to identify edges that are contained within the tour created by the TSP. The applications of this can be applied in many areas such as, DNA Sequencing, Route Planning, Logistics Management, etc.

Chapter 1: Project Specification

Core

- Identify features of edges that can discriminate between edges in the shortest TSP tour and edges that are not.
- Take random samples of the pub crawl dataset and use a state-of-the-art heuristic on that to get the ground truth for training
- Use the identified features and the ground truth on random samples to train a classification model that learns the edges in the shortest TSP tour
- Evaluate the accuracy of the classification model

Advanced

- Compare the running time vs. optimality trade-off of the learning approach with the Concorde TSP solver (http://www.math.uwaterloo.ca/tsp/concorde.html)
- Improve the accuracy of the classification algorithm by careful feature selection

Chapter 2: **Introduction**

The Travelling salesman problem (TSP) is a combinatorial graph optimization question that is. This problem is NP-hard and has caused considerable interest by theory and algorithm design communities in the past. In the realm of computational complexity, one of Karp's 21 NP-complete problems is the Hamiltonian Cycle Problem, this is a special case of the travelling salesman problem.

The TSP was first introduced in the 1800s by the Irish mathematician Sir William Rowan Hamilton and the British mathematician Thomas Penyngton Kirkman. However, optimisation of the TSP only came about in the 1930s from Karl Menger in Vienna and Harvard. Menger defined the problem as, "Given a finite number of points, with known pairwise distances, find the shortest path connecting the points."[1]

Techniques that are used to solve these graph optimisation problems comes in three main type: exact algorithms, approximation algorithms and heuristics. Exact algorithms always find the most optimal solution however it requires exponential time which does not scale well for large inputs. Approximation algorithms offer the desirable polynomial time solution however it is hard to guarantee the optimality for certain problems. Heuristics are typically fast but require problem specific research and experimenting from the algorithm designers.

Designing algorithms for NP-hard combinatorial optimization problems is a complex task. It has been shown with use of good heuristics and approximation algorithms that these problems can be solved in polynomial time. However, these solutions require problem-specific knowledge and trial-and-error. Is it possible to automate this learning process with the use of machine learning classification? In real world applications, the data is very rarely similar however the structures within the data do not differ. In this paper, we will explore a method to solve the Travelling Salesman Problem (TSP) using classification to identify edges that are contained within the tour created by the TSP. We will see if it is possible to learn the shortest path to all pubs in the UK. The applications of this can be applied in many areas such as, DNA Sequencing, Route Planning, Logistics Management, etc.

Chapter 3: Related Work and Ideas

Reinforcement learning for Combinatorial optimization Reinforcement learning (RL) is used as a natural framework for learning the evaluation function in [2]. An *off-policy* RL algorithm such as Q-Learning was utilized here which updates its rules on the Q-Value rather than looking at past examples to learn. This technique was used over graph problems such as: Minimum Vertex Cut (MVC), Maximum Cut (MAXCUT), and the Travelling Salesman Problem (TSP). This type of approach lends itself to designing greedy heuristics for difficult combinatorial optimization problems. Heuristics as we know are commonly fast but in the area of optimization require certain knowledge about the underlying problem, this approach attempts to build these heuristics whilst learning about the underlying problem.

Deep reinforcement learning

Chapter 4: **Data Considerations**

In this section you should characterise the nature and scale of the data you are working with. Outline the shape of the data, where you expect to obtain it, and the size of the data. Is it static or dynamic, local or remote, stored or streaming? Is it raw or structured? Is it unfiltered user data, or is it curated by a specialist? What is your rationale for using this data and not other data? If your project looks at callout times for Spanish ambulances, usage rates of French parking lots, alcohol consumption in Germany, and so on, then explain why you are not using Irish data for the project. Indicate the data-cleaning processes that you anticipate will be necessary. What licensing restrictions, if any, apply to your data? Will you be making this data public after your project is completed? Are there any privacy or ethical issues with how the data is to be collected or used? If so, discuss here.

Some or many of these questions may be moot in the case of specific projects, but you should provide compelling answers to any that seem relevant. Since this provides the foundation for your project, your reviewers will be looking closely.

Chapter 5: Outline of Approach

In this section present an outline of your considered approach to the problem at the centre of your project. Clearly present your design choices, or your choice of algorithms, and any pertinent model parameters. For instance, if you plan to use a genetic algorithm, outline here a sense of your fitness function, major variables, population size, and so on, so that your reviewers can critique your choices. If you opt for a neural architecture, describe your chosen framework, and motivate the number and kinds of layers in your network. In short, be specific about the choices you are committing to at this stage. Being vague and non-specific will not help your case, as your report will be graded in large part on the specificity and perceived wisdom of your choices. Remember also that feedback is intended to help you as you progress to the next stage of your project. If you give reviewers little to chew on, they will not be able to give you specific feedback and guidance.

Chapter 6: **Project Workplan**

In this section you will present a work plan for the remainder of your project. Show that you have considered the issues carefully, and that you can be trusted to lead a research or development effort. Be as specific as you can about the time you expect to allocate to each work component, and the dependencies they have to each other. A Gantt chart is helpful in this respect, but do show some sense in how you present your plan. A naïve understanding makes for a simplistic plan.

A key part of a successful project is evaluation. It is not enough to just state that your project is a success, or that your friends seem to like it. You must have a plan for evaluating the end result. How you evaluate will depend on the nature of your project, and you should have a serious conversation with your supervisor about evaluation before you get to this stage. Will your work yield quantitative results that can be compared to past work or to established benchmarks? Does your work consider different configurations of a system or a solution that you can compare to each other, allowing you to empirically find the best one? Do you have a sample user pool for your planned application, and are they willing to give you structured qualitative and quantitative feedback (e.g. via a questionnaire)? However you plan to evaluate your project, please sketch your intentions here.

Chapter 7: **Summary and Conclusions**

In this section you will sum up your report, draw some conclusions about your work so far, and make some general observations about the work to come. You may also use this opportunity to express points of view, or make factual claims, that are more pertinent here than in other sections of the report. If your project raises some ethical concerns, for example about how data or users are treated, then address them here in a thoughtful manner.

Regarding this document, here are some concluding points that you should keep in mind when writing your own. You may use screenshots in your report, but do not overfill your report with them, or with figures of any kind. Make sure that figures earn their keep, and are not just present as space fillers or as eye candy. If you use diagrams or figures from other people's work, including the web, be sure to cite the creator in the corresponding caption. All things being equal, it is better to construct your own figures than to copy and paste those of others. In any case, always make sure that your images are readable, do not suffer from pixelation or aliasing effects, and that each is clearly numbered, captioned and meaningfully referenced in the main body of the text.

Ensure that there is a cohesive argument expressed in the text of the report and that it is not simply a bag of diagrams, screenshots and wishful thinking. Every report should tell a story, so know what story you want to tell. When you include images, make sure they are readable and truly add to the discussion.

Make sure your language is professional throughout, and steer a course between pompous and colloquial. Maintain authorial distance and do not overuse "me," "I" and "our." Your are writing for a professional audience who will judge you on the quality of your prose, so use a grammar and a spelling checker.

Use LaTeX if you wish – this is recommended if you plan to use mathematical formulae in your report, but in any case, keep the general spacing and font/style you find here (Single or 1.5 spacing, 12 pt. font for text, etc.). Be sure to submit a PDF (never a .DOC or .DOCX file) as your report. If you prepare your report in MS Word, as this document has been, save it as a PDF before you submit it. Overall it should be about 18 – 20 pages, including figures, front matter and references, A significant portion of the report will be textual, with approx.. five or six thousand words. Do not rely on images or other filler to write your report for you. The dates and means of submission will be communicated to you separately.

Chapter 8: Latex Pointers

This chapter contains some examples on the usage of latex. Do not include in your final report.

8.1 Figures

From time to time, it's necessary to add pictures to your documents. Using LaTeX all pictures will be indexed automatically and tagged with successive numbers when using the figure environment and the graphicx package. We can reference the figure below using its label like this: Fig. 8.1.

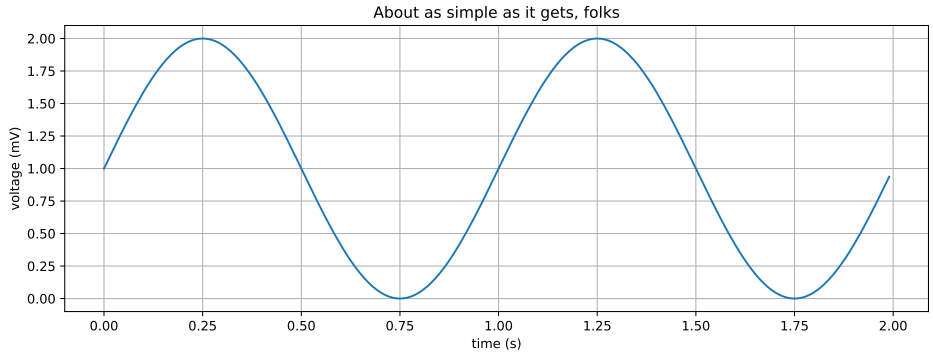

Figure 8.1: A sample graph

Lots more information in this tutorial.

8.2 Code Listing

You can list code using the *listings* package:

Listing 8.1: Borg Pattern

```
class Borg(object):
    __shared_state = {}

    def __init__(self):
        self.__dict__ = self.__shared_state
        self.state = 'Init'

    def __str__(self):
        return self.state
```

Lots more examples here.

8.3 Math

Here 8.1 is an example of including an equation

$$\lim_{x \to \infty} f(x) \tag{8.1}$$

More examples here and here.

8.4 References

Look up the bibtex references on google scholar or import from Mendeley or other reference managers. Add the bibtex snippet to the *references.bib* file. Then cite the reference like this:

As explained in [3], we also find that...

Lots more details here.

Acknowledgements

In your Acknowledgements section, give credit to all the people who helped you in your project.

Bibliography

1. Menger, K. Das botenproblem. *Ergebnisse eines Mathematischen Kolloquiums 2,* 11–12 (1932).
2. Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B. & Song, L. Learning Combinatorial Optimization Algorithms over Graphs. *CoRR* **abs/1704.01665.** arXiv: 1704.01665. http://arxiv.org/abs/1704.01665 (2017).
3. Knuth, D. E. *Art of computer programming, volume 2: Seminumerical algorithms* (Addison-Wesley Professional, 2014).