Morgan Saville

Exploring a Continuous Variant of the Travelling Salesman Problem

Computer Science Tripos – Part II

King's College

November 16, 2022

Declaration of Originality

I, Morgan Saville of King's College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose. I am content for my dissertation to be made available to the students and staff of the University.

Signed: `// TODO: Fill this in`

Date: November 16, 2022

Proforma

Name:	**Morgan Saville**
College:	**King's College**
Project Title:	**Exploring a Continuous Variant of the Travelling Salesman Problem**
Examination:	**Computer Science Tripos – Part II, July 2022**
Word Count:	**1290**
Project Originator:	**Morgan Saville**
Supervisor:	**Dr Timothy Griffin**

Original Aims of the Project

```
// TODO: Fill this in
```

Work Completed

```
// TODO: Fill this in
```

Special Difficulties

```
// TODO: Fill this in
```

Acknowledgements

// TODO: Fill this in

Contents

List of Figures

Chapter 1

Introduction

The Travelling Salesman Problem is infamous in the field of Complexity Theory. In plain English, the problem can be described as follows:

> Given a set of cities, find a tour which visits all of them, returning to the starting point, which minimises the distance travelled.

The first known instance of this problem being posed was in 1832 by an author cited only as "old clerk traveller"[3].

The problem was formalized by mathematician Karl Menger in the 1930s[1]. Translated into English[2], Menger described the problem in the following way.

> We denote by messenger problem (since in practice this question should be solved by each postman, anyway also by many travelers) the task to find, for finitely many points whose pairwise distances are known, the shortest route connecting the points. Of course, this problem is solvable by finitely many trials. Rules which would push the number of trials below the number of permutations of the given points, are not known. The rule that one first should go from the starting point to the closest point, then to the point closest to this, etc., in general does not yield the shortest route.

To this day, Menger remains correct in his claim that no known algorithm exists to find the optimal solution, faster than trying every possible tour. Each tour corresponds to a certain permutation of the cities, so if there are n cities, there are $n!$ possible tours, and so the time complexity of this brute-force algorithm is at best $\mathcal{O}(n!)$.

1.1 A Definition of the Discrete Travelling Salesman Problem

Many variations of the Travelling Salesman Problem exist. For the purposes of this dissertation, and to avoid ambiguity, we will define the problem rigorously in this section

Given:

A finite set V of n vertices for some $n \in \mathbb{N}$

A total function $C : V^2 \to \mathbb{R}$
such that $\forall v \in V.\ C(v, v) = 0$
and $\forall u, v \in V.\ C(u, v) = C(v, u)$

Find:

A permutation (allowing repetition) T of V
with $m := |T|$
and T_i denotes the i^{th} element of T for any $0 \leq i \in \mathbb{N} < m$
and $\hat{C} := C(T_{m-1}, T_0) + \sum_{i=0}^{m-2} C(T_i, T_{i+1})$
such that $\forall v \in V.\ v \in T$
and $\hat{C}$ is minimised

We will refer to this problem as the Discrete Travelling Salesman Problem (DTSP).

An instance of the DTSP can be parameterised by (V, C), and the solution is T. If T really does minimise $\hat{C}$, then we call this solution "optimal". However, it is still useful to consider suboptimal tours. Tours form a total order, with $T \leq T'$ meaning that the value of $\hat{C}$ for tour T is less than or equal to that of T'. We will interchangeably refer to this relation as T being "at least as good as" T'. In this total order, the optimal solution is the least element. In future, when we refer to *the* solution of a DTSP instance, this refers to any of the optimal solutions. When instead we refer to *a* solution with the aim of comparing their "goodness", this can include sub-optimal solutions.

V corresponds to the set of cities, and $C(u, v)$ represents the distance between city u and city v. T represents the tour through the cities, starting at T_0, and finally returning from T_{m-1} to T_0. (Note that this forms a cycle — the starting point doesn't really matter. The cycle can be shifted so that T_0 is any starting point you want, and the value of $\hat{C}$ will remain unchanged)

$\hat{C}$ corresponds to the total distance travelled — the sum of the distances between adjacent cities in T, plus the distance to return home from T_{m-1} to T_0.

We have stated that T is allowed repetitions. This is to say that some cities can be visited more than once. Intuition suggests that any optimal tour will include no repetitions. However, that assumption relies on the distance metric C obeying the triangle inequality, which we do not require.

The triangle inequality can be described with the following equation:

$$\forall u, v, w \in V.\, C(u, w) \leq C(u, v) + C(v, w)$$

Intuitionally, it will never be more costly to go from u to w directly, rather than go via v. We do not require that our distance metric obeys this inequality because this allows the DTSP to be a closer allegory to the CTSP which we will define in the next section.

```
// TODO: Illustration
```

1.2 The Continuous Travelling Salesman Problem

Suppose you are a hiker, hiking (as one often does) in a mountainous region. You have several locations which you want to visit, and then you want to return home. Furthermore, you want to do so while climbing the least distance vertically.

```
// TODO: Illustration of height map
```

This problem could be represented by an instance of the DTSP. V would be the set of locations you want to visit, including your starting point, and (assuming we count all vertical movement as climbing, whether up or down) $C(v, w)$ would be the minimum amount of climbing required to get from v to w. The solution T would then be the route you want to take.

Unfortunately, one of the locations is at the top of a very steep and very tall mountain, and you would be satisfied to get *near* it instead of reaching it exactly, if that means you can avoid a lot of climbing. The DTSP is no longer equipped to help you solve this. The inherent discreteness precludes the notion of being "near" a city.

To combat this, we can generalise the DTSP. Instead of the cities being abstract discrete vertices, we will model them as being points in a continuous space. Likewise, we can no longer characterise the distance between two cities as a single number, as it depends on the path taken between them. We will call this generalised problem the Continuous Travelling Salesman Problem (CTSP) and define it as follows:

Given:

A continuous connected set D

A finite set $P \subset D$

A continuous total function $f : D \to \mathbb{R}$

A continuous total function $t : D^2 \to \mathbb{R}$
such that $\forall x \in D.\ t(x, x) = 0$
and $\forall x, y \in D.\ t(x, y) = t(y, x)$

Find:

A closed loop $L \subseteq D$
with $F := \int_L f(\mathbf{s})\ d\mathbf{s}$
and $T := \sum_{p \in P} \min_{l \in L} t(p, l)$
and $C' := F + T$
such that C' is minimised

An instance of the CTSP is parameterised by (D, P, f, t) and the solution is L. Much like the DTSP, if L truly does minimise C', then L is an optimal solution, but other non-optimal solutions exist, forming a total order.

D is the domain in which our points of interest reside. P is the set of points we want to visit, and is analogous to V in the definition of the DTSP. L gives us the route to reach (or get close to) all of the points in P, and then return home.

$f(x)$ is the cost density function. It represents the cost per unit distance of travelling through an arbitrarily small neighbourhood around x. Furthermore, if we have a path (or contour) through D, the total cost of travelling along the path can be calculated as the integral of f along that path. This is loosely analogous to C from the definition of the DTSP, except instead of telling us the cost of travelling between two points, f allows us to calculate the cost of moving along a path. This is the meaning behind F — the total cost of travelling along L.

Unlike with the DTSP however, solutions to the CTSP can accrue cost in more than one way. The first way (captured by F) is to move through high-cost-density areas. The other way is by not getting close enough to the points in P. This is the motivation behind defining the distance metric t. Suppose you have a point $p \in P$, but instead of reaching it, the closest you get it some other point $x \in D$, then $t(p, x)$ tells you the additional cost incurred. We will refer to this cost as a "near-miss tax".

To calculate the near-miss tax a solution L will incur from a point $p \in P$, we take the minimum distance (according to t) from p to any point on L.

T, therefore, is the sum of the near-miss tax for each of the points in P.

The total cost of the solution is C', the sum of F and T.

```
// TODO: Illustration
```

The remainder of this dissertation will explore the $CTSP$, and find algorithms which approximate the optimal solutions.

Chapter 2

Preparation

Chapter 3

Implementation

Chapter 4

Evaluation

Chapter 5

Conclusions

Bibliography

[1] Sebastian Lotz. “Lösung des Traveling Salesman Problems und Darstellung in einer Webapplikation”. In: (2014), p. 1.

[2] Alexander Schrijver. “On the history of combinatorial optimization (till 1960)”. In: *Handbooks in operations research and management science* 12 (2005), pp. 40–41.

[3] *The traveling salesman as he ought to be and what he has to do in order to get commissions and be sure of a happy success in his business*. 1832. URL: `https://zs.thulb.uni-jena.de/receive/jportal_jparticle_00248075?lang=en`.

Appendices

Index

Copy of Project Proposal

Introduction and Description

This project explores algorithms to approximate solutions to a continuous variant of the Travelling Salesman Problem.

In this variant, the cities are embedded in a continuous space (the domain). The domain must be connected (i.e., for any two points in the domain, there exists a path between them which is fully contained within the domain). There is a real-valued, continuous cost density function defined over this domain, and the cost of a given path (the terms *path* and *contour* will be used interchangeably) is defined by the integral of the cost density function along the contour. The goal is to find the contour inside the domain which gets as close as possible to every city, with the minimum cost.

In the discrete TSP, approximate solutions are Hamiltonian paths. The total cost of the approximation is solely based on the total cost of this path.

However, in the continuous variant, the total cost of an approximate solution can be measured by *two* statistics. One is the cost of the contour (as defined above). The second comes from the fact that the cities exist in a continuous domain. This means that a contour can get arbitrarily close to a city without passing through it, and so the total cost of the approximation can depend on how close the contour comes to each of the cities.

To quantify this, we define the total cost of an approximate solution to be equal to the cost of the contour, *plus* a "near-miss tax". This tax is the sum of the distances (according to a given distance metric) of closest approach between the contour and each of the cities.

An example is shown below

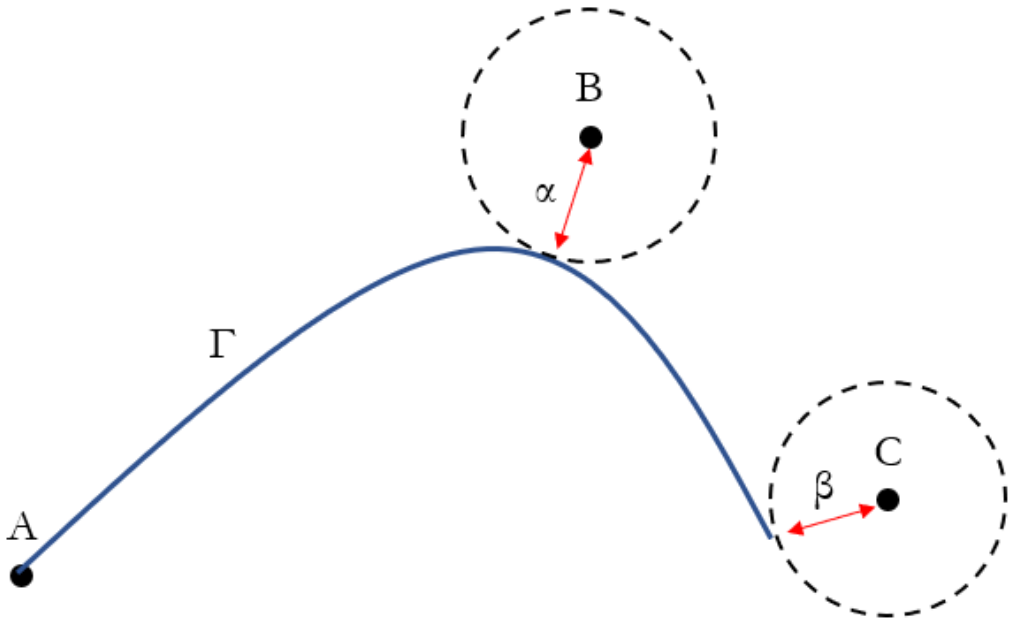

Figure 1: Near Miss

In this example, the cost of the shown solution would be the integral of the cost density function along Γ, plus α, plus β.

A precise definition of both the discrete and continuous variants of the TSP as they will be used in this project are given below:

Definition of the Discrete TSP

Given:

- A set of points S
- A cost total function $C : S \times S \to \mathbb{R}$ with the following property:
  - $\forall A \in S.\quad C(A, A) = 0$

Find:

- A route $Y \in S^n$ where $n \geq |S|$ with the following properties:
  - $\forall A \in S.\quad A \in Y$
  - $\sum_{i=0}^{n-2} C(Y_i, Y_{i+1})$ is minimised

An instance of the discrete TSP can be parameterised by (S, C) and the solution is Y.

Definition of the Continuous TSP

Given:

- A continuous connected domain D
- A finite set of points $S' \subseteq D$
- A continuous cost density total function $f : D \to \mathbb{R}$
- A distance metric $\Delta : D \times D \to \mathbb{R}$ satisfying the following properties:
  - $\forall A \in D.\quad \Delta(A, A) = 0$
  - $\forall A, B, C \in D.\quad \Delta(A, B) + \Delta(B, C) \geq \Delta(A, C)$ (the triangle inequality)

Find:

- A contour $Y' \subseteq D$ with the following properties:
  - $\forall A' \in S'.\quad A' \in Y'$
  - $\left(\int_{Y'} f(z)\, dz\right) + \sum_{A \in S} \min(\{\Delta(A, X) \mid \forall X \in Y'\})$ is minimised

An instance of the continuous TSP can be parameterised by (D, S', f, Δ) and the solution is Y'.

It should be noted that approximate solutions to the continuous TSP do not necessarily map onto approximate solutions to the discrete TSP. The goal of this project is not to use the continuous variant in order to better solve the discrete variant, but instead to explore the continuous TSP in its own right.

The main deliverable of this project is a collection of algorithms which approximate solutions to the continuous TSP. Some of these algorithms will be existing approximations to the discrete TSP but generalised to the continuous variant (such as Ant Colony Optimisation (Brezina Jr and Čičková 2011) and the Slime Mold Algorithm (Liu et al. 2020)). Some research has been done into similarly-defined continuous variant of the TSP (Andrews and Sethian 2007), but an extension to this project will be for at least one of the delivered implementations to be a novel algorithm specifically tailored to the continuous case. For the novel algorithm/s, the dissertation will include the detailed mathematical and logical reasoning behind them.

These algorithms will be evaluated on the following metrics:

- The relative cost of the approximate solutions found by the algorithm
- The time complexity of the algorithm
- The space complexity of the algorithm
- The parallelisability of the algorithm

This project will also contain a proof that an instance of the discrete TSP can be transformed into an instance of the continuous TSP, and the solution of the latter can be transformed back into the solution of the former.

A result of this fact is that the continuous TSP is at least as hard as the discrete TSP. As such, the solutions found by the algorithms will not be comparable to some "ground truth" perfect solution, so they will instead be compared to one another.

The algorithms will be implemented in Python. The cost function and distance metric inputs to the algorithm will be represented using either SymPy (Meurer et al. 2017) or TensorFlow (Abadi et al. 2015) variables (or something similar) as they use symbols to represent mathematical relationships between variables. This allows the cost function and distance metric to be represented as pure mathematical functions. These symbolic maths libraries also allow for integrals and derivatives to be performed on variables with respect to other variables, and for differential equations to be solved. An example using SymPy is shown below, but TensorFlow allows us to do similar things.

```
from sympy import symbols, integrate

# suppose the cost function is defined on R^2
x1, x2 = symbols("x1 x2")

# the cost function is the square distance from the origin
y = x1 * x1 + x2 * x2

# if we wanted to e.g. integrate y with respect to x1 as x1 goes from -2 to 2:
print(integrate(y, (x1, -2, 2)))
# prints: "4*x2**2 + 16/3"
```

The domain will either be defined in a similar way, or will not need to be explicitly defined at all. (Note how in the above example, the domain is implicitly defined as the set of all pairs of values for `x1` and `x2`)

Starting Point

There is no existing code or materials forming the basis of this project. This is a project I first started thinking about approximately a year ago, but I have made no serious attempts at solving it.

Success Criterion

This project will be a success if I am able to deliver at least three implementations of algorithms to approximate solutions to the continuous TSP. The implementations need to work when all of the following conditions are met:

- The domain is $\mathbb{R}^2$
- The distance metric is Euclidean
- The cost function is differentiable

Work Plan

Work Package	Description	Deliverables	Deadline
0	Research algorithms to approximate discrete TSP	Have a list of existing TSP algorithms which can be generalised to the continuous variant (a generalised algorithm)	23/10/22
1	Research algorithms to approximate continuous variants of discrete problems	Be able to describe a rough outline of at least one algorithm which approximates a solution to the continuous variant (a direct algorithm)	06/11/22
2	Continue research and start implementation of generalised algorithms	Have at least one generalised algorithm implemented and working	20/11/22

Work Package	Description	Deliverables	Deadline
3	Continue research and, continue implementation of generalized algorithms, start implementation of novel algorithms	Have every generalised algorithm from the list from Work Package 0 implemented and working	04/12/22
4	Write progress report, continue implementation of novel algorithms	Have at least one direct algorithm implemented and working, and submit progress report	18/12/22
5	Optimise novel algorithm	Have at least one direct algorithm implemented efficiently	08/01/23
6	Perform evaluation of algorithms	Produce a written comparison of all implemented algorithms containing numerical/statistical analysis as well as a plain english interpretation	22/01/23
7	Write dissertation	Complete draft of chapters 1, 2, and 3	05/02/23
8	Write dissertation	Complete full draft	19/02/23
9	Write dissertation	Dissertation ready to submit	05/03/22

Special Resources

Testing of the suite of algorithms will likely require the use of Google CoLab (Bisong 2019) or other similar cloud computing services. Fortunately, I already have a subscription to Google CoLab Pro for personal use, so I do not need the university to provide one.

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." https://www.tensorflow.org/.

Andrews, June, and JA Sethian. 2007. "Fast Marching Methods for the Continuous Traveling Salesman Problem." *Proceedings of the National Academy of Sciences* 104 (4): 1118–23.

Bisong, Ekaba. 2019. "Google Colaboratory." In *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, 59–64. Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-4470-8_7.

Brezina Jr, Ivan, and Zuzana Čičková. 2011. "Solving the Travelling Salesman Problem Using the Ant Colony Optimization." *Management Information Systems* 6 (4): 10–14.

Liu, Meijiao, Yanhui Li, Ang Li, Qi Huo, Ning Zhang, Nan Qu, Mingchao Zhu, and Liheng Chen. 2020. "A Slime Mold-Ant Colony Fusion Algorithm for Solving Traveling Salesman Problem." *IEEE Access* 8 (January): 202508–21. https://doi.org/10.1109/ACCESS.2020.3035584.

Meurer, Aaron, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, et al. 2017. "SymPy: Symbolic Computing in Python." *PeerJ Computer Science* 3 (January): e103. https://doi.org/10.7717/peerj-cs.103.