Phase 2 Full Proposal Draft — Exploring a Continuous Variant of the Travelling Salesman Problem

Morgan Saville, jbs52

Supervisor: Timothy Griffin

This project relates to two problems: The discrete (classic) Travelling Salesman Problem, and a continuous variant of the Travelling Salesman Problem. These are defined below:

Definition of the Discrete TSP

Given:

- A set of points S
- A cost total function $C \subset S \times S \times \mathbb{R}$ 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 set of points $S' \subseteq D$
- A continuous cost density total function $f \subset D \times \mathbb{R}$

Find:

- A contour $Y' \subseteq D$ with the following properties:
  - $\forall A' \in S'. \quad A' \in Y'$
  - $\int_{Y'} f(z)\, dz$ is minimised

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

Project Description

This project explores algorithms to approximate solutions to the continuous TSP as defined above. 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 and the Slime Mold Algorithm). At least one will 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 paths 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. The reverse mapping from continuous to discrete will also be proved.

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 input to the algorithm will be represented using either SymPy or TensorFlow variables (or something similar) as they use symbols to represent mathematical relationships between variables. This allows the cost function to be represented as a pure mathematical function of the position vector of its input. These symbolic maths libraries also allow for integrals and derivates to be performed on variables with respect to other variables, and for differential equations to be solved. The domain input will be implicitly defined as the set of points for which the cost function is defined.

Timetable

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	23/10/22
1	Research algorithms to approximate continuous variants of discrete problems	Be able to describe a rough outline of at least one novel algorithm which approximates a solution to the continuous variant	06/11/22
2	Continue research and start implementation of generalised algorithms	Have at least one generalised algorithm implemented and working	20/11/22
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 novel algorithm implemented and working, and submit progress report	18/12/22
5	Optimise novel algorithm	Have at least one novel algorithm implemented efficiently	08/01/23

Work Package	Description	Deliverables	Deadline
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 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.