2022

Advocating the value of Biologically Inspired Hybridization Methodologies using the Traveling Salesman Problem

Elihu Essien-Thompson (C14460702)

Advanced Software Development - Part-Time (ASD/TU060)

Extra Details

Number of required modules left – None

Pending Results: Advanced Databases, Research Design & Proposal Writing, and Universal Design

Intended dissertation start date – January 2022

Proposal Submission attempt – 1st

Dissertation title - *Advocating the value of Biologically Inspired Hybridization Methodologies using the Traveling Salesman Problem*

People Associated – None

Sources of Data – None

1. The Research Background

In the past, as part of the design process, physical prototypes had to be designed and constructed in order to perform experiments. However, the development of computers over the years has stimulated engineers to first simulate models using computers in order to investigate different aspects of real systems, reducing both the time and cost required for innovations (Mirjalili, 2019c). Since then, many optimization algorithms have been designed, utilizing the high level of processing power provided by modern computers, allowing the computer itself to find optimal solutions to the simulated models of given problems.

Biologically Inspired Algorithms are a term used to denote algorithms that arise from an algorithm engineer's understanding, gained through analysis of nature's solution to common problems. These mostly heuristic-based algorithms operate by balancing between two conflicting techniques: *exploration* (discovering new solutions) and *exploitation* (narrowing down on the best solution) (Thangaraj et al., 2011). They fall under subcategories like Evolutionary algorithms (founded on Darwin's theory of Evolution; using the concept of gene-crossovers), Swarm Intelligence (modeled after the behaviors of creatures that operate in swarms; using a team of multiple simplistic agents working together), and Neural Networks (gaining inspiration from the workings of the brain; using a model neural structure) among many others.

With regards to combinatorial optimization problems, one of the most important issues is the traveling salesman problem (TSP) (Yousefikhoshbakht, 2021). In the problem, there is a designated group of cities to visit distributed on a map and the salesman, starting from any one city, is required to make the shortest round trip by visiting all designated cities one time each. The TSP has been one of the tools used to benchmark the performance of and demonstrate the merits that the use of Bio-Inspired Optimization Algorithms (BIOA) can bring (Chen & Chien, 2011).

2. Problem Description

Each BIOA has been created to address a naturally occurring problem like the shortest path or optimum location finding. As such, each of these targeted algorithms comes with specific advantages and disadvantages (V.Selvi & Dr.R.Umarani, 2010). An important finding is that it is possible to increase performance gains by combining the strengths of various BIOAs through hybridization. A study done by Huang et al.

(2013), which aimed to combine the Ant Colony Optimization (ACO) and the Particle Swarm Optimization (PSO) algorithms using 4 different hybridization strategies, also proved that even more benefits could be drawn if proper care was given to the methodology behind hybridization. Based on the results of this paper, it can then logically be assumed that there are multiple versions of other BIOA combinations, each yielding different performance results.

Unfortunately, there is very little research being done on hybridization methodologies for BIOAs to isolate heuristics or principal guidelines to use when combining them. This might be because not enough light is being shone on the value of these hybrid algorithms. So, my dissertation aims to advocate the value of hybridization in hopes to stimulate interest and promoting further research into hybridization methodologies.

Due to monetary limitations over quarantine, I have chosen to carry out this experiment on a laptop borrowed from the college, having an Intel® Core™ i5-10210U CPU @1.60GHz 2.11GHz processor, a 16BG ram capacity, and a 64-bit Operating System. I have also chosen not to directly tackle hybridization methodologies due to the limitation in the time provided for and the difficulty level of my study. The paper mentioned earlier was a study undertaken by 5 people, so to attempt to replicate the appropriate procedures required to continue their study by myself in a master's program is not feasible.

3. Literature Review, Gaps, and State-Of-The-Art Solutions

The book '*Evolutionary Algorithms and Neural Networks: Theory and Applications'* authored by Seyedali Mirjalili (2019c) gave an overview of the inner workings of various evolutionary and swarm-based BIOAs. Algorithms mentioned in the book that was found relevant for my project included the Ant Colony Optimization (ACO) Algorithm, the Particle Swarm Optimization (PSO) Algorithm, and the Genetic Algorithm (GA) (Mirjalili, 2019a, 2019b, 2019d). The book '*Swarm Intelligence in Optimization'* by Christian Blum & Xiaodong Li (2008) adds more in-depth descriptions of "two of the most successful examples of optimization techniques inspired by swarm intelligence"; the ACO and PSO algorithms. Because BIOAs are mostly heuristic-based algorithms, they find great application in systems that require a heuristic-based

approach like fuzzy logic controllers (Castillo et al., 2012; Martínez-Soto et al., 2014). A study performed by Dorigo et al. (2006) notes that there are 3 main types of ACO algorithms in use today and they highlight some of the key areas where this algorithm has been applied. Using the terms "self-confidence" and "swarm confidence", Das et al. (2008) gave very easily understood explanations to the weights and techniques used in PSO. Poli et al. (2007) went in-depth in their analysis of the workings of the PSO algorithm, highlighting optimum values to tune its components and weights in order to get the best performance from the algorithm. Bai (2010) notes that the PSO algorithm is a very easily completed and simple heuristic optimization method based on swarm intelligence when compared with other algorithms. Mukund Nilakantan et al. (2015) proposed a method to highlight the best stop condition or 'maximum number of iterations' for the PSO algorithm, which could possibly be generalized to other algorithms. Whitley (1994) gave a report on the technical workings of the Genetic Algorithm with its variations and considerations. He remarks on how simple changes in the algorithm can give rise to surprising kinds of emergent behavior. It is possible to create an optimum solution that performs in an unanticipated manner. Selvi & Umarani (2010) labeled ACO as best for discrete optimization problems while PSO is more towards multi-objective, constrained optimization problems or ones having dynamically changing domains. Rohini & Natarajan (2016) found that out of the 3, the GA is the most appropriate to be used for scheduling systems.

Martinez-Estudillo et al. (2006) found that, though the computational cost is slightly higher, the differences in accuracy/generalization performance were significantly better for their hybrid clustering model over the base versions. While creating an ACO/PSO hybrid algorithm, Mandloi & Bhatia (2016) also demonstrated that hybrid algorithms can be designed with low complexity but still output higher results than the base algorithms. In a study done by Khourdifi & Bahaj (2019) the results highlighted a 15.5% increase in best precision average value before joint application of the PSO and ACO algorithms (from 84.2% to 99.7%). Shelokar et al. (2007) found that they were able to extend the application of the ACO algorithm from discrete to continuous problems by combining it with the PSO algorithm. They noted better performance results with the hybrid model as the problem dimensions increased.

Yang & Yoo (2018) found that the best performance for the GA/ACO hybrid algorithm was obtained at 0.5 for the crossover probability and 0.08 for the mutation probability. Luan et al. (2019) also used a GA/ACO hybrid algorithm and were able to determine the best time to switch between them by analyzing the speed curve of the two algorithms.

Moradi & Abedini (2012) found that the hybrid combination of the GA/PSO algorithm offered the best solutions when compared to their base algorithms, however, the base PSO algorithm was noted to have the fastest running time of the group. Thangaraj et al. (2011), in their analysis of GA/PSO hybrid as well as other hybrid PSO models, isolated 3 main methods through which hybrid algorithms can be created and showed that 34% of all studies done on a PSO hybrid since 2001-2010 have been done using the GA/PSO hybrid algorithm. Omidinasab & Goodarzimehr (2019) also found that the hybrid GA/PSO algorithm, when compared to the other chosen meta-heuristic algorithms for convergence, required fewer iterations for convergence. Soleimani & Kannan (2015) performed a study that supported the superiority of the GA/PSO model to its base versions and also showed that duplicating the population size for the GA does not guarantee better performance.

The TSP has been said to be easy to describe but difficult to solve (Hoffman & Padberg, 2001). While ACO was developed directly to tackle this problem, GA has also been used to accomplish this (Braun, 1991; Moon et al., 2002) along with many others. Wang et al. (2003) adapted the original PSO algorithm to allow the capability to tackle the TSP. Their algorithm performs using swap sequences to dictate the order of cities to visit. Majid Yousefikhoshbakht (2021) proposed a new Modified PSO algorithm named MPSO, developed to tackle the TSP while avoiding the PSO's inherent tendency to get stuck on local rather than global optimums. Hybrid solutions have been used to solve the TSP as well, like in the study done by Dong et al. (2012) with a hybrid GA/ACO algorithm called a cooperative genetic ant system, or the one done by Mahi et al. (2015) using PSO, ACO, and 3-Opt. Chen & Chien (2011) performed an experiment to solve the TSP using an algorithm that they call genetic simulated annealing ant colony system with particle swarm optimization techniques (a hybrid created from 4 base algorithms) outperforming the base models.

Gaps: Variations in the design of each hybrid pairing have been noted in the body of work supporting my earlier claim. Unfortunately, many studies done only consider a single hybridization method and show in-consideration to other variants.

State of the art solutions: The study done by Huang et al. (2013) was the only paper of its kind that I have found. The speed curve used by Luan et al. (2019) also proposed a valid heuristic to solve this problem.

4. Research Question

"When an independent T-Test is performed on the results between the Hybridized and Base versions, which of the chosen bio-inspired optimization algorithm is statistically the fastest to converge on the optimal solution to the Traveling Salesman Problem, given standardized population size, number of maximum iterations, and a statistical significance threshold of 0.1?"

5. Hypothesis

From my research, beginning with the ACO, I have found that the GA, PSO, 3-Opt, and Simulated Annealing algorithms have been used to solve the TSP in the past. For this experiment, I have decided to narrow down on the GA, ACO, and PSO algorithms because of how often they have been used together in the body of work done. I intend to build 3 hybrid algorithms based on the 3 base versions (ACO/GA, PSO/GA, and ACO/PSO) to use for comparison.

Because this experiment is based on the TSP, the first assumption that comes to mind would be that the ACO algorithm is all that is needed. It was intrinsically designed to tackle this problem and, as such, it is usually the default solution. So, my null hypothesis, in this case, would be: *If I perform an independent T-Test on the results between the chosen Hybridized and Base algorithms, then the base ACO algorithm will converge on the optimum solution for the TSP statistically faster than any other algorithm, given standardized population size, number of maximum iterations and a statistical significance threshold of 0.1, because it was intrinsically developed to solve "shortest path" optimization problems*

The goal of this experiment is to show the value that hybridization can bring. So, the alternate hypothesis would be: *If I perform an independent T-Test on the results between the chosen Hybridized and Base algorithms, then all hybrid optimization algorithms (ACO/PSO, ACO/GA, PSO/GA) will converge on the optimum solution for the TSP statistically faster than any of the base algorithms (ACO, PSO, GA) AND the ACO/GA algorithm will be the fastest, given standardized population size, number of maximum iterations and a statistical significance threshold of 0.1.*

I assume ACO/GA to be the fastest because of the perceived complementary nature of the two algorithms.

6. Research Objectives and Experimental Activities

O1 – Prepare algorithms and tools required for the experiments

O2 – Code an automatic random generator and display system for the traveling salesman problem, having a pre-definable number of cities to visit (vertices) randomly distributed on a map (list of vertices) using a simple visual tool like Processing 3D

I. Generate a few random maps with varying numbers of cities to visit (10, 20, and 50) to use for algorithm testing (e.g. 9 maps, 3 for each city count)
II. Store generated maps persistently in a text file

O3 – Code and test the base algorithms to use for the experiments using Python

I. Load in the generated maps from the text file
II. Code, Debug, and Error check the ACO algorithm and test it on the generated maps. Record and display the:
    - The final solution converged on (*shortest path*), and its score ($1/distance$)
    - Iteration number the solution was found on (*F-number*)
    - History of best solutions found since the first iteration (*list of scores*)
III. Plot best solution for display
IV. Repeat for the PSO and GA

O4 – Code and test the Hybrid Algorithms to use for the experiments using Python

I. Using the code developed from objective 2, blend the ACO and PSO algorithms to create the first hybrid ACO/PSO algorithm and run it on the generated maps. Again, record and display the:
    - The final solution converged on (*shortest path*), and its score ($1/distance$)
    - Iteration number the solution was found on (*F-number*)
    - History of best solutions found since the first iteration (*list of scores*)
II. Plot best solution for display
III. Repeat for ACO/GA and PSO/GA

O5 – Set standardized parameters for the experiments

O6 – Analyse the results from objective 1 to get a vague understanding of performance speeds and iterations. Using this data drawn from research, assign cut-off parameters that the experiment will be done under for each algorithm and number of cities.

O7 – Test the hypothesis

O8 – Experiment: Test the hypothesis for 500 maps with a TSP size of 10 in Python

I. Using the tools and settings established in objectives 1 and 5, Generate 500 graphs, each having 10 cities, and run all 6 algorithms across them generating the required data.
II. Store data persistently as CSV

09 – Examine the results using either Python or R. (The critical data to examine for a conclusion is the F-number, the others are just for extra information).

I. Plot the *F-number* using histograms with interval ranges and the *list of scores* as a linear graph of average score vs iteration number per algorithm.
II. Plot standard deviation on the *F-number* for each Algorithm
III. Perform an ANOVA test on the *F-number*.
IV. Remove any outliers on the *F-number* using the standard deviation, ANOVA, and other plots.
V. Perform either an Independent T-test or Mann-Whitney U Test on the *F-number*, depending on the results from steps 2-4, for each algorithm, to draw a conclusion from.

O10 - Repeat for 500 maps of 20 cities, and again for 500 maps of 50 cities.

O11 – Analyse all results drawn from the experiment and state the concluding stance on the hypothesis.

7. Evaluation of Designed Solution with Statistical tests

The *score* variable is used by the algorithms themselves to narrow down on the optimum, while the *F-number* will be used by the researcher (me) to find the best algorithm(s). The goal of my research question is to find which algorithm is the fastest to converge on the best solution. I will be measuring speed based on the number of iterations needed before the best solution is found (*F-number*). So, the lower the *F-number*, the better the algorithm.

For the sake of creating a fair experiment, this test will be run multiple times (as shown by the objective list), generating a list of *F-numbers* and, after processing, performing the Independent T-test or Mann-Whitney U Test on the *F-number*, depending on the nature of the dataset, will ultimately show which algorithm(s) succeeds with the lowest *F-numbers* when compared with the rest.

I am expecting to see very close results between some of the algorithms, so I have chosen to set the statistical significance at 0.1 for all statistical tests. A successful outcome is if all hybrid algorithms are each shown to have a lower *F-number* statistically than all of the chosen base algorithms given the threshold. Along with this, I will also be looking for the ACO/GA algorithm to come out the best overall. If these outcomes are not met after the experiment, I will reject my hypothesis.

8. Bibliography

Bai, Q. (2010). Analysis of Particle Swarm Optimization Algorithm. *Computer and Information Science*, *3*(1), p180. https://doi.org/10.5539/cis.v3n1p180

Blum, C., & Li, X. (2008). Swarm Intelligence in Optimization. In C. Blum & D. Merkle (Eds.), *Swarm Intelligence: Introduction and Applications* (pp. 43–85). Springer. https://doi.org/10.1007/978-3-540-74089-6_2

Braun, H. (1991). On solving travelling salesman problems by genetic algorithms. In H.-P. Schwefel & R. Männer (Eds.), *Parallel Problem Solving from Nature* (pp. 129–133). Springer. https://doi.org/10.1007/BFb0029743

Castillo, O., Martínez-Marroquín, R., Melin, P., Valdez, F., & Soria, J. (2012). Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile

robot. *Information Sciences*, *192*, 19–38. https://doi.org/10.1016/j.ins.2010.02.022

Chen, S.-M., & Chien, C.-Y. (2011). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, *38*(12), 14439–14450. https://doi.org/10.1016/j.eswa.2011.04.163

Das, S., Abraham, A., & Konar, A. (2008). Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. In Y. Liu, A. Sun, H. T. Loh, W. F. Lu, & E.-P. Lim (Eds.), *Advances of Computational Intelligence in Industrial Systems* (pp. 1–38). Springer. https://doi.org/10.1007/978-3-540-78297-1_1

Dong, G., Guo, W. W., & Tickle, K. (2012). Solving the traveling salesman problem using cooperative genetic ant systems. *Expert Systems with Applications*, *39*(5), 5006–5011. https://doi.org/10.1016/j.eswa.2011.10.012

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. https://doi.org/10.1109/MCI.2006.329691

Hoffman, K. L., & Padberg, M. (2001). Traveling Salesman Problem (TSP)Traveling salesman problem. In S. I. Gass & C. M. Harris (Eds.), *Encyclopedia of Operations Research and Management Science* (pp. 849–853). Springer US. https://doi.org/10.1007/1-4020-0611-X_1068

Huang, C.-L., Huang, W.-C., Chang, H.-Y., Yeh, Y.-C., & Tsai, C.-Y. (2013). Hybridization strategies for continuous ant colony optimization and particle swarm optimization applied to data clustering. *Applied Soft Computing*, *13*(9), 3864–3872. https://doi.org/10.1016/j.asoc.2013.05.003

Khourdifi, Y., & Bahaj, M. (2019). Heart Disease Prediction and Classification Using Machine Learning Algorithms Optimized by Particle Swarm Optimization and Ant Colony Optimization. *International Journal of Intelligent Engineering and Systems*, *12*. https://doi.org/10.22266/ijies2019.0228.24

Luan, J., Yao, Z., Zhao, F., & Song, X. (2019). A novel method to solve supplier selection

problem: Hybrid algorithm of genetic algorithm and ant colony optimization. *Mathematics and Computers in Simulation*, *156*, 294–309. https://doi.org/10.1016/j.matcom.2018.08.011

Mahi, M., Baykan, Ö. K., & Kodaz, H. (2015). A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem. *Applied Soft Computing*, *30*, 484–490. https://doi.org/10.1016/j.asoc.2015.01.068

Mandloi, M., & Bhatia, V. (2016). A low-complexity hybrid algorithm based on particle swarm and ant colony optimization for large-MIMO detection. *Expert Systems with Applications*, *50*, 66–74. https://doi.org/10.1016/j.eswa.2015.12.008

Martinez-Estudillo, A. C., Hervas-Martinez, C., Martinez-Estudillo, F. J., & Garcia-Pedrajas, N. (2006). Hybridization of evolutionary algorithms and local search by means of a clustering method. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *36*(3), 534–545. https://doi.org/10.1109/TSMCB.2005.860138

Martínez-Soto, R., Castillo, O., & Aguilar, L. T. (2014). Type-1 and Type-2 fuzzy logic controller design using a Hybrid PSO–GA optimization method. *Information Sciences*, *285*, 35–49. https://doi.org/10.1016/j.ins.2014.07.012

Mirjalili, S. (2019a). Ant Colony Optimisation. In S. Mirjalili (Ed.), *Evolutionary Algorithms and Neural Networks: Theory and Applications* (pp. 33–42). Springer International Publishing. https://doi.org/10.1007/978-3-319-93025-1_3

Mirjalili, S. (2019b). Genetic Algorithm. In S. Mirjalili (Ed.), *Evolutionary Algorithms and Neural Networks: Theory and Applications* (pp. 43–55). Springer International Publishing. https://doi.org/10.1007/978-3-319-93025-1_4

Mirjalili, S. (2019c). Introduction to Evolutionary Single-Objective Optimisation. In S. Mirjalili (Ed.), *Evolutionary Algorithms and Neural Networks: Theory and Applications* (pp. 3–14). Springer International Publishing. https://doi.org/10.1007/978-3-319-93025-1_1

Mirjalili, S. (2019d). Particle Swarm Optimisation. In S. Mirjalili (Ed.), *Evolutionary Algorithms*

and Neural Networks: Theory and Applications (pp. 15–31). Springer International Publishing. https://doi.org/10.1007/978-3-319-93025-1_2

Moon, C., Kim, J., Choi, G., & Seo, Y. (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research*, *140*(3), 606–617. https://doi.org/10.1016/S0377-2217(01)00227-2

Moradi, M. H., & Abedini, M. (2012). A combination of genetic algorithm and particle swarm optimization for optimal DG location and sizing in distribution systems. *International Journal of Electrical Power & Energy Systems*, *34*(1), 66–74. https://doi.org/10.1016/j.ijepes.2011.08.023

Mukund Nilakantan, J., Ponnambalam, S. G., Jawahar, N., & Kanagaraj, G. (2015). Bio-inspired search algorithms to solve robotic assembly line balancing problems. *Neural Computing and Applications*, *26*(6), 1379–1393. https://doi.org/10.1007/s00521-014-1811-x

Omidinasab, F., & Goodarzimehr, V. (2019). A Hybrid Particle Swarm Optimization and Genetic Algorithm for Truss Structures with Discrete Variables. *Journal of Applied and Computational Mechanics*, *Online First*. https://doi.org/10.22055/jacm.2019.28992.1531

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, *1*(1), 33–57. https://doi.org/10.1007/s11721-007-0002-0

Rohini, V., & Natarajan, A. M. (2016). Comparison of Genetic Algorithm with Particle Swarm Optimisation, Ant Colony Optimisation and Tabu Search based on University Course Scheduling System. *Indian Journal of Science and Technology*, *9*(21). https://doi.org/10.17485/ijst/2016/v9i21/85379

Shelokar, P. S., Siarry, P., Jayaraman, V. K., & Kulkarni, B. D. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, *188*(1), 129–142. https://doi.org/10.1016/j.amc.2006.09.098

Soleimani, H., & Kannan, G. (2015). A hybrid particle swarm optimization and genetic algorithm

for closed-loop supply chain network design in large-scale networks. *Applied Mathematical Modelling*, *39*(14), 3990–4012. https://doi.org/10.1016/j.apm.2014.12.016

Thangaraj, R., Pant, M., Abraham, A., & Bouvry, P. (2011). Particle swarm optimization: Hybridization perspectives and experimental illustrations. *Applied Mathematics and Computation*, *217*(12), 5208–5226. https://doi.org/10.1016/j.amc.2010.12.053

V.Selvi & Dr.R.Umarani. (2010). Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications*, *5*. https://doi.org/10.5120/908-1286

Wang, K.-P., Huang, L., Zhou, C.-G., & Pang, W. (2003). Particle swarm optimization for traveling salesman problem. *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, *3*, 1583-1585 Vol.3. https://doi.org/10.1109/ICMLC.2003.1259748

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, *4*(2), 65–85. https://doi.org/10.1007/BF00175354

Yang, Q., & Yoo, S.-J. (2018). Optimal UAV Path Planning: Sensing Data Acquisition Over IoT Sensor Networks Using Multi-Objective Bio-Inspired Algorithms. *IEEE Access*, *6*, 13671–13684. https://doi.org/10.1109/ACCESS.2018.2812896

Yousefikhoshbakht, M. (2021). Solving the Traveling Salesman Problem: A Modified Metaheuristic Algorithm. *Complexity*, *2021*, e6668345. https://doi.org/10.1155/2021/6668345

9. Activities

O1 – Prepare algorithms and tools required for the experiments (~9 weeks)

O2 – Code an automatic random generator and display system for the traveling salesman problem (1-2 days)

O3 – Code and test the base algorithms to use for the experiments using Python (~6 weeks -> 2 weeks each)

O4 – Code and test the Hybrid Algorithms to use for the experiments using Python (~3 weeks -> 1 week each)

O5 – Set standardized parameters for the experiments (~1 week)

O6 – Analyse the results from objective 1 and assign cut-off parameters that the experiment will be done under (~1 week)

O7 – Test the hypothesis (~5 weeks)

O8 – Experiment: Test the hypothesis for 500 maps with a TSP size of 10 in Python (~2-3 days)

09 – Examine the results using either Python or R. (~ 2 weeks)

O10 - Repeat for 500 maps of 20 cities, and again for 500 maps of 50 cities. (~3 weeks)

O11 – Analyse all results drawn from the experiment and state the concluding stance on the hypothesis. (~ 1 day)