A New Heuristic Approach for TSP using Approximative Recursive Clustering

Max Nyström
Björn Forsberg

Abstract

The traveling salesman problem is a common NP-hard problem that needs quick and adequate solutions in many different areas e.g. in logistics, biology and manufacturing, to name a few. NP-hard problems can not be solved in polynomial time by brute-force solutions, which is why heuristic algorithms that can give suboptimal solutions quickly are needed. This study introduces a new heuristic algorithm based on subdividing the traveling salesman problem by use of clustering and proposes a new heuristic for connecting the subproblems together by approximating endpoints between clusters, resulting in a complete tour. Therefore, the posed research questions of the study concerns how the application of different clustering methods and the proposed heuristic of endpoint approximating affect the quality of the solutions produced by the algorithm. To enable the research, an experiment was conducted, data collection was done by observation, and the data was analyzed by the use of descriptive statistics. The results indicate that the choice of clustering method greatly impacts the closeness of the generated solutions to the optimal solution, where KMeans and Agglomerative Hierarchical clustering using an average inter-cluster distance metric clearly outperformed the other methods. Also, the approximation heuristic was concluded to improve the algorithm's results. The proposed algorithm can be used to quickly find a suboptimal solution for large instances of the problem. The proposed heuristic for approximating endpoints for open-loop subsolutions may be implemented into other solutions to enhance other heuristic algorithms and would need further research.

Sammanfattning

Handelsresandeproblemet är ett vanligt förekommande NP-hard problem som kräver snabba och adekvata lösningar vilka kan appliceras inom en mängd olika områden såsom logistik, biologi och tillverkning. NP-hard problem kan inte lösas i polynomiell tid genom brute-force lösningar, vilket gör att det finns ett behov för heuristiska algoritmer som kan generera suboptimala lösningar på kort tid. Denna studie presenterar en ny heuristisk algoritm som delar upp instanser av handelsresandeproblemet i subproblem genom användandet av olika klustringsmetoder och föreslår en ny heuristik för att koppla samman subproblemen genom att approximera ändpunkter mellan kluster för att erhålla en komplett lösning. Studiens frågeställningar rör därav hur användandet av olika klustringsmetoder och den föreslagna approximeringsheuristiken påverkar kvaliteten på algoritmens lösningar. För att möjliggöra forskningen skapades ett experiment, med datainsamlingsmetod observation, och insamlad data analyserades sedan genom användandet av deskriptiv statistik. Resultaten indikerar att valet av klustringsmetod har stor påverkan på hur nära algoritmens lösningar ligger i förhållande till den optimala lösningen. KMeans och Agglomerativ Hierarkisk klustring med ett average inter-cluster distansmått överträffade tydligt de andra använda klustringsmetoderna. Slutsatsen drogs att approximeringsheuristiken förbättrar algoritmens resultat. Den presenterade algoritmen kan användas för att snabbt erhålla en suboptimal lösningar för stora insatser av problemet. Den föreslagna heuristiken för att approximera ändpunkter för open-loop sublösningar skulle kunna implementeras i andra lösningar för att förbättra andra heuristiska algoritmer och är i behov av vidare utforskning.

Synopsis

Background

Combinatorially difficult problems cannot be solved exactly since no general method to do so effectively exists. Instead, heuristic methods are applied that attempt to find as good a solution as possible but can find a suboptimal solution. This study creates and assesses a new algorithm that can use different clustering methods as a way of subdividing the problem and a new heuristic that approximates the endpoints to be connected for the reassembling of the subproblems back together.

Problem

Subdividing the overall traveling salesman problem into subproblems and then reassembling them in a correct way is a hard problem and could potentially ruin the chances of a good solution. However, this is crucial for large instances of the traveling salesman problem in order to decrease the otherwise large computational time, stemming from the inherent time complexity of the problem.

Research Question

How do different clustering methods affect the quality of the solution in the proposed algorithm?

How does approximating the endpoints affect the quality of the solution in the proposed algorithm?

Method

Computer supported experiments were conducted measuring the expenditure of time for computation with different settings of the ARC (Approximative Recursive Clustering algorithm) artifact applied to various inputs of the traveling salesman problem. Also, the closeness to the proven optimum of these instances of TSP was compared with the resulting paths produced by ARC. The purpose of these experiments was to measure the performance of the proposed method.

Result

The results show that the approximation heuristic improves the overall solution for the algorithm and that the choice of clustering method is even more important with respect to the overall solution of the algorithm than the approximation heuristic.

Discussion

The algorithm created in this study can be used to suboptimally solve TSP problem instances, and the proposed new heuristic could be used in conjunction with earlier research solutions of the problem using a divide-and-conquer approach. Further research is needed to conclude the usefulness of the heuristic and further improvements of the algorithm.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

The traveling salesman problem (TSP) is a well-studied NP-hard combinatorial optimization problem. Figuratively speaking, the problem is finding the shortest possible route for a salesman visiting a set of cities, whereby all cities must be visited exactly once, lastly returning to the first city. By the use of graph theory, a mathematical field which can explain TSP in a formal manner, the problem can be described as finding the minimal Hamiltonian cycle in a graph. (Gutin & Punnen, 2006)

Computationally calculating an optimal route, meaning the proven shortest possible route, is a complex and time-consuming task, even for TSP problem instances with a relatively low amount of cities; for a computer, applying a brute-force approach in which all possible routes are tested and evaluated, finding the optimal route may take years. The reason for this is that the number of possible routes between each and every city has a factorial growth rate dependent on the amount of cities in the set. Above certain boundaries the problem might not even be practically solvable at all. (Gutin & Punnen, 2006)

To tackle the inherent complexity of the problem computer scientists and researchers apply heuristics (Pearl, 1984). In the context of this study the most important heuristics are; techniques used for the purpose of obtaining suboptimal TSP solutions of acceptable quality within reasonable time (Mariescu-Istodor & Fränti, 2021). In a recently published research paper Mariescu-Istodor & Fränti (2021) discusses the application of such heuristics in the designing of algorithms for large instances of TSP, i.e. large set of cities.

As stated by Mariescu-Istodor & Fränti (2021), the three major parts of importance for the overall performance of the algorithms are: a TSP solver which evaluates the routes; a dividing strategy which divides the problem into subproblems; and a merging strategy which merges the solutions of the subproblems to form a global solution.

When dividing the problem, the resulting subproblems are either closed-loop TSP cycles or open-loop TSP paths. A closed-loop route exhibits the properties of the original problem as it is required to return to the starting city. The open-loop TSP removes this requirement, instead ending at the last city. Removing the connection between the last city and the starting city in a closed-loop can result in either an optimal or a suboptimal open-loop. (Mariescu-Istodor & Fränti, 2021)

By choosing a suitable strategy for dividing the problem, the total computational time for evaluating paths can be significantly lowered and also increases the likelihood of attaining high quality solutions. Mariescu-Istodor & Fränti (2021) suggests using clustering, for dividing the problem into subproblems. However, as clustering techniques operate in different ways and produce different results in terms of the generated clusters, the issue of making the right choice revolves around which technique will produce the most optimized clusters. (ibid.)

Upon utilizing a dividing strategy, then arises an issue of how to assemble the solutions of the subproblems into a global solution. This process is referred to as merging the subsolutions (Mariescu-Istodor & Fränti, 2021). What particularly matters is figuring out which cities to assign as endpoints in the subsolutions, i.e. which city of a route in one subsolution should be connected to another subsolution route, as this greatly impacts the overall solution.

1.2 Problem

As of today, due to the inherent complexity of NP-hard problems, different heuristics need to be applied to the design of an algorithm for TSP. Specific aspects have a great impact on the algorithm's performance.

Designing a TSP algorithm is multifaceted and to assess the implications of the algorithm's inner properties, the designer merely has the tool of practically implementing and testing the execution. Due to the fact that well performing algorithms for TSP potentially are of great use in many real-world cases, attempting to design and test new methods or algorithms and to further improve state-of-the-art techniques is of great importance both for the scientific field and society in all.

As discussed above, heuristic for dividing the problem into subproblems, e.g. through the use of clustering, applied for the purpose of reducing the computational time for near optimal or optimal subsolutions, give rise to new challenges in merging the subsolutions to obtain a global solution.

Designing a well functioning merging strategy isn't trivial and this step of the algorithm may often lead to globally suboptimal choices by the algorithms. Furthermore, the dividing strategy together with the applied merging strategy has a great impact on the algorithm's overall performance and the quality of the global solutions. However, there is no definitive answer to what specific techniques should be applied. (Mariescu-Istodor & Fränti, 2021)

In essence, for this study, choosing the clustering method used to divide the problem, and then choosing appropriate endpoints between the clusters greatly impact the overall performance of the algorithm.

1.3 Research Question

Considering the stated problem, for this study a new algorithm was created. The algorithm uses clustering to subdivide TSP, creating open-loop subproblems, and a new approximation technique for choosing endpoints between the clusters, to connect the overall closed-loop TSP. The research questions for this paper is therefore:

Which clustering method best helps in providing the best TSP solutions?

By precomputing and approximating the connections between the open-loop subproblems, is it possible to improve the overall solution?

2 Extended Background

TSP and the methods considered in this study are related to such scientific fields as computer science, complexity theory, graph theory and combinatorics.

2.1 Graph Theory

To formally explain TSP, graph theory can be used. Graph theory is a field in mathematics dealing with finding relationships between data points (West, 2001).

The onset of graph theory is recurrently attributed to *The Köeningsberg Bridge Problem.* In essence, the problem is walking a route over a set of bridges connecting islands and the mainland of a city center, without crossing the same bridge twice. 16th century mathematician Leonard Euler used graph theory as a general model for exploring possible solutions to the problem, eventually concluding it unsolvable (ibid.).

TSP can be explained as a graph problem as follows. A graph consists of a set of vertices, representing the cities, which are connected together by edges, representing the routes. The edges have assigned weights, represented by the traveling distance, which make the graph a weighted graph. The graph may be undirected, representing symmetric TSP where the distance between connected cities is the same in both directions, or the graph may be directed, representing asymmetric TSP where the distance between cities may be of different size in opposite directions and routes may or may not exist in both directions. A complete graph is a graph in which every vertex contains an edge to all other vertices. (ibid.)

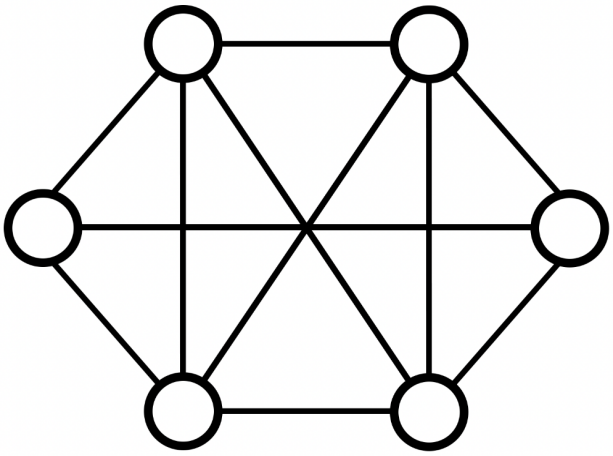

Figure 1. A Complete Undirected Symmetric Graph without weights

In graph theory, the shortest closed-loop TSP corresponds to the minimum Hamiltonian cycle and the shortest open-loop TSP corresponds to the minimum Hamiltonian path**.** A hamiltonian cycle is when all vertices in a set are connected through edges without any repeated vertex except the starting and ending vertex, resulting in a loop. A hamiltonian path is a path in a graph that visits each vertex exactly once. (ibid.)

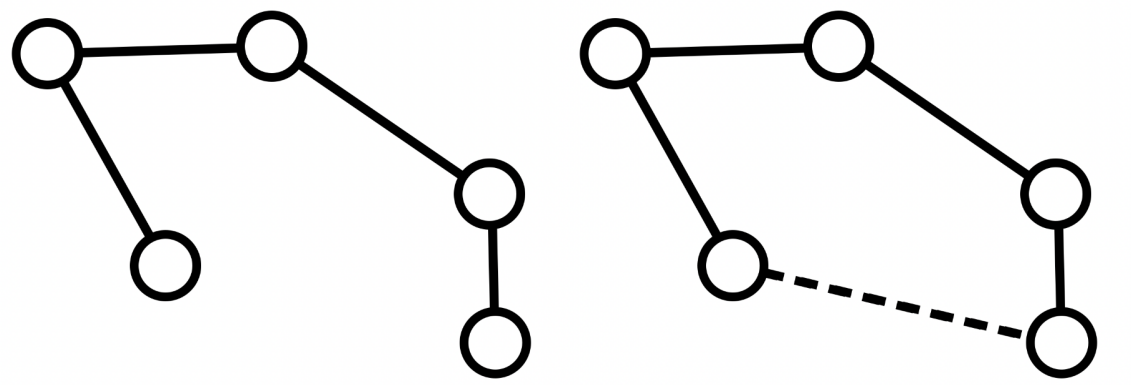

Figure 2. Difference Between a Hamiltonian Path (Left) and a Hamiltonian Cycle (Right)

2.2 The Traveling Salesman Problem and its variations

Records can be found of studying TSP in the early 1930's whereas the first systematic research of the problem began roughly twenty years later and from then on the research area has steadily been expanding and advancing. The level of difficulty in solving larger problem instances, along with the numerous real-world usages for the optimal as well as suboptimal solutions has contributed to the vast and in-depth research performed on many variations of TSP. (Gutin & Punnen, 2006)

Practical applications of TSP range from less complex problems**,** such as properly lacing a shoe to more advanced ones such as printed circuit board assembly, planning and scheduling, routing of vehicles and many more. (Gutin & Punnen, 2006; Mariescu-Istodor & Fränti, 2021)

Many variations on the base problem exist, each having its own specific properties and conditions, e.g. the MAX TSP which attempts to find the longest path and clustered TSP where the problem is subdivided into clusters where the path found must include all vertices in a cluster before connecting the clusters together. See (Gutin & Punnen, 2006) for an extensive exposition on TSP variations.

One particularly useful variation is The Vehicle Routing Problem (VRP), a generalization of TSP. VRP has a wide range of practical real-world applications (Gutin & Punnen, 2006). Originally introduced by Dantzig and Ramser (1959) as "The truck dispatching problem", VRP is concerned with finding a set of optimal delivery routes from depots to customers (Toth & Vigo, 2002). Routes are assigned to a corresponding number of vehicles, each starting at its individual depot and completing its assigned route by returning back to the same location. In other instances of the problem, all vehicles may depart from a central depot. The goal is ascertaining that customer satisfaction, operational constraints and minimum traveling cost for the full set of routes is achieved. A graph representation of such a problem is generally asymmetric since a road network may have two way streets and alternative routes in opposite directions with differing traveling costs (ibid.).

VRP takes into account more factors than TSP and is harder to solve optimal or near optimal. As a consequence, problem instances with no more than around 30,000 customers

is at present the upper bound considered in scientific research. However, there is a need for solutions to much larger instances since for example currently operating delivery companies make millions of deliveries in a single day. (Mariescu-Istodor & Fränti, 2021)

2.3 Complexity Theory

The Theory of P vs NP

In mathematics, problems can be categorized through application of a theory called P vs NP. TSP is classified as an NP-hard problem. The fact is that as of today NP-hard problems cannot be optimally solved in polynomial time and it is suspected that this may never be possible, this is part of what is formally called the P versus NP problem. (Goldreich, 2010)

By the formal logic presented by Gutin & Punnen (2006), it is stated that as polynomial time algorithms constitute the class P, symmetric TSP belongs to the class NP. The authors maintain that if a computational problem, in this case symmetric TSP, "is as 'hard' as an NP-complete problem but not necessarily in NP" (Gutin & Punnen, 2006, pp. 756) it is categorized as NP-hard. See Gutin & Punnen (2006) for a more in depth explanation.

Time Complexity

In computer science, algorithms belong to classes of time complexity, an approximation measurement for assessing the execution time of an algorithm, e.g. logarithmic, linear, quadratic or exponential complexity. This categorization of an algorithm is done by using Big-O notation, a mathematical notation for measuring the growth rate of functions. Big-O is applied in algorithmic analysis to roughly estimate the worst case of execution time of an algorithm. The measurement is deliberately rough and avoids insignificant details with the benefit of being applicable independently of programmatic implementation and language specifics. (Weiss, 2012)

The time complexity classes in order, from fastest to slowest, are as follows: O(1) constant; O(log N) logarithmic; O(N) linear; O(N log N); $O(N^2)$ quadratic; $O(2^N)$ exponential.

In certain cases an algorithm's computational speed is preferred over accuracy, regarding the obtained solution. In selecting a suitable algorithm, computer scientists use time complexity as a guiding tool. (Weiss, 2012)

2.4 Optimal solutions for TSP

Obtaining optimal solutions for TSP is the main difficulty of the problem because the number of possible paths to calculate before reaching a final solution is often very large making it impractical in terms of time consumption. Heuristics generally compromise with the quality of the solutions due to this fact. However, for small enough problem instances it is possible to produce optimal solutions for TSP, in a reasonable amount of time. A brute-force approach can be applied, meaning all possibilities are evaluated which quite logically would result in the finding of the optimum. (Mariescu-Istodor & Fränti, 2021)

On the other hand brute-force is in most cases a resource-intensive approach. As stated above, the issue of computational time is closely connected to the number of cities in a TSP instance. Consider applying a brute-force approach to solving TSP:

For a TSP instance of *N* cities the number of possible paths in the set is (*N* - 1)!/2. Hence, for a TSP problem of a relatively small size of 20 cities, the time required, at the rate of a million possibilities tested every second, exceeds a thousand years to compute (Karp, 1986).

There exists other techniques which can find optimal solutions for TSP. Branch and bound which uses branching for subdividing the problem (Murty et al., 1963), and dynamic programming approaches (Gutin & Punnen, 2006).

2.5 Cluster analysis

Cluster analysis as stated by Wierzchoń & Kłopotek (2018) is the process of categorizing a set of objects into groups called clusters where in whichever two objects share more similarity than whichever two objects in another cluster. The purpose of the analysis is to obtain the inner structure of a data set, in the case of TSP finding relations between the vertices in a graph. Also, the authors argue that when performing cluster analysis, a similarity measure, used for the comparison of different objects, must be chosen as well as an approach for using the similarity when generating clusters (ibid.).

Inter-cluster-distance functions, used in hierarchical agglomerative clustering methods, are used to measure the similarity or dissimilarity between two clusters (Wierzchoń & Kłopotek, 2018). For this study *single-linkage*, *average* and *complete* was considered. This is explained further below.

2.6 Clustering Methods

By means of reducing the time complexity for generating a global solution, TSP algorithms may use clustering as a heuristic for dividing the larger problem into subproblems (Mariescu-Istodor & Fränti, 2021). Presented in this section are some of the clustering methods applied in previous research and those that have been considered in the process of this study.

2.6.1 DBSCAN

Originally presented by Ester et al. (1996), the DBSCAN algorithm (an acronym for density-based spatial clustering of applications with noise) is as the name implies, a density-based clustering method. Such a method is generally applied to data sets containing clusters of varying shape with outliers (Wierzchoń & Kłopotek, 2018). DBSCAN categorizes points in clusters as either internal points or border points, depending on the points relation to the value of the parameter *minPts*, or as outliers (ibid.). *MinPts* together with the parameter *k*-distance need to be specified. The accuracy of the value of *k*-distance is especially important since a too large or too small value may yield unwanted results. A shortcoming of the algorithm is the requirement to specify the *minPts* value beforehand as it forces the user to more or less blindly estimate the cluster density. This was later solved by HDBSCAN (ibid.).

The sequence of steps of DBSCAN are executed as follows: Categorize the points in the data set; remove the outliers; connect adjacent internal points, positioned at a relative upper bound distance from each other, by an edge; create a cluster made of adjacent internal points; and lastly connect border points to their closest cluster. (Wierzchoń & Kłopotek, 2018)

A lack of available research for the use of DBSCAN as a clustering method in solving TSP instances may indicate a foundational inefficiency. However, Sriraman & Bays (2014) presents an algorithm which utilizes DBSCAN clustering in combination with a greedy TSP algorithm for transiting and surveying underwater paths.

2.6.2 HDBSCAN

HDBSCAN, a later modification of DBSCAN, tries to improve on some of its predecessors inadequacies. Unlike DBSCAN, it can produce clusters of varying density. The algorithm makes use of hierarchical structuring as used in agglomerative methods. From a flat set of clusters, hierarchical structures are obtained and modified. The algorithm initially builds a minimal spanning tree (MST) working on the distances between elements. Also, a dendrogram is created and too small clusters are categorized as outliers. The dendrogram is processed from the leaf nodes up to the top, producing the resulting actual set of clusters. Two parameters require specification, the number of neighbors *k* and the minimal cluster size. (Wierzchoń & Kłopotek, 2018)

As in the case of DBSCAN, our investigation of the research indicates that HDBSCAN clustering as a heuristic for TSP is considered or applied in relatively few research papers. However, one interesting utilization is the context-aware system presented by Korakakis et al. (2017) which is intended to advise tourists on traveling routes that the system generates by use of HDBSCAN applied to a variation of TSP.

2.6.3 *K*-Means

The *k*-means algorithm, first conceived in the 1950's, is part of the family of algorithms of combinatorial cluster analysis. The algorithm is nondeterministic and regularly used in data mining. A downside of the algorithm is that the user needs to know the number of clusters in the data set beforehand. (Wierzchoń & Kłopotek, 2018)

Its core functionality revolves around iteratively assigning elements to clusters with the cluster centroids as the reference point and continuously reevaluating cluster centroids until convergence**.** Being a heuristic, the algorithm does not guarantee an optimal solution. However, if the targeted elements of the data set is easily categorized by means of differing properties, the algorithm has the prerequisites for good performance. The more disparate groups, the less iterations are needed for full analysis of the data set. (ibid.)

Starts with specifying a target data set and number of groups *k.* Find centroids and assign the elements of the data set to classes. Step one selects centroids' gravity centers and assigns weights to objects. As the second step, the assignments to clusters based on proximity to centroids are updated for each element. Third, the cluster centroids are updated. The last step is repeating step 2 and 3, which is usually done until reassignment possibilities are exhausted. (ibid.)

It was discovered that research on *K*-means clustering applied to TSP seems to be more common than the previously mentioned clustering techniques, especially in combination with genetic algorithms. The papers by Nallusamy et al. (2009), Deng et al. (2015) and Ferrandez et al. (2016) are excerpts of research pertaining to the area.

2.6.4 *K*-Medoids

K-medoids, constructed to be able to better handle outliers of a data set, is closely related to *k*-means. The most significant difference between the two lies in the calculation of the cluster centers. In *k*-medoids, medoids are actual points in the clusters categorized as the center based on their location, whereas the *k*-means centroids are calculated mean values of a cluster's points and do not need to correspond to an existing point. Also, *k*-medoids make use of a dissimilarity matrix which extends the applicability of the algorithm to non-Euclidean space. A disadvantage of *k*-medoids is however that, in terms of time complexity, it far exceeds *k*-means. (Wierzchoń & Kłopotek, 2018)

A scarcity of good quality, available, cited and peer reviewed research using *k*-medoids as a heuristic for TSP may indicate the method is unfit or not well researched.

2.6.5 Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering is a clustering method, with a bottom-up approach, which uses specified inter-cluster-distance functions for connecting clusters. The resulting division into clusters is presented as a dendogram, displaying the relations between objects and clusters generated from a data set. The algorithm initially categorizes each object as a separate cluster and then iteratively connects clusters until all objects are contained in one large cluster. This aggregating process makes use of a distance matrix, checking and modifying it, and adding relations and objects to the dendrogram. (Wierzchoń & Kłopotek, 2018)

Depending on the inter-cluster-distance function used, clusters with different properties are obtained. The most important functions for this study is: *single-linkage*, which measure the distance between the two closest points in two clusters; *average*, which finds the mean points in two clusters and compares the mean points' distance as the distance between the clusters; and *complete*, which is the distance between two points in two clusters which are the farthest away in the clusters. (ibid.)

With *single-linkage,* clusters are built bottom-up from finding the closest distance between points or clusters and then merging them. Let two clusters be *X* and *Y*. In *single-linkage*, measuring the distance between *X* and *Y* is done by calculating the minimum distance *x* to *y* where *x* is a point in *X* and *y* is a point in *Y.* The single-linkage algorithm, on a weighted complete graph, corresponds to a minimum spanning tree (MST) of that graph (Alpaydin, 2020).

2.7 Current Research and Heuristics

Heuristics can be applied to the algorithmic design, generally trading quality of the solutions against faster computational time which result in suboptimal solutions with an actual

real-world usage (Mariescu-Istodor & Fränti, 2021). Scalability becomes an obstacle since the vast majority of heuristic TSP algorithms do not have a linear time complexity (ibid.).

The algorithm designer has the possibility of experimenting with the relation between time consumption and procentual closeness to the optimum, i.e. quality of the solution (Mariescu-Istodor & Fränti, 2021). Certain algorithm design choices have a great impact on the scalability, performance and accuracy of a TSP algorithm. However, due to the complex nature of TSP, scalability is still an issue even with heuristic strategies applied (ibid.). Practically, TSP algorithms of time or space complexity $O(N^2)$ or worse may not be useful when applied to large enough problem instances because of memory or computational time limitations (ibid.).

2.7.1 Non-clustering strategies

State-of-the-art heuristics for solving TSP are presented by Mariescu-Istodor & Fränti (2021). A central point for many TSP algorithms is the usage of local search, one of the most commonly used heuristics for obtaining close to optimal solutions (Sengupta et al., 2019). The key concept of local search is trying to enhance an initial solution through incrementally exploring neighboring state space paths via a set of vertices (ibid.).

Mariescu-Istodor & Fränti (2021) argues that the most important thing to decide is the operator dictating the local search strategy. The operator is the component of the algorithm that will determine in what manner the initial solution is altered thus affecting the possible variations of substitutional paths generated (Mariescu-Istodor & Fränti, 2021).
Relocate (Gendreau et al., 1992), Link swap (Sengupta et al., 2019), 2-opt (Croes, 1958) and *k*-opt (Shen and Kernighan, 1973) are operators that in different, however similar, ways rearrange links of an initial solution. As it is concluded by Strutz (2021) that Relocate and Link swap are special cases of *k*-opt, and *k*-opt is in fact the generalization of 2-opt (Mariescu-Istodor & Fränti, 2021), for concise purposes a brief explanation of only *k*-opt follows. Basically, the *k*-opt operator decides on which *k* links in the initial solution to be redirected in between the containing vertices in order to obtain a better solution (ibid.).

In addition to the operators, local search algorithms generally implement a strategy for choosing one of the generated solutions. Mariescu-Istodor & Fränti (2021) describes three such strategies: *random search*, which accepts any solution better than the initial; *best improvement*, which evaluates all possibilities and chooses the best one; and *prioritization*, which chooses operators and their parameters based on certain criteria (ibid.).

Due to the fact that only a small amount of the acquired new paths are good options, stemming from the inherent workings of the operators, Mariescu-Istodor & Fränti (2021) suggests the use of one of two localization strategies for limiting the search space, *grid* and *neighborhood graph* strategy. This is done by restricting operators to evaluation of vertices in close proximity. The grid strategy, which creates cells over the graph, forces the operator to evaluate the current cell's edges and vertices and close proximity targets in adjacent cells (ibid.). The neighborhood graph strategy uses specific types of graphs onto which linear performing algorithms are applied to generate 2D geographic data (ibid.). Localization is concluded to be an overall effective way of minimizing the computational load in evaluating

different paths, despite the fact that both strategies in the process may reject an amount of viable options (ibid.).

2.7.2 Clustering strategies

Many algorithms for solving TSP apply a step where the full set of vertices is divided into smaller subsets using spatial clustering. Basically, the full problem is divided into smaller parts which are solved by calculating a closed-loop TSP solution over the clusters and solving the individual clusters as open-loop TSP solutions, and then merged to form a global solution. Mariescu-Istodor & Fränti (2021) suggests several methods for this task such as *k*-means, grid-based methods and single-link clustering. Furthermore, they argue that *k*-means is not always optimal, due to the fact that it creates spherical clusters which may not reflect the optimal TSP path, grid-based is even worse because it divides the cities into squares. On the other hand, single-linkage clustering may provide a possibility for the clusters to allow for the optimal solution to be calculated (ibid.).

2.7.3 Merging Strategy

Apart from the clustering method and the solving of the subproblems a decision must be made about how and when to assemble the subsolutions into a global TSP solution. This merging strategy can be applied before or after the subproblems are solved. However, designing a merging strategy, and its accompanying dividing strategy, that generates optimal or close to optimal results is a complex issue (Mariescu-Istodor & Fränti, 2021). One possible heuristic approach for merging, as done by Kobayashi (1998) and discussed by Mariescu-Istodor & Fränti (2021), is to create a sort of higher level abstraction solution by calculating a route between clusters of cities and thereafter solving the subproblems in each cluster recursively. Using such an algorithmic divide-and-conquer approach, the optimization for local and global solutions are dealt with separately (Mariescu-Istodor & Fränti, 2021). Mariescu-Istodor & Fränti (2021) claim there is a high probability for solutions being suboptimal, but that this can be calibrated as a later step in the algorithm, using the *k*-opt as a subsequent optimization step on the global solution.

As has been shown in this section, when constructing a TSP algorithm the designer faces a multitude of design choices. State-of-the-art algorithms and combinations of methods are continuously being tested and improved (Mariescu-Istodor & Fränti, 2021). In this study's context there remain open questions to what may or may not be the best choice for a particular implementation or algorithmic design. With this in mind, the aim of this study has been to evaluate the clustering and merging strategies through construction of a new algorithm, test its performance and evaluate the results.

3 Method

This study utilizes the methods listed below based on directives and arguments stated by Denscombe (2014) and Johannesson & Perjons (2014). Also, ethical guidelines and considerations regarding the research are based on Denscombe (2014) and Johannesson & Perjons (2014).

In scientific research there exists an array of research strategies and methods, the suitable choice is depending on the study's goals and inherent properties. The research strategy provides a means of structuring, monitoring and executing the study from start to finish (Johannesson & Perjons, 2014) and the research method serves the purpose of obtaining a relevant data set (Denscombe, 2014). Both should be chosen with respect to the research question and, if executed properly, enforcing the establishment of scientific rigor to the study, i.e. validity and reliability of the research results (Johanneson & Perjons, 2014).

3.1 Research Strategy

According to Johanneson & Perjons (2014) and Denscombe (2014), the three main considerations for choosing an appropriate research strategy are: *suitability*, if it provides adequate data for answering the research question; *feasibility,* if the research project is practically doable; and *ethics*, can the research be conducted in accordance with ethical standards?

Experiments are *suitable* since the central subject of this study is an artifact, the ARC algorithm, as the strategy enables close control of artifact parameters and test conditions which facilitate the isolation of factors of cause and outcome and the exactitude of empirical measurements and observations - all of which aid the ensuring of internal validity (Johannesson & Perjons, 2014). Also, it is *feasible* given that time estimates of the duration of conducting experiments is predictable and experiments may be performed in a relatively short time compared to other strategies (Denscombe, 2014), which is convenient given the rather tight time constraints of this research project. The ethical considerations of this study are discussed later in section 3.5.

A downside to experiments may be that of a low external validity due to the artificiality of the circumstances in which the experiment is conducted, making the results hard to replicate thus implying a lack of generalizability of the findings. However, in an effort to increase external validity, the algorithm's source code used during the experiments will be attached in appendix A. Specific programming language versions and hardware environments would plausibly affect the validation to some extent and therefore the experiment's exact hardware and software environment will be specified in the section concerning the application of the research method.

As argued by Denscombe (2014) when considering using experiments as a research strategy, an in-depth research on the topic and relevant theories should be performed and then analyzed to distinguish important factors to examine. For this study the conclusion was that in particular the impact of the algorithms dividing and merging strategies were good

choices to be targeted as factors for the experiment and the focus has therefore been to investigate how specific changes in the algorithmic design affect the results.

An experiment assists the researcher in establishing if changes in independent variables have an impact on observable changes in specific dependent variables (Johannesson & Perjons, 2014). For this particular study e.g. a specific clustering method producing a certain quality of the solution or the precomputation and approximation of the connections between the open-loop subproblems improving the overall solution.

Similar research has been performed by Kobayashi (1998) Sengupta et al. (2019), Strutz (2021) and several others, which also apply the strategy of performing experiments with the purpose of observing and controlling specific variables for comparison of results. Hence, this approach seems rather established, if not even conventional, and therefore arguably a suitable choice for this study.

3.2 Alternative Strategy

A case study, where one, or possibly a small number of instances of a phenomenon are studied in depth and in context (Denscombe, 2014), was considered as an alternative to experiments. A case study should focus on one single instance's distinctive properties in a natural context (Johannesson & Perjons, 2014), the instance should preferably be a non artificial pre-existing instance and not one constructed for controlling and measuring certain factors (Denscombe, 2014). On the basis of this and due to the fact that the aim of this study has been to investigate a variety of different configurations of the proposed algorithm and comparing their respective performance in controlled settings, the strategy was deemed unfit**.** Also, the intention to examine the algorithm's performance on a set of problem instances of TSP, measuring an average procentual closeness to the optimal solutions, is an investigation of breadth and quite the opposite of an in-depth exploration of a single instance.

According to Denscombe (2014), a case study should put less emphasis on the produced results of an instance and more on the dynamics of an instance in its setting and the reasons for the involved occurrences, which give the researcher the possibility to explore as to why a certain result was obtained. As the intent has been to focus on comparing the produced results of the algorithm, contextual factors such as the possible impact of specific underlying hardware and other factors plausibly affecting the outcome were of minor importance to this study. Furthermore, Denscombe suggests case studies as a means of attaining a more holistic view on a research subject, how the parts of an instance affect each other and are affected by the settings. Instead, the interest of this study lies in how isolated factors, such as a particular choice of clustering method, affects the quality of the solution produced by the algorithm.

Another argument for conducting a case study rather than experiments is that when an artifact, such as the ARC algorithm, is subjected to controlled laboratory experiments the external validity may be negatively affected due to the conditions of the experiment settings being different from other possible experiments or real-life applications (Johanneson & Perjons, 2014). To increase the likelihood of attaining greater external validity, a case study

where the artifact is practically applied could be performed (ibid.). However, as the focus of this study was to see the effects of different clustering methods and the approximation of endpoints, an experiment was deemed more suitable.

3.3 Data Collection Method

A research method requires a data collection method that supports obtaining an adequate dataset for the study. The resulting data set contains either qualitative, quantitative or both types of data (Johannesson & Perjons, 2014). According to Denscombe (2010), there are four applicable types of data collection: *survey*, in which information about the research subject is gathered by researchers asking informants a set of pre-created questions; *observation*, researchers collect data through observing the research subject; *interview*, where data is collected through different types of interviews with informants; and *documents*, where data is retrieved from pre-existing sources such as reports, media or similar.

As this study evaluates an artifact which is not intended to be assessed or used by a user, questionnaires and interviews are not suitable as these are regularly used for gathering information of informants experience and attitudes towards a specific research subject. This is irrelevant to this study as the purpose of the data collection is to assess the performance of the algorithm based on the numerical ratio data produced by the testing framework. Documents are neither suitable as there are yet no documents about the algorithm. However, documentation of similar pre-existing algorithms have been studied and have been considered as sources for inspiration and ideas in the construction of the new algorithm. (Denscombe, 2014)

Based on what is stated above, observation seems to be the only suitable data collection method in this study because of the nature of the conducted experiments, which will measure the algorithm’s overall performance and compare the results producing observable quantitative data intended to answer the research questions.

3.4 Data Analysis Method

Denscombe (2014) describes data analysis as a process of describing, explaining and interpreting the data set.

In this study, as the data is quantitative, descriptive statistics is a suitable data analysis method (Denscombe, 2014). The analysis and latter presentation of the data set will be facilitated through the use of descriptive statistics by compiling, comparing and illustrating relevant aspects of the data. By compiling the data in such a manner, a clear exposition of the data is obtained which aids the process of drawing conclusions (Johanneson & Perjons, 2014). Measurements of central tendency and measurements of dispersion are used to compare and analyze the algorithm's performance on different inputs, over a set of executions and with different settings. During this process the research question is kept in mind and will inform how the data analysis will be performed.

Credibility of the quantitative data can be ensured if it is accurately and consistently produced and in accordance with the research question, i.e. the validity is high. This can be

achieved through the use of a measuring instrument of high reliability (Denscombe, 2014). Since the measuring instrument for this study is a computer, its reliability is assumed to be high. However, there might be interfering factors affecting the execution of the algorithm, and in turn affecting the dependent variables. Also, one known interfering variable is that some of the chosen clustering methods used in the algorithm also contain a randomized part which could impact the results. Therefore, measurements of central tendency will be applied to a determined amount of executions, producing average measurements in an effort to equalize possible fluctuations in the execution time and results. The reason behind this is to compensate for unknowns possibly negatively impacting reliability of the measurements.

3.5 Ethical Considerations

Studies involving humans, animals or the environment calls for ethical considerations (Denscombe, 2014). Since neither the research strategy, the data collection method, nor the data analysis method have these entities as the direct subject or involved in the process, such considerations were not an issue. No direct ethical concerns were needed to take into account due to the nature of the experiment only including the subject software and hardware, and not living beings, plants or the environment. However possible repercussions are considered as part of the study’s discussion.

4 Application of Research Method

4.1 Experiment

The aim of the study is to investigate and evaluate the performance of the constructed artifact, the ARC algorithm. More precisely, the intention is to assess how the two proposed heuristic components, the applied clustering method and the endpoint approximation, affect the algorithm's resulting output when applied to selected instances of TSP. To achieve this, an experiment was conducted with the purpose of comparing the algorithm's performance, with or without endpoint approximation and with different clustering methods applied.

The experiments were performed on a 2021 Macbook Pro with an Apple M1 Pro processor and 16 GB RAM. The algorithm was written in Jupyter and executed using Python 3.8.9.

The problem instances used as input were taken from TSPLIB (Reinelt, 2018) which contains a library of TSP's with proven optimal solutions. The instances were chosen randomly with the requirement of an euclidean measure. The numeric part of a problem's name indicates the number of points contained in the particular problem. Solution lengths were measured with an euclidean distance metric.

Table 1. Problem Instances and Optimal Length

TSPLIB Instance	Optimal Solution Length
pr76	108159
kroC100	20749
eil101	629
tsp225	3916
a280	2579
pcb442	50778
pr2392	378032
rl5934	556045
usa13509	19982859
d18512	645238
pla33810	66048945

Two specific variables were measured: *execution time*, the time taken from providing an input to the algorithm until a solution has been outputted and *generated solution length*, the total euclidean distance of the output solution. With the algorithm's generated solution length, *percentage change* from the closeness to the optimal path could be calculated.

Due to the time constraints of this study multiple executions of the same instances with the same configurations was not possible.

The comparisons of the output were illustrated and analyzed by the use of descriptive statistics in the form of tables and charts, as recommended by Denscombe (2014).

4.2 Algorithm Parameters

The experiments consisted of testing the algorithm with different configurations on the same input data. The parameters that was changed for each configuration were: *clustering method*, either KMeans, two variants of Hierarchical Agglomerative clustering using different inter-cluster distance measures (single-linkage or average) and KMedoids; the *cluster amount* the algorithm used (either 8 or 5); and *endpoint approximation*, which was either turned on or off.

5 Results

Results presented are both the created artifact i.e. The ARC algorithm and details surrounding how it behaves, and the results it gives on TSPLIB inputs.

5.1 Created Artifact

5.1.1 Pseudocode

The ARC algorithm consists of three major components. The closed-loop, which dictates the overall execution of the algorithm from input to output using the two other components, the approximation, which evaluates and chooses endpoints, and the open-loop, which, depending on the number of points in a given cluster, either recursively or by brute-force approach generates a path.
Each component is explained in pseudocode below.

5.1.1.1 Closed-Loop

Input: Points
Parameters: Clustering Method, Approximation on/off
Output: Closed-Loop TSP solution

1. Read input data and group it into clusters
2. Calculate the centroid of every cluster
3. Calculate coarse closed-loop solution using the cluster centroids
4. Calculate temporary endpoints using single-linkage between the clusters adjacent to each other in the coarse closed-loop solution
5. Choose new endpoints based on the result of Approximation()
6. Recursively solve each cluster as a open-loop solution using Open-Loop()
7. Return a list containing all the points in the order of the solution

5.1.1.2 Approximation

Input: Clusters in an iterative data structure, Temporary endpoints for each cluster
Output: new endpoints

1. For each cluster:
    a. Calculate a coarse open-loop solution for the cluster before and after
    b. Use the last point in the before solution and the first point in the after solution as new endpoints.
    c. Save the new endpoints in a data structure
2. Return the new endpoints for every cluster

5.1.1.3 Open-Loop

Input: All the clusters in an iterative data structure, Endpoints for the clusters
Output: Open-Loop TSP solution

1. For each cluster:

a. If (num points in cluster > threshold):
    i. Group the points into new clusters
    ii. Calculate new coarse open-loop solution
    iii. Approximate endpoints for the new open-loop solution
    iv. Solve recursively using Open-Loop()
b. else:
    i. Brute-force optimal solution
    ii. Return list containing open-loop solution of the points in the cluster.

5.1.2 Approximation details

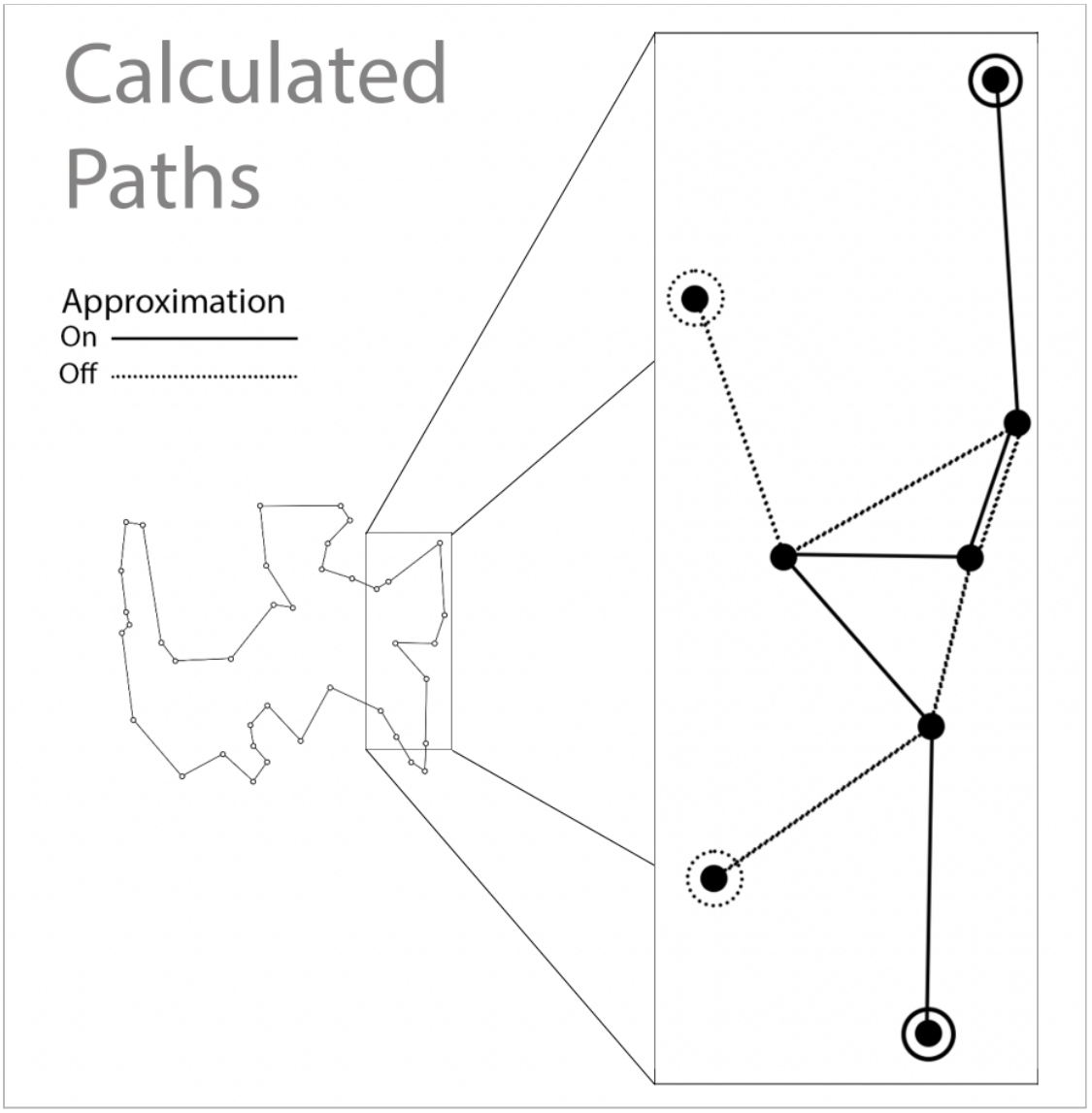

Figure 3. The two different paths calculated by different endpoints.

The algorithm's approximation step affects the final output as shown in figure 3. By letting the computer choose new endpoints (shown with black circles around them in figure 3) the end result of an open-loop solution of a cluster could change. The filled line solution in figure 3 shortens the overall closed-loop solution, compared to the dotted line solution, and thereby gives a better result in this instance.

5.1.3 Example

Here is an example of the execution of the algorithm shown with 40 randomly generated points with approximation on using hierarchical agglomerative clustering with average inter-cluster distance metric.

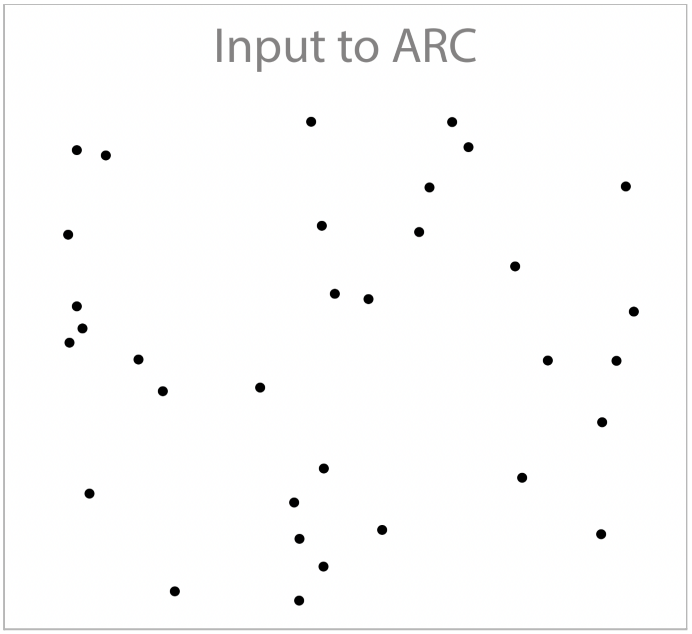

Figure 4. Cities represented as points

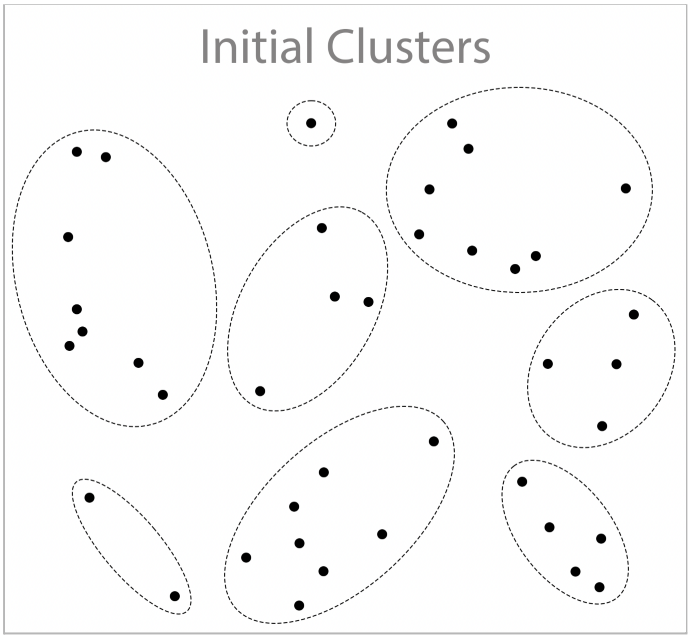

Figure 5. First grouping of cities. Clusters are created using Agglomerative average clustering. Dashed line indicates which points belong to which cluster.

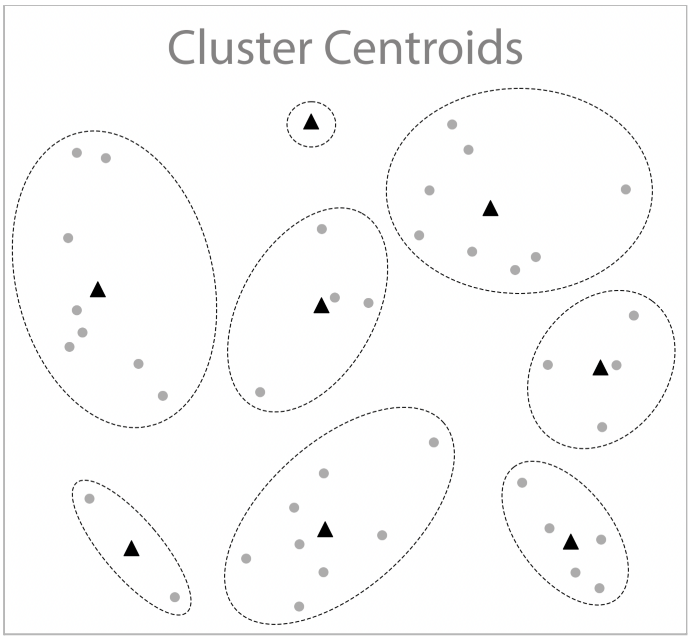

Figure 6. Calculated centroids represented by the black triangles.

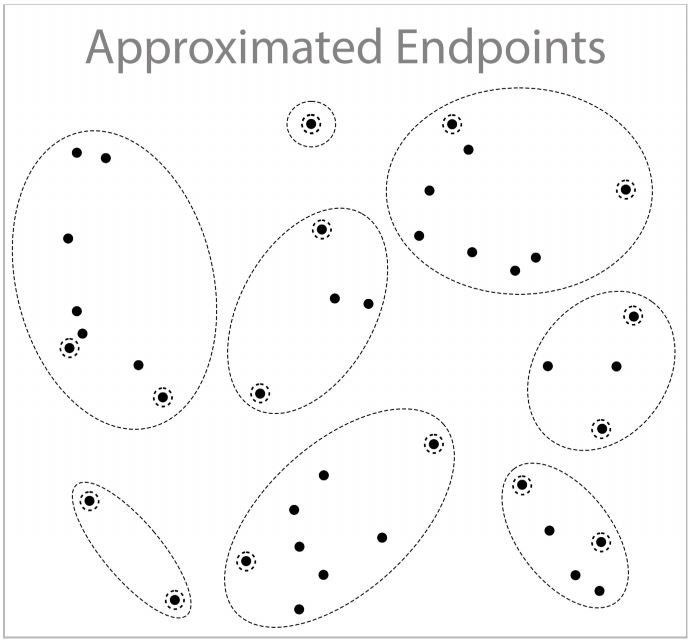

Figure 7. Calculated approximated endpoints are annotated with dashed circles around them.

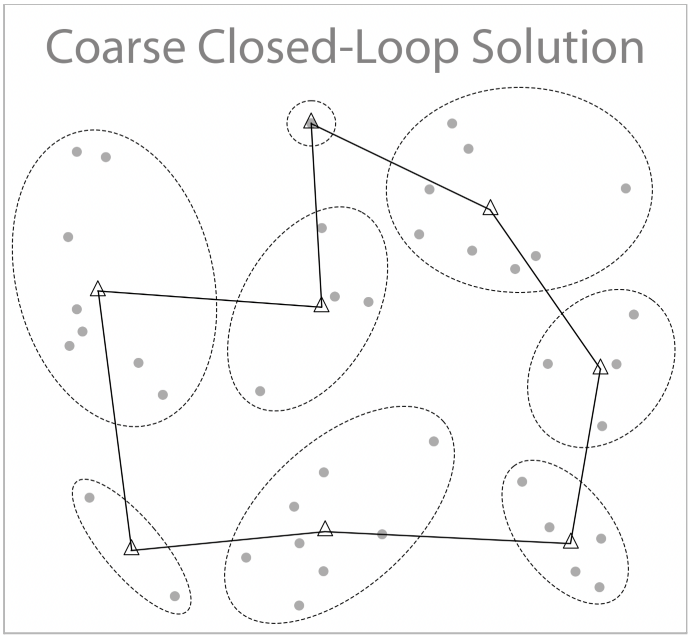

Figure 8. First closed-loop solution with centroids calculated using a brute-force approach.

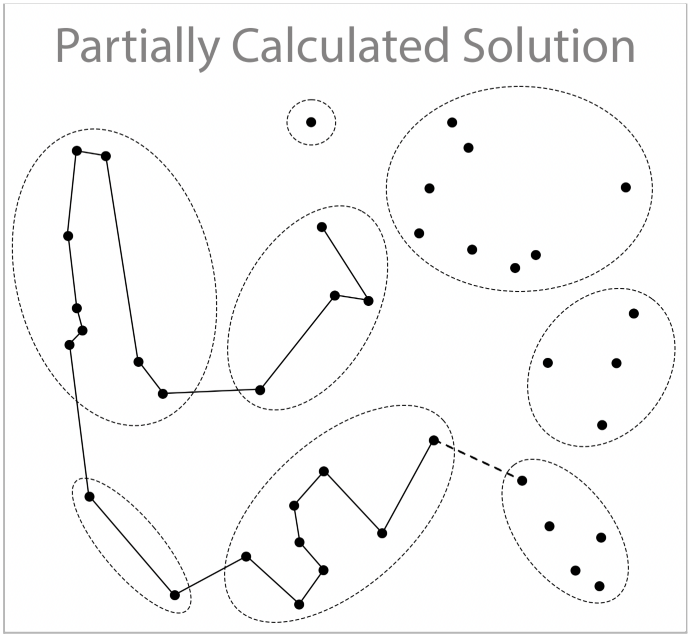

Figure 9. Midway step in the open-loop part of the algorithm. Edges shown in black are final.

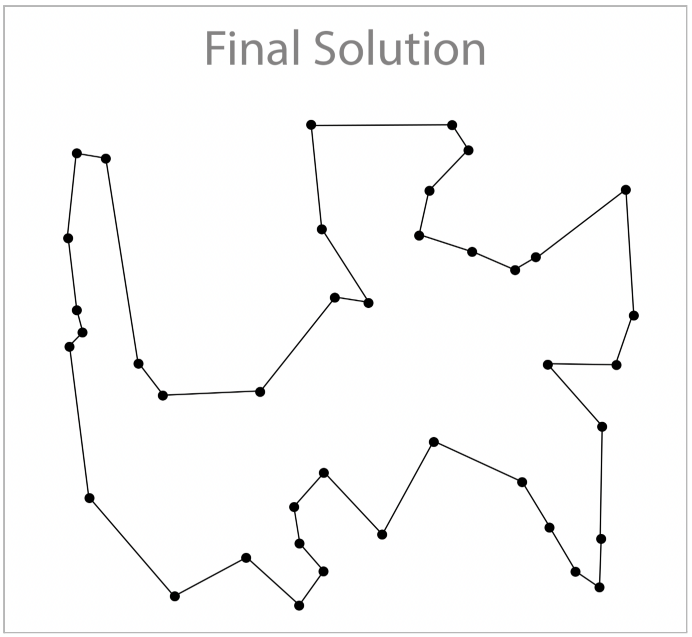

Figure 10. The algorithm's final output.

5.2 Experiment Results

The results of measuring the percentage change from the optimal solution, the algorithm's assumed time complexity, and the impact of 5 versus 8 clusters applied to the overall best performing clustering method Aggl. Avg. are presented below.

Table 2. Measurements of central tendency of percentage change from optimal distance using 8 clusters by Approximation and chosen Clustering Method.

	KMeans		**KMediods**		**Aggl. Avg**		**Aggl. Single**	
Approximation	On	Off	On	Off	On	Off	On	Off
Mean	12.99	13.10	19.73	20.79	9.79	11.34	76.45	79.08
Median	13.74	14.50	19.85	19.55	9.98	11.05	45.52	51.35
Std. dev	5.75	6.13	7.01	7.42	4.61	4.89	70.62	70.26
Max	21.96	22.71	36.38	39.18	17.40	19.55	218.53	220.24
Min	3.07	1.75	11.61	12.72	2.39	4.96	19.49	20.51

The aggregated data in table 2 indicates that the chosen clustering method has a big impact on the generated solution's closeness to the optimal solution. In this regard, the clustering method that produced the overall best results was the Aggl. Avg (Hierarchical Agglomerative method, with an average inter-cluster distance metric). Also, as concluded by comparing the *standard deviation* and *median* values between the clustering methods, Aggl. Avg produces the most consistent results.

The approximation heuristic improves the overall result for all clustering methods, as illustrated in table 2. However, the mean values of KMeans regarding the approximation shows that the results for some clustering methods could be relatively small as this difference is of only 0.11. On all the other clustering methods the approximation heuristic improves the mean value by at least 1.06, using KMedoids, and at most 2.63, using Aggl. Single (Agglomerative Hierarchical, with a single-linkage inter-cluster distance metric).

The *min* values show the best result from the data that the algorithm produced with those parameters. The best solution was found by KMeans with the approximation turned off. The worst solutions are shown by the *max* values in table 2.

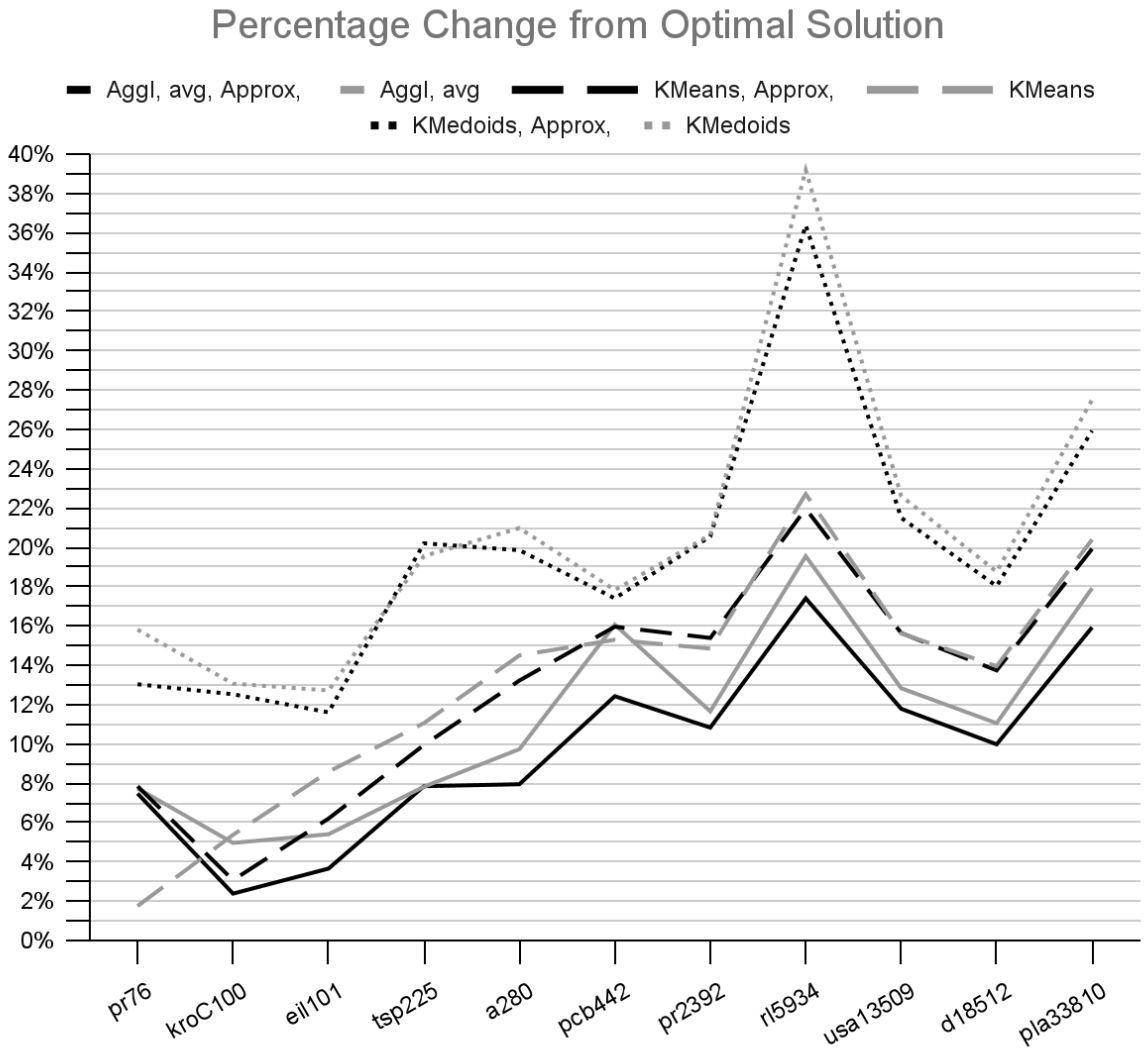

Figure 11. Each line represents different configurations for the algorithm and shows the percentage change from the optimal solution for every TSPLIB instance.

The chart in figure 11 is not linear on the x-axis compared to the input sizes. The instances shown are sorted by size. The Aggl. Single was excluded from the chart because the results

were very poor in comparison with the other methods, making the chart hard to read if it were to be included.

Apart from Aggl. Single, KMedoids consistently produced results with the highest percentage change from optimum, both with approximation on and off. At most the percentage change was 39.18%. The lowest percentage change, 1.75%, was produced by KMeans without approximation on the smallest instance, *pr76.* Aggl. Avg. with approximation had the overall second least percentage change, 2.39%, on the *kroC100* instance.

Aggl. Avg with approximation produced the best result on all instances except on *pr76* and tsp225, where KMeans with approximation and Aggl. Avg without approximation had the lowest percentage change respectively. On KMeans, the approximation had a bigger impact for the percentage change on instances with fewer points. The approximation had an impact on Aggl. Avg on every instance except *tsp225.*

A tendency of the Aggl. Avg. with approximation is that for the instances of up to 280 points inclusive, the percentage change span is 2.39-7.95%, and for instances above 280 points the percentage span is 9.98-15.93%.

The algorithm performed the worst overall on instance *rl5934* with a percentage change of 17.40% at best. The algorithm does better at instance *d18512* (containing 18512 points), with a percentage change score of 11.79% at best, than all the other instances with input sizes over 280 points. Comparatively the best result for instance *pcb442* (with 442 points) was 12.43%.

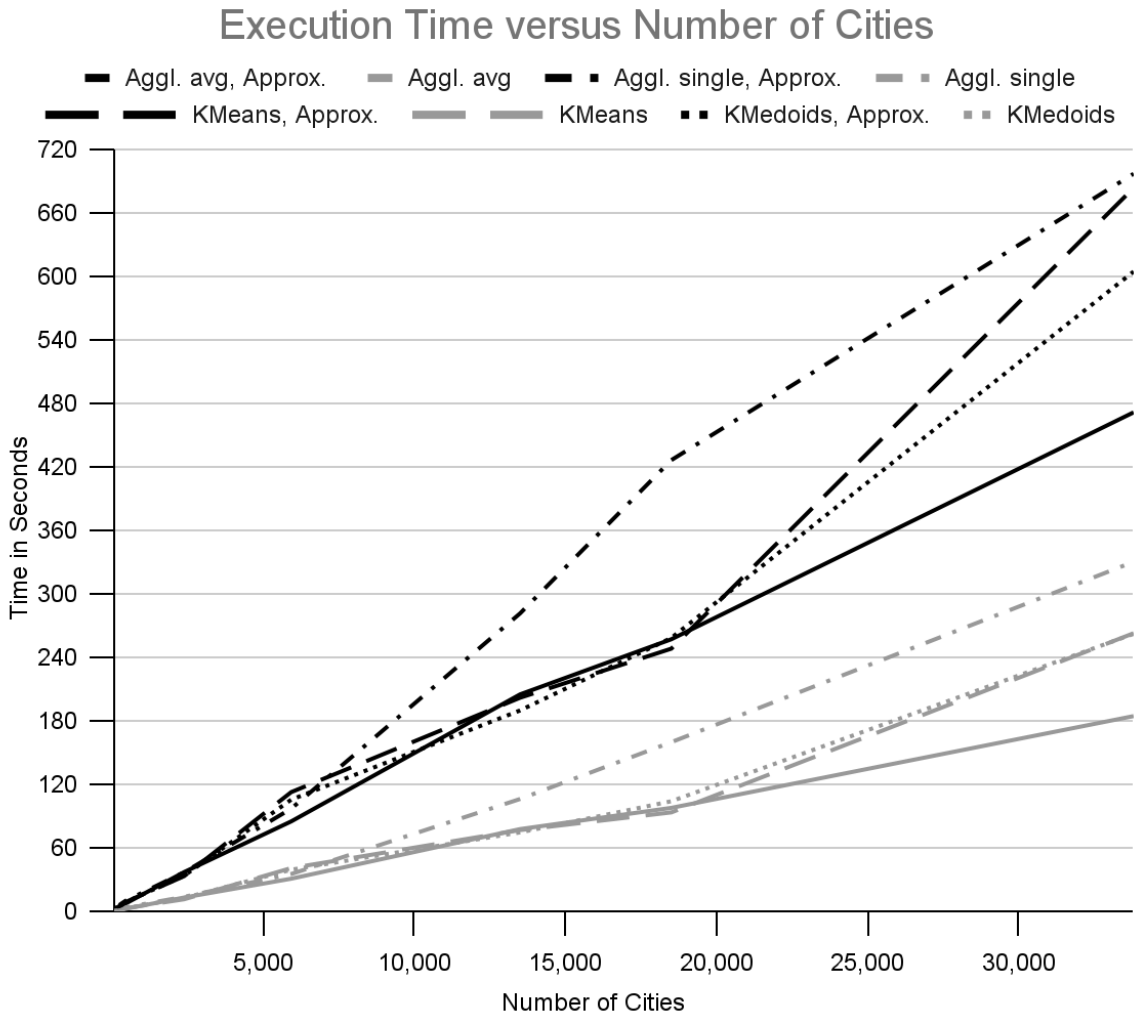

Figure 12. Shows time taken for computing every instance when using 8 clusters with different configurations.

Both with approximation on and off, Aggl. Avg has the lowest execution time in most cases, when compared to the other clustering methods, especially when applied to instances with more cities. Comparatively, Aggl. Single has the highest execution times in both groups in all large instances.

As indicated by the black lines in the chart, the execution time is always higher when the approximation is on.

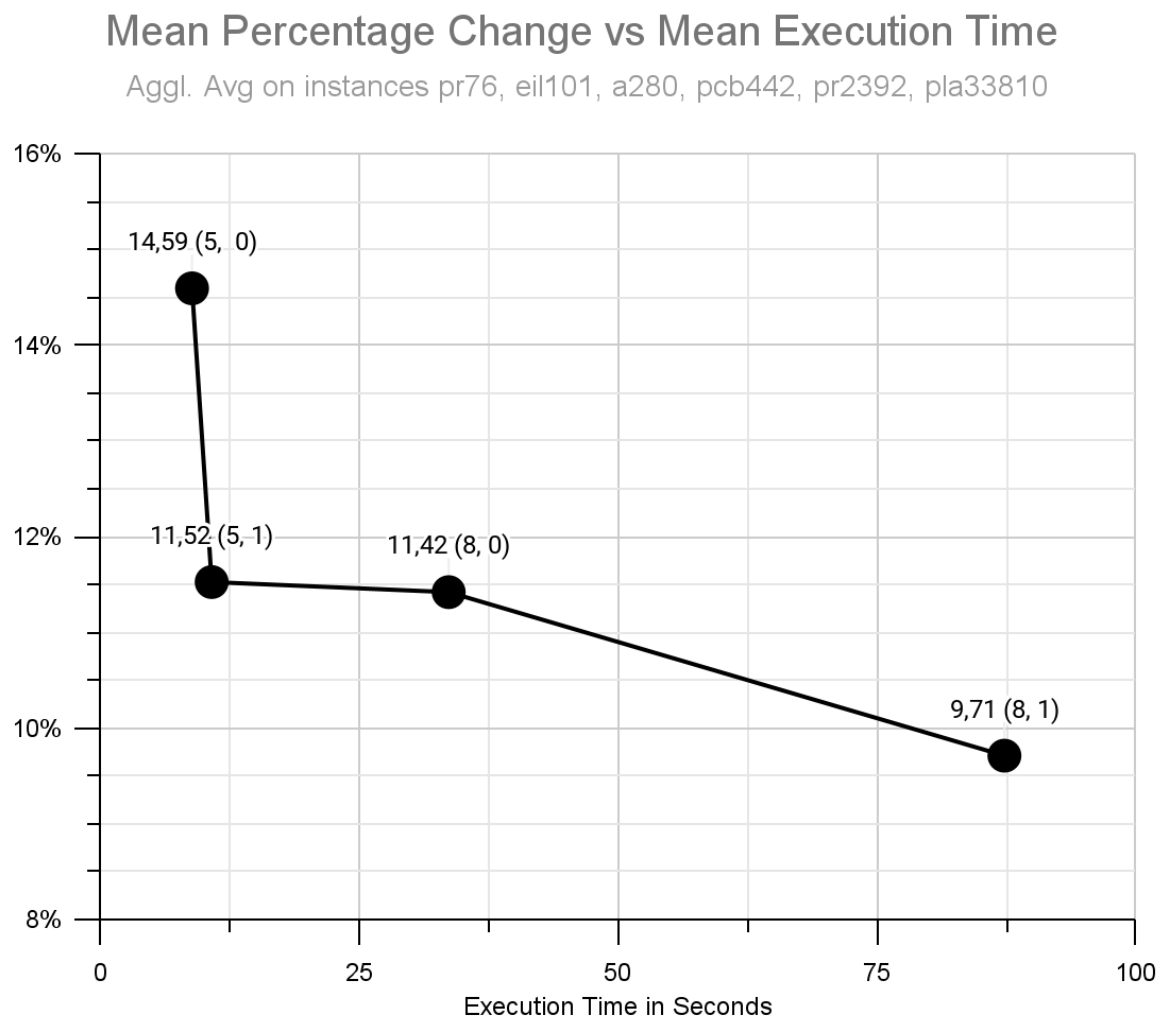

Figure 13. The different points on the chart represent different configurations of the algorithm. The part in brackets meaning (Number of clusters, Approximation(1=On, 0=Off))

The results of the executions of different configurations using Aggl. Avg. are aggregated in figure 13 by getting the mean value from the results of the configurations applied to the instances *pr76, eil101, a280, pcb442, pr2392* and *pla33810.*
The configurations are annotated in the data label inside the parentheses in the chart.

In comparing 5 clusters without approximation (5, 0), and 5 clusters with approximation (5, 1), the closeness to optimum differs by close to 3% whilst the execution time is only slightly affected. Using 5 clusters and approximation (5, 1) gives a similar percentage change result as using 8 clusters without approximation (8, 0), while saving computation times. 8 clusters with approximation produces results with the lowest percentage change at a great cost in increased execution time.

6 Discussion and Conclusions

6.1 Discussion

In this study an algorithm was created that subdivides the TSP problem using clustering and a new heuristic solution for choosing endpoints, merging the subsolutions together. Mariescu-Istodor and Fränti (2021) mentions that choosing a merging strategy that joins open-loop cluster solutions together is an open question and the approximation heuristic used in this study is an attempt at approaching that problem. The authors also suggest that because a divide-and-conquer approach might lead to suboptimal choices by the algorithm, using other optimizing operators such as k-opt as an extra step on the overall solution might improve it further (ibid.).

The experiment showed that the approximation step seems to improve the total solution by shortening the total closed-loop solution by a few percent on most instances. This was the case for most of the clustering methods, having the least effect on KMeans clusters and the largest on Hierarchical Agglomerative clusters, using both average and single-linkage as inter-cluster distance metrics. The result shows that this heuristic could be used with divide-and-conquer solutions and might improve the result; this heuristic could be used in conjunction with further optimization steps, like k-opts, as shown by earlier research, e.g. Sengupta et al. (2019) is using such heuristics in improving on the solution.

The experiments also indicate that the chosen clustering method has a bigger impact on the results than approximation, as seen in table 2. Therefore, the choice of clustering method used by the algorithm is of great importance for the overall solution.

Agglomerative Hierarchical clustering with an average inte-rcluster distance metric, was the overall best performing clustering method applied concerning the generated solutions' closeness to the optimal path. Using KMeans gave the best result on a specific problem instance with only 1.75% percentage change from the optimal solution, but was subpar compared to the best performing clustering method on all other instances. KMedoids performed worse than the other clustering methods except single-linkage, comparing both execution times and percentage change. Mariescu-Istodor and Fränti (2021) discusses the usage of single linkage as a possible means for attaining optimal results. Taking this into account, the method was implemented in the ARC algorithm. As it turned out, the results indicate that the specific implementation of single linkage provided the weakest results in terms of performance among the applied clustering methods.

The algorithm appears to have a linear time complexity, see figure 12. However, this cannot be concluded on the basis of the relatively small sample data utilized in the context of the study. Additional empirical measurements taken on the algorithm's performance, when applying a larger set of TSP instances with a more consistent spread of input sizes, would possibly strengthen such a generalizing claim. Another method of analyzing the time complexity is by a thorough investigation of the code in the implementation to conclude the time complexity theoretically.

6.2 Conclusions

The choice of clustering method for the algorithm greatly affects the overall solution of the algorithm and the best clustering method found in this study for this task was Hierarchical Agglomerative clustering using an average inter-cluster distance metric.

The approximation heuristic improves the overall solutions for all tested clustering methods.

6.3 Limitations

Time and space complexity of the algorithm was not theoretically established because of time constraints.

Extensive test runs performed prior to the experiment indicated that when running the algorithm multiple times on a TSP instance, the results and the time expenditure were consistently similar, thus, a high reliability was assumed. However, applying KMeans and KMedoids, this can not be concluded and further runs on the tests would be needed to ensure consistent results using those clustering algorithms, ensuring reliability. This was not done because of time constraints.

6.4 Ethical and Societal Impact

Future societal and ethical repercussions of this study, if the algorithm is used for certain purposes such as military use or other uses, may in second hand impact or harm the environment, people or animals. On the other hand, the algorithm may have a positive impact in such ways as aiding efficient use of resources, calculating shorter routes for companies or other possible optimisations whereby the TSP problem is a subproblem.

6.5 Further Research

The implementation of the algorithm could be further improved by optimizing the data structures and flow of data through the functions of the algorithm. This would improve the overall execution times when using the approximation. Further optimisations could also be explored.

The differing results of the algorithm on different instances of TSP might be due to the topology of the instances and how well they can be subdivided into clusters using the clustering methods tested in this study. This would need further research to investigate the connection between clustering method, topology of the points and the resulting solution.

Other possible questions that have come up during this study is however another clustering method, like HDBSCAN, could be used to improve the solutions further. HDBSCAN was not used in this study because the amount of clusters created by the clustering method cannot be predetermined. A solution for using it with this algorithm is to use it only to cluster the initial input and solve the first coarse closed-loop recursively instead. This might give a better division of the problem and create better overall solutions but further research is needed.

Another possibility for improvement is the use of cluster medoids instead of centroids. This could improve the coarse open-, and closed-loop solutions by being better representations of the centers of the clusters. Further research is needed to know for sure.

Bibliography

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research*, *6*(6), 791-812.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, *6*(1), 80-91.

Deng, Y., Liu, Y., & Zhou, D. (2015). An improved genetic algorithm with initial population strategy for symmetric TSP. *Mathematical Problems in Engineering*, *2015*.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, No. 34, pp. 226-231).

Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., & Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management (JIEM)*, *9*(2), 374-388.

Gendreau, M., Hertz, A., & Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, *40*(6), 1086-1094.

Goldreich, O. (2010). *P, NP, and NP-Completeness: The basics of computational complexity*. Cambridge University Press.

Gutin, G., & Punnen, A. P. (Eds.). (2006). *The traveling salesman problem and its variations* (Vol. 12). Springer Science & Business Media.

Hjerm, M., Lindgren, S., & Nilsson, M. (2014). *Introduktion till samhällsvetenskaplig analys*. Gleerups utbildning AB.

Johannesson, P., & Perjons, E. (2014). *An introduction to design science* (Vol. 10, pp. 978-3). Cham: Springer.

Karp, R. M. (1986). Combinatorics, complexity, and randomness. *Communications of the ACM*, *29*(2), 98-109.

Kobayashi, K. (1998, September). Introducing a clustering technique into recurrent neural networks for solving large-scale traveling salesman problems. In *International Conference on Artificial Neural Networks* (pp. 935-940). Springer, London.

Korakakis, M., Spyrou, E., Mylonas, P., & Perantonis, S. J. (2017). Exploiting social media information toward a context-aware recommendation system. *Social Network Analysis and Mining*, *7*(1), 1-20.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, *21*(2), 498-516.

Mariescu-Istodor, R., & Fränti, P. (2021). Solving the Large-Scale TSP Problem in 1 h: Santa Claus Challenge 2020. *Frontiers in Robotics and AI*, *8*.

Little, J. D., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations research*, *11*(6), 972-989.

Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., & Parthiban, P. (2009). Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. *International Journal of Nonlinear Science*, *8*(4), 480-487.

Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc..

Reinelt, G. (2018). Traveling Salesman Problem - TSPLIB. Retrieved May 19, 2022, from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

Sengupta, L., Mariescu-Istodor, R., & Fränti, P. (2019). Which local search operator works best for the open-loop TSP?. *Applied Sciences*, *9*(19), 3985.

Strutz, T. (2021). Travelling Santa Problem: Optimization of a Million-Households Tour within One Hour. *Frontiers in Robotics and AI*, *8*.

Sriraman, A., & Bays, M. J. (2014, September). Efficient reacquire and identify path planning over large areas. In *2014 Oceans-St. John's* (pp. 1-7). IEEE.

Toth, P., & Vigo, D. (Eds.). (2002). *The vehicle routing problem*. Society for Industrial and Applied Mathematics.

Vetenskapsrådet. (2017). God forskningssed [ONLINE]. (Reviderad utgåva). Stockholm: Vetenskapsrådet.

Weiss, M. A. (2012). *Data structures and algorithm analysis in Java*. Pearson Education, Inc.

West, D. B. (2001). *Introduction to graph theory* (Vol. 2). Upper Saddle River: Prentice hall.

Wierzchoń, S. T., & Kłopotek, M. A. (2018). *Modern algorithms of cluster analysis* (Vol. 34). Springer International Publishing.