Erasmus University Rotterdam

Erasmus School of Economics

Bachelor Thesis
Econometrics & Operations Research

Lin-Kernighan and a Modified Heuristic for Solving the 2-Cycle Cover Problem

Author
Sara Keetelaar
433250

Supervisor
T. van Breugem MSc

Second Assessor
R. Hoogervorst MSc

July 7, 2019

Abstract

In this thesis the focus lies on the Lin-Kernighan heuristic algorithm for the symmetric Traveling Salesman Problem (Lin & Kernighan, 1973). This heuristic dates from 1973 but is still considered a top-notch heuristic. It has an iterative approach and starts with a feasible solution. It finds better solutions by sequentially exchanging edges. When no more improvements can be found a solution is obtained which is better than the previous. In this study the algorithm is implemented in its most basic form. The heuristic performs well on small instances and finds the optimal solution on the smallest instances we tested on. As in the original implementation, effectiveness decreases with problem size. In this thesis, also a new heuristic is proposed for the 2-CCP. In this problem the goal is to find two disjoint tours of minimum total length. The proposed algorithm starts with a feasible TSP solution and transforms it into a 2-CCP also by a sequential exchange of edges. This algorithm is shown to perform well on small instances as well. The heuristic performs well under the condition that the starting tour is a good solution for the TSP. We conclude that the modified algorithm is an effective method for transforming TSP solutions into 2-CCP solutions.

Contents

1 Introduction **1**

2 Terminology **2**

3 Theoretical Framework **3**
3.1 The TSP . . . 3
3.2 Lin-Kernighan Algorithm . . . 3
3.3 The Cycle Cover Problem . . . 4
3.4 Motivation for Study and Contribution to Literature . . . 5

4 The TSP **6**
4.1 Problem Description . . . 6
4.2 The Lin-Kernighan Algorithm . . . 6

5 The 2-CCP **10**
5.1 Problem Description . . . 10
5.2 The Modified Lin-Kernighan Algorithm . . . 10

6 Computational Results **14**
6.1 Traveling Salesman Problem . . . 14
6.1.1 Data Resources . . . 14
6.1.2 Computational Results . . . 14
6.2 2-CCP . . . 16
6.2.1 Data Resources . . . 16
6.2.2 Computational Results . . . 17

7 Conclusion and Recommendations **20**

A Appendix **25**

1 Introduction

The Traveling Salesman (TSP) is one of the most widely-studied problems in combinatorial optimization. Given a set of n cities, the problem consists of finding the shortest route which visits every city exactly once. In the symmetric TSP, distances between each pair of cities are equal in opposite directions. Moreover, in the Euclidean TSP, coordinates of the cities are given and edge distances are calculated as Euclidean distances between the cities. This specific problem class is the problem that is studied in this thesis.

The TSP has many applications, obviously in routing problems but also other real-world problems can be reduced to a TSP. For instance the drilling of printed circuit boards, computer wiring and robot control can be modeled as a TSP (Helsgaun, 2000). Furthermore, other problem classes can be seen as extensions of the TSP. For example the well-known Vehicle Routing Problem (VRP) can be modeled as a TSP with side constraints as vehicle capacity and customer demands.

The TSP belongs to the class of NP-hard problems, which means it can not be solved in polynomial time. Exact algorithms do exist, but can only solve problems up to a certain size. To find solutions for large instances in reasonable time, heuristics are necessary.

In 1973, Lin and Kernighan proposed a heuristic algorithm for the symmetric TSP which is still considered a top-notch heuristic. This particular algorithm is investigated in this study. When this heuristic was introduced, little research had been done on instances with more than 60 cities and only optimal solutions were known for problem instances up to 57 cities. This heuristic was highly effective. Effectiveness however decreased with problem size but with some improvements, as made in Helsgaun (2000) and Helsgaun (2017), the heuristic finds optimal solutions for large instances.

A problem that is closely related to the TSP, is the m-Cycle Cover Problem (m-CCP). In this problem m disjoint cycles must be found which together cover the entire graph. This problem is a direct generalization of the standard TSP since the m-CCP coincides with the TSP for $m = 1$. This problem has first been described in Sahni and Gonzales (1976). Applications of this problems can also be found mostly in routing problems, as the cycles can be seen as vehicle routes. In this study, this problem is further investigated, and the focus lies on the specific case of $m = 2$. This problem is also NP-hard and optimal solutions can thus not be found for large instances.

The aim of this study is first to implement the basic Lin-Kernighan algorithm and test it on symmetric TSP-instances. Second, we propose a Lin-Kernighan-based algorithm for the 2-CCP. We will compare the results to some benchmark solutions and reflect on the results.

The remainder of this thesis is organized as follows. First in Section 2, some terminology used in this study is explained. In Section 3, the theoretical background is given. Hereby first the TSP is covered, second we elaborate on the Lin-Kernighan algorithm, and finally the 2-CCP is discussed. The methodology is divided in two parts. The first part is given in Section 4, and gives a detailed description of the basic Lin-Kernighan algorithm. Hereby differences between this implementation and the original implementation are discussed. Section 5 contains a a precise problem description of the 2-CCP, with an explanation of the algorithm we propose, and in particular differences with the standard algorithm are described. Moreover in Section 6, computational results are presented and discussed and finally in Section 7, a conclusion of this research and some limitations are described. Also recommendations for further research are given.

2 Terminology

In this section some terminology that is used in this thesis is explained. Some basic knowledge of graph theory is hereby assumed.

Complete Undirected Graph The TSP is defined on a so-called complete undirected graph, also simply called a complete graph. A complete graph is a graph in which each pair of nodes is connected by exactly one edge (Wilson, 1985).

Node degree The degree of a node is defined as the number of edges incident to this node (Wilson, 1985).

Circuit A circuit on a graph is a path that begins and end at the same node (Rosenkrantz et al., 1977).

Tour A tour for a TSP graph G is a circuit on a graph which visits each node exactly once (Rosenkrantz et al., 1977). In this study, a tour often refers to a Hamiltonian Cycle.

Subtour A subtour is a tour on a subset S of V, where V denotes the set of all nodes on the graph (Rosenkrantz et al., 1977).

Cycle A cycle is a circuit which does not repeat nodes. This is equivalent to a tour for the TSP as described above. A simple cycle does not have to contain all nodes of a graph (Bender & Williamson, 2005).

Hamiltonian Cycle A Hamiltonian cycle is a cycle which contains all nodes in the graph. A Hamiltonian Cycle is by definition a solution of the TSP (Rahman & Kaykobad, 2005).

Hamiltonian Path A Hamiltonian path is a path which containes all nodes exactly once. The difference with a Hamiltonian Cycle is that the path does not return to its starting point, that is, the path is not closed (Rahman & Kaykobad, 2005).

Node-disjoint Cycles In this thesis, the term node-disjoint cycles or simply disjoint cycles is used for cycles in the same instance which have no nodes, and thus no edges, in common.

Cycle Cover If $C_1, ..., C_k$ is a set of node-disjoint cycles such that $V(C_1) \cup ... \cup V(C_m) = V$, where V denotes the set of all nodes in the graph. That is, if there is a set of k cycles which do not have any nodes in common, and if all cycles together contain all nodes of the graph, the set $C = C_1, ..., C_k$ is called a k-size cycle cover of a graph (Khachay & Neznakhina, 2014).

3 Theoretical Framework

In this section the theoretical framework of the problems of interest is given. First, in Section 3.1, the background of the TSP is given. In Section 3.2 the theoretical framework of the Lin-Kernighan algorithm is described. In Section 3.3, the background of the 2-CCP is given and finally in Section 3.4, the contribution to the existing literature is explained.

3.1 The TSP

The TSP is a problem that has been studied for a long time. In the 1800's, Hamilton and Kirkman designed a puzzle where a Hamiltonian path had to be found, this is where the term Hamiltonian originates from (Biggs et al., 1967). The first publication of the actual TSP came a centure later by Karl Menger (1932), who called it "The Messenger Problem".

Dantzig et al. (1954) first defined the TSP as an integer linear programming model. Also they brought up cutting planes methods for solving the TSP for the first time. This is an exact algorithm for the TSP, which can solve the problem, but only for a specific class of problems. Also other exact algorithms have been introduced. For instance, Held and Karp (1962) proposed an algorithm using dynamic programming, which used a man-machine approach. This algorithm gave appropriate solutions but took much time to solve. Moreover, another way of solving the TSP is by using branch-and-bound methods. Applegate et al. (2006) solved a 85.900-city TSP using this type of algorithm, which is nowadays known as the largest instance solved to optimality. Still this is considered the fastest method for exactly solving large instances of the the TSP.

Heuristic methods have been proposed over time for finding (near)optimal solutions for large instances. Hereby, a distinction can be made between constructive and iterative heuristics. A constructive heuristic starts with an empty solution, while an iterative heuristic starts with a feasible solution and moves to better solutions in the solution space. A simple constructive algorithm is the nearest neighbour heuristic. This algorithm starts from a city and chooses the nearest city which is not yet contained in the tour as the next city (Rosenkrantz et al., 1977). The Lin-Kernighan heuristic is an iterative heuristic. In the following section we will elaborate on this algorithm.

3.2 Lin-Kernighan Algorithm

The Lin-Kernighan algorithm was first introduced in 1973, when not so many heuristic algorithms for the TSP were designed yet. The Lin-Kernighan algorithm is based on the interchange algorithm of Croes (1958), which consists of a sequential exchange of two edges in a tour to obtain a better solution. This is known as the 2-opt algorithm. This algorithm was only considerably successful. The Lin-Kernighan algorithm is a generalization of this idea, as it is a so-called k-opt algorithm, in which k edges are switched, with k undetermined in advance. This algorithm turned out to be highly effective: it found optimal solutions on all tested instances and it took three minutes to obtain the optimum with 95% confidence on a 100-city instance. Remarkable is the fact that only optimal solutions up to 57 cities were known. For larger instances, a random distance matrix was generated, and empirical proof for occurrence of the optimum was given. Also for the larger instances, the optimal value was reached. However, for smaller instances the optimum occurred

in 100 % of the trials, where for instances with 100 cities, this probability dropped to 20-30%. The largest instance the algorithm was applied to was a 318-city instance. This problem was first split into three different subproblems, because of computer storage-limitations, and afterwards the resulting tours were joined (Helsgaun, 2000).

Helsgaun (2000) proposed a more effective implementation of the original algorithm. In this work also the 318-city instance from Lin and Kernighan (1973) was tested, of which the solution was actually known by then, and he concluded that the solution found by Lin and Kernighan (1973) was only 1,5% above optimum. This result confirmed the fact that the original algorithm was highly effective already. However, while in the original algorithm for instances of size 100 the optimal solution occurred only in 20% of the cases, the improved implementation gave optimal solutions in 70% of all trials for an instance of 7374 cities. Also, for all instances with known optimal solution, the optimal solution was found and for some larger instances, an even better solution was found than the best known solution so far. This implementation is feasible for problems with fewer than 100.000 cities. The largest known TSP instance is the so-called world TSP [1], which consists of 1.904.711 cities. The best solution so far is found by Helsgaun (2017), using a highly improved algorithm, the LKH-3 (Helsgaun, 2017). This shows that the algorithm performs well compared to other heuristic algorithms, also on very large instances.

The original algorithm, and also the improved implementations are specifically feasible for the symmetric TSP. However, some Lin-Kernighan-based algorithms have been introduced for related combinatorial optimization problems. For instance, in 1980 an extension of the algorithm was presented by Kanellakis and Papadimitriou. In this work an algorithm was introduced which gave feasible solutions for the asymmetric TSP for large instances (up to 90 cities). Also, Karapetyan and Gutin (2011) applied a modified algorithm to the generalized TSP, in which all determined subsets of cities have to be visited exactly once. This algorithm has been shown to be the best performing heuristic for the generalized TSP on smaller instances. Notable results were presented in Helsgaun (2017). The LKH-3 implementation presented in this work, was applied to all sorts of TSP-related problems. This research showed that the algorithm is also suitable for generalizations and extensions of the Traveling Salesman Problem. Hereby, contrary to Kanellakis and Papadimitriou (1980) and Karapetyan and Gutin (2011), no adjustments were made with respect to the standard algorithm. Instead, the problems were solved by transforming the problem instances into instances of the symmetric TSP. For most types of problems best known solutions were obtained and sometimes even new best solutions were found. However, the algorithm has not yet been applied to the CCP, which is discussed in the following section.

3.3 The Cycle Cover Problem

The minimum 2-Cycle Cover Problem consists of finding two disjoint cycles of least-cost, which together cover all cities of the graph. This problem is not a very well-known problem but some modifications and special cases of the problem class have been studied. For instance, a well-known variant is the L-cycle cover, which limits the length of the cycles to be in some set L (Manthey, 2008). The Cycle Cover Problem with maximum distance constraint is also a well studied topic. Here the goal is to find the minimum number of cycles with total cost smaller than λ (Yu, Liu

[1] retrieved from http://www.math.uwaterloo.ca/tsp/world/

& Bao, 2019). In this problem, the number of cycles is the variable and the cost is fixed, as opposed to the variant on which the focus lies in this thesis. An approximation algorithm for the Euclidean min-2-SCCP, the minimum 2-size Cycle Cover Problem as we also consider, is introduced by Khachay and Neznakhina (2014). Here an approximation algorithm is defined as a polynomial time algorithm that for all instances of a problem produces a solution whose value is within a certain factor α of the value of an optimal solution (Williamson & Shmoys, 2011). This means that such an algorithm guarantees a certain performance, which is not the case for heuristic algorithms. An asymptotically optimal solution is obtained in this paper. In this research however, the problem is defined on a complete weighted directed graph, in contrast to the undirected graph that is considered in the problem we study.

In Gimadi and Rykov (2016), the m-Cycle Cover Problem is studied. In this problem the goal is to cover a complete undirected graph in m disjoint cycles. This is exactly the problem of interest. In this work, a so-called TSP-approach is used to construct an approximation algorithm for the m-CCP. The basic idea of this approach is that a Hamiltonian cycle, obtained from a heuristic algorithm for the TSP, is used as starting point and is divided in m non-adjacent segments. The edges between the segments are removed and closing edges are added. This approximation is asymptotically optimal for the number of cycles $m = o(n)$ (with n the total number of cities).

In Bläser et al. (2006) it is shown that cycle covers can be used for solving Traveling Salesman Problems, and also in Sunil Chandran and Shankar Ram (2006), research has been done on the relation between the TSP and the CCP. In this first-mentioned paper, first two disjoint cycles are constructed and from there on the optimal solution for the Traveling Salesman Problem can be approximated. The research focuses on the asymmetric TSP, but it is also mentioned that there is an appropriate connection between the Cycle Covers and the symmetric TSP. It can thus be concluded that CCP solutions can be successfully obtained from TSP solutions and vice versa.

3.4 Motivation for Study and Contribution to Literature

The Lin-Kernighan algorithm has been shown to be a good approach for solving not only the symmetric TSP, but also related problems. Also the heuristic seems to be suitable to modify such that it is feasible for related problems. The Lin-Kernighan algorithm has however not been applied yet to the 2-CCP, which is such a TSP-related problem.

Moreover, for the 2-CCP in particular approximation algorithm exist, but no heuristic methods have been designed. An advantage of approximation algorithms is that they give a guaranteed performance. However, a downside is that these algorithms often do not give much information on computational results. This is why heuristic methods are also desired.

Furthermore, it has been shown that TSP solutions can be successfully used as a starting point for finding CCP solutions (Gimadi & Rykov, 2016). This is why in this thesis we present a Lin-Kernighan-based heuristic algorithm for the 2-CCP which uses TSP solutions as starting point and gives computational results.

4 The TSP

In this section the TSP is further discussed. First an exact problem description is given in Section 4.1. Furthermore, a description of the Lin-Kernighan algorithm is given in Section 4.2. Also the differences between this implementation of the heuristic and the original heuristic algorithm are described.

4.1 Problem Description

The TSP consists of finding a least-cost Hamiltonian cycle on a given graph. The cost of an edge is here defined as the distance between the two endpoints. In this thesis, the Euclidean case is the problem of interest, which means the edge costs are calculated as Euclidean distances between the cities. By definition the Euclidean TSP is also symmetric. The Lin-Kernighan algorithm is specifically feasible for the symmetric TSP and therefore also for this problem.

4.2 The Lin-Kernighan Algorithm

The algorithm is directly derived from Lin and Kernighan (1973), although some of the refinements added in their work are not included in this implementation. In this section, first an intuitive description of the heuristic is given. Second an example is given to illustrate the working of the algorithm and finally a detailed step-by-step explanation of the algorithm is given.

The algorithm starts with a random feasible tour T. Any node can be chosen as starting point with adjacent edge x_1. From the other endpoint of x_1, an edge y_1 which is not in the initial tour, is chosen such that the gain of exchanging these edges is positive. From this point on, edges for x_i and y_i are sequentially chosen, where the x_i edges are contained in the starting tour and the y_i edges are not. The exchanges are sequential in the sense that in every step x_i and y_i share an endpoint, and y_i and x_{i+1} again share an endpoint. In every step, the cumulative gain of the exchanges must be positive. Also, important is that the tour must be feasible at all times. This means that there must always be a possibility to close-up the tour. The closing-up check is an important step of the algorithm. After choosing x_i in every check, a close-up edge y^*, from the endpoint of x_i to the starting point is added. After choosing y_i there is only one option for x_{i+1} which can give a feasible close-up. This fact is further explained below. If the tour, obtained by the closing-up check forms a better solution than the previously best solution, this is the best solution so far. After each close-up check, the algorithm continues with choosing edge y_i (note hereby that the close-up edge is only a check and is not actually added).

An illustration of the algorithm and close-up check is given in Figure 1.

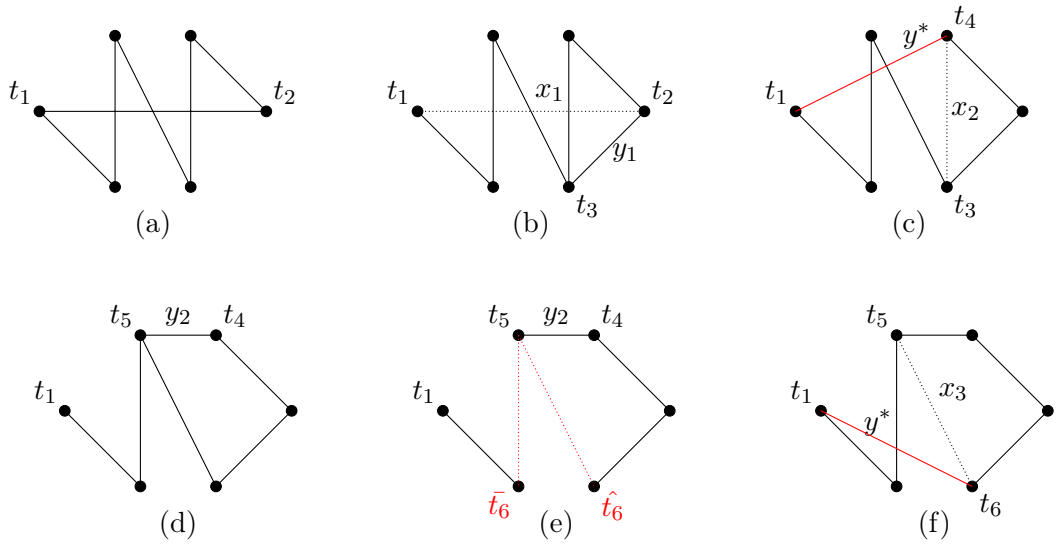


Figure 1: The Lin-Kernighan Algorithm

In Figure 1(a) the initial tour is given. In (b), the x_1 and y_1 with positive gain are chosen. Next, in (c) the unique x_2 is chosen and the close-up check is performed. In this case the total gain is positive, so this tour is the best tour so far. The algorithm continues in (d), in which y_2 is chosen which satisfies the criteria. In (e) the two options for x_i are considered. We can observe that node $\overline{t_6}$ lies between t_1 and t_5, and if this edge would be removed, the path is broken. However if $\hat{t_6}$ is chosen, and corresponding x_3 is removed, there still is a path. Finally in (f) the close-up which gives a feasible tour is shown.

On the Uniqueness of x_i

After removing x_1 from the starting tour, which is a Hamiltonian Cycle, one edge is removed. This means a Hamiltonian path is created. If then an edge y_1 is added, which does obviously not close up the tour since this is only possible by placing edge x_1 back and by definition x_i and y_i are disjoint, a subtour is created, as displayed in Figure 1(b). This is the point where the decision on x_2 has to be made. Then, because of the subtour that is created, by definition one of the candidate edges is contained in this subtour and the other candidate is not. If the edge that is contained in the subtour is removed, there is no subtour anymore and again a Hamiltonian path is created, as in Figure 1(c) where the choice for x_2 is displayed. By definition of a Hamiltonian path, this can be changed into a Hamiltonian cycle only by adding a close-up edge. We can thus conclude that removing the candidate edge that is contained in the subtour created by adding y_1 can give a feasible close-up. This is thus the edge for x_2 that must be chosen. The other candidate edge was not contained in the subtour, which means that when breaking this edge, the subtour still exists and is disconnected from the remaining edge. This can not give a feasible close-up.

After removing x_2 again a Hamiltonian path is created, and the edge y_2 added after again creates a subtour. When we must decide on x_3 we have arrived in the same situation as when the decision on x_2 had to be made. This implies that after adding every y_{i-1}, a subtour is created and there is a unique option for x_i which gives a feasible close-up.

Exact Algorithm Description

The exact description of the algorithm is given here. First some notation is introduced.

- The length of edges x_i and y_i is denoted respectively by $|x_i|$ and $|y_i|$.
- The gain of exchange i is denoted by g_i and can be calculated as $g_i = |x_i| - |y_i|$.
- The cumulative gain is denoted by G_i and is calculated as $\sum_{j=1}^{i} g_i$.
- The best gain so far is denoted by G^*.

An exact description of the algorithm follows below.

1. Start with initial tour T.
   Define G^* the best improvement so far and set $G^* = 0$.

2. Choose starting point t_1, adjacent edge x_1 and take t_2 the other endpoint of x_1.
   Choose y_1, which is not in the initial tour, such that the gain of this exchange g_1 is positive.

3. Choose x_i and y_i sequentially:
   Pick x_i, connecting nodes t_{2i-1} and t_{2i} such that

   (a) it is not an edge previously joined (disjunction criterion).

   (b) if a closing-up edge connecting t_{2i} and t_1 is added, a tour is created (feasibility criterion).

   Choose y_i, based on nearest neighbours, connecting nodes t_{2i} and t_{2i+1} such that

   (a) it is not an edge previously broken (disjunction criterion).

   (b) the total gain $G_i = \sum_{j=1}^{i} g_j > 0$ (gain criterion).

   (c) this edge permits the breaking of an adjacent edge x_{i+1} (feasibility criterion).

4. In every iteration, a check for closing-up the tour is performed. After choosing x_i, the closing-up edge y^*, with corresponding gain $g^* = |x_i| - |y^*|$, is considered.
   If the total gain $G_{i-1} + g^*$ is higher than the best gain so far G^*, this is considered the best tour so far. Update the variables:
   – $G^* = G_{i-1} + g^*$
   – $k = i$

5. The construction of x_i and y_i continues until either no more x_i or y_i satisfy the constraints or if $Gi \leq G^*$.

   (a) If $G^* > 0$, take the best tour found by removing all $X = \{x_1, ..., x_k\}$ edges and adding all $Y = \{y_1, ..., y_k\}$ edges and the best close-up found. This tour is the starting tour for the next iteration.

   (b) If $G^* = 0$, no better tour can be found, the algorithm terminates here.

We define a LK iteration, also simply an iteration, as one exchange of the two sets of edges $X = \{x_1, ..., x_k\}$ and $Y = \{y_1, ..., y_k\}$. The value of k is determined according to after how many exchanges of x_i and y_i one of the stopping criteria occurs. After exchanging these edges, that is removing X and adding Y, a new tour is created at lower cost than the previous tour. If no tour of lower cost can be found, the algorithm finally terminates. However, in Lin and Kernighan (1973), when we arrive at this point of termination, a backtracking mechanism is invoked.

Backtracking

In Lin and Kernighan (1973), backtracking on the first two levels is implemented. This backtracking mechanism is invoked when no improvement can be found on a certain tour (Step 5(b) in the algorithm). The backtracking is only used on the first two levels. This means that if for the choice made for y_2, based on nearest neighbours in increasing order, other options are considered. If none of these options yields any gain, the algorithm returns to the point where x_2 is chosen. Here for once a breaking of the feasibility criterion is permitted. If this also does not lead to any improvement, the algorithm returns to the first level. Other options, again based on nearest neighbours, for y_1 are investigated. If this again does give a total gain of zero, the algorithm finally goes back to the point where x_1 is chosen. Other starting edges are considered. If none of these options eventually yields any gain, the procedure is terminated.

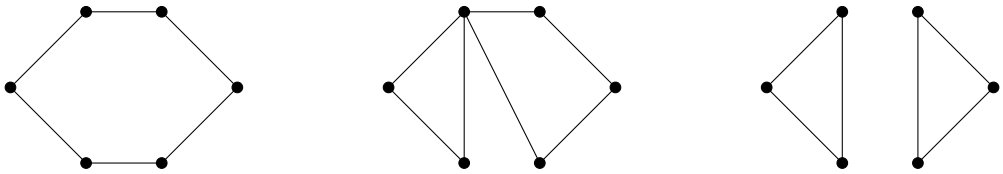
Backtracking on all levels would give an exact solution, however, a trade-off between running time and effectiveness must be made. Lin and Kernighan (1973) decided that backtracking only on the first two levels was the best trade-off. In Helsgaun (2000), no backtracking is implemented but this algorithm is still considered highly effective. To decrease running time and given Helsgaun's results, no backtracking is implemented in this implementation.

5 The 2-CCP

In this section the problem description of the 2-CCP is given in Section 5.1. Also in Section 5.2 a new heuristic algorithm for the 2-CCP is introduced, which is based on the Lin-Kernighan heuristic.

5.1 Problem Description

The 2-CCP consists of finding two disjoint cycles of minimum total length. We consider the problem defined on Euclidean instances, which means the problem can be considered a symmetric CCP. The similarities with the TSP are easy to see. More specifically, in the TSP the goal is to find a single cycle which covers the entire graph, and in the 2-CCP two such cycles must be found. This shows that the 2-CCP is a direct generalization of the TSP. The 2-CCP can also be easily linked to the 2-TSP, which is another generalization of the TSP. In this problem two tours of minimum length must be found, and each city must be contained in exactly one tour. The only node the two routes have in common is the depot node. This is the only difference with the 2-CCP, in which no depot node is involved. An illustration of solutions for the TSP, the 2-TSP and the 2-CCP is given below, to emphasize the differences between the three problems.



TSP 2-TSP 2-CCP

As we see, the TSP solution consists of a single tour which visits all nodes. The 2-TSP solution consists of two tours which together visit all nodes and have a single node in common. The 2-CCP solution finally consists of two tours which together cover all nodes, and the tours are completely disjoint, that is, each node is contained in exactly one tour.

5.2 The Modified Lin-Kernighan Algorithm

We construct a Lin-Kernighan-based algorithm for the 2-CCP using TSP solutions as starting point. This algorithm needs to give two disjoint cycles as an outcome instead of the single cycle in the original algorithm. Herefore we have to make an adjustment with respect to the original algorithm. We use the observations found in the closing-up step of the standard Lin-Kernighan algorithm. This is the main step we modify to make the algorithm feasible for the 2-CCP.

In the algorithm, after choosing y_i there are, as we have elaborated on earlier, two options for x_{i+1}, of which only one can give a feasible close-up for the TSP. We on the other hand take a closer look at the other "infeasible" option. If this edge, which connects points t_{2i-1} and point t^* would be removed and corresponding close-up edge y^*, which connects points t^* and t_1, would be added, we can observe that two separate tours are created. An illustration is given in Figure 2.

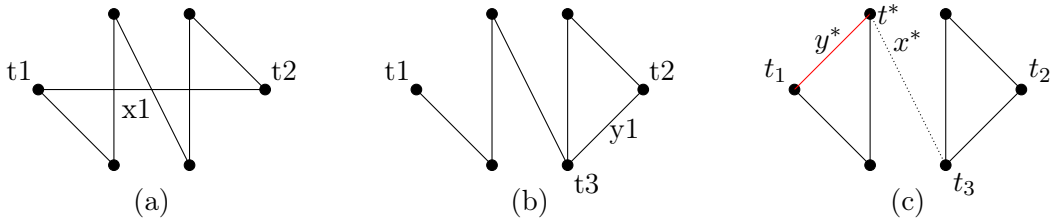


Figure 2: The Modified Algorithm

In Figure 2(a), the initial tour is given with corresponding x_1. In (b), the choice of y_1 is made equivalently to the original algorithm. In (c) the difference between the two algorithms is displayed. Where we would first choose x_2 and try the close-up afterwards, we now perform the close-up check by choosing the other candidate edge for x_i: x^*, with corresponding y^*. It can be observed that two disjoint tours are created. An intuitive proof of this fact is given at the end of this section. The algorithm continues as the standard algorithm, and chooses the regular choice for x_2. After each choice for y_i, this alternative close-up check is performed. If this closed-up solution gives a better solution than the previously found solution, this is considered the best solution so far. Because of the gain criterion, with each exchange the total gain must still be positive. This is also the case in the extended algorithm, which guarantees the solution obtained after a LK iteration to be better than the initial solution.

If we follow the exact same steps as the original algorithm, we might reach a point at which a feasible close-up for the 2-CCP is not possible. An example is given in Figure 3.

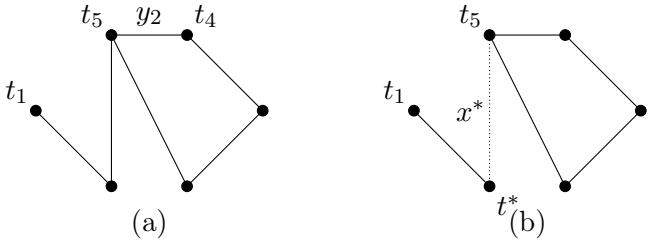


Figure 3: Infeasible Choice

If we would arrive in the next step of the Lin-Kernighan algorithm, the choice for y_2 would be made as in Figure 3(a). This is the exact same step as in Figure 1(d). In Figure 1(e) the two possible options for x_i are displayed and in Figure 1(f) we finally see which edge is chosen. If we however would perform the close-up check for the extended algorithm, we consider removing the other option as viewed in Figure 1(d). As we see in Figure 3(d), the close-up edge y^* would be the edge connecting t^* and t_1, but we see that this edge is already contained in the original tour. Because of the disjunction criterion, this edge can not be chosen as close-up. This is a boundary case, which only occurs when t^* is directly connected to t_1 in the original tour. To prevent this from happening, an extra constraint is laid upon the choice of y_i, which looks forward and checks whether a x^* can be found which gives a feasible close-up for the 2-CCP. In this example, the edge y_2 as in Figure 3(a) is rejected and other options have to be considered. This extra constraint is implemented in the modified algorithm.

In the original algorithm, the new tour obtained by interchanging sets $X = \{x_1, ..., x_k\}$ and $Y = \{y_1, ..., y_k\}$ can easily be used as the new starting tour, as it is perfectly feasible for the TSP. However, the modified algorithm is built on the idea of obtaining a 2-CCP solution from a TSP solution. The resulting configuration, which forms two disjoint tours, can thus not directly be used as starting tour for the next LK iteration. This is why there must be some way of adjusting the solution such that the algorithm can run another iteration. Therefore we propose to merge the two disjoint tours in a certain way, such that this merged tour can be used as the starting point for the next iteration. The precise merging method used is described below.

On the Merging of Tours

In the original algorithm, the resulting tour after a LK iteration can be used directly as new starting point for the next iteration. For the extended algorithm we want to do something similar, but since the algorithm is built on the idea of creating 2-CCP solutions from a TSP solution, this is not directly possible. We therefore want a new starting tour which is as close as possible to the configuration obtained from the previous LK iteration.

Furthermore, the next LK iteration is desired to give a better solution than the previous solution, and we thus prefer to have a merging with positive gain. Combining these observations, we want a merging method with a minimum number of edges changed at lowest possible cost.

The merging method is performed as follows:

1. Consider two edges, one edge from each resulting tour.
2. Remove both edges. There are now two ways of joining the tours by adding two edges between the endpoints of the edges removed in step 2.
3. The cost of this merging method is calculated as the length of the edges to be added minus the length of the edges to be removed. Choose of the two options as described in step 3, the option with least associated cost.
4. Perform step 1-4 for all possible pairs of edges with one edge from each tour.
5. Pick the pair of edges with least associated merging cost. This is the best way of merging the two tours.

For a specific pair of edges this merging method is illustrated in Figure 4.

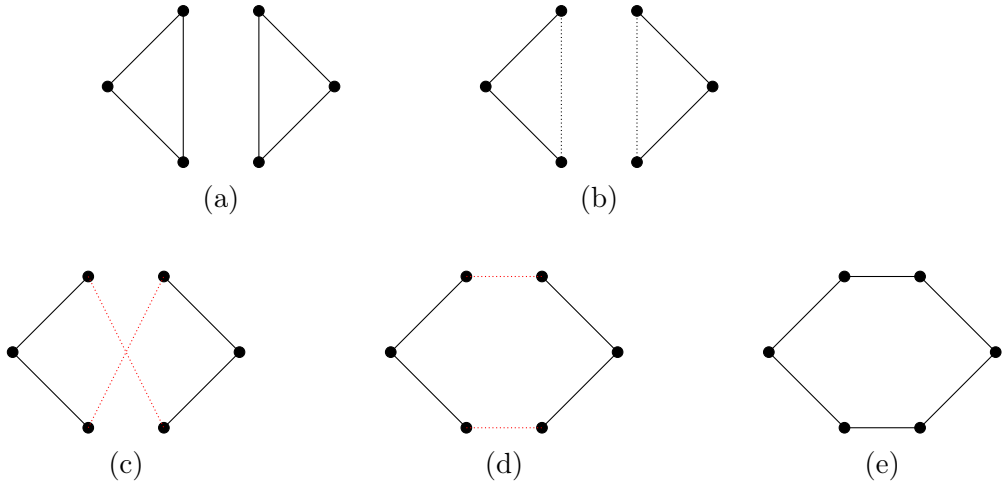


Figure 4: The Merging of Tours

In Figure 4(a) the tours obtained after one iteration are given. All combinations of two edges, one edge from each tour, are considered as removal edges. Such a combination is shown in (b). If these edges are removed, the two ways of joining the tours together are displayed in (c) and (d). The least-cost merge of these two options is chosen in (e). We do this for all possible pairs of edges, an edge of each tour, and finally pick the pair which has the least cost associated with the exchange of edges.

As the solution after the following LK iteration must be better than the previous solution, positive merging is desired. This is the only way the gain criterion can be strictly satisfied. However, the gain criterion then states that an improvement must be found on this merged tour, but eventually the goal is only to find an improvement on the previous solution. If the merging is done at very low (and negative) cost, it might be hard to find an improvement on this tour in the next iteration. The least-cost merging might thus give a starting solution which is too tight. This fact must be taken into account when discussing the computational results.

On the Choice of x^*

We here verify that the edge chosen for x^* leads to a feasible close-up for the 2-CCP. As shown before, when arriving at the point we need to choose x_i, a subtour has been created after adding y_{i-1}. Two options for x_i are available from this point. One of these edges is contained in this subtour and the other edge is not. We have shown that this first candidate gives a feasible close-up for the TSP.

If the other candidate edge would be removed, we concluded that the subtour is disconnected from the other edges, in Figure 2(c) this is the right-most subtour. If we take a closer look at those remaining edges, we observe that there is a path which connects all the remaining edges. Hereby both nodes t^* as defined in 2(c), and t_1 have degree 1. This is the case because of the removal of x_1 at the beginning of the algorithm and the removal of x^* as before. This means we can close-up this path by adding the close-up edge connecting t_1 and t^*. This way another subtour is created, which is disconnected from the other subtour by the removal of x^*. This implies that the alternative close-up gives a feasible solution for the 2-CCP.

6 Computational Results

In this section, computational results of this study are given. In Section 6.1 the algorithm for the TSP is considered. In Section 6.2 results on the 2-CCP are given. For both algorithms, data resources are included and for the 2-CCP a description of benchmark solutions used is given. Finally, the running times are analyzed.

6.1 Traveling Salesman Problem

6.1.1 Data Resources

The Lin-Kernighan algorithm is applied to several instances of the symmetric TSP. These instances are retrieved from the so-called TSP-library[2]. The instances used are given in Table 1. Hereby the instance names are given with corresponding number of cities n.

Instance	**n**
Ulysses16	16
bayg29	29
eil51	51
eil76	76
KroA100	100
ch150	150
KroB200	200
lin318	318

Table 1: TSP Instances

The largest instance we use is the 318-city instance from Lin and Kernighan (1973). This was also the largest instance used in the original paper.

6.1.2 Computational Results

Computational results of the Lin-Kernighan algorithm applied to the symmetric TSP are given in Table 2. For all instances, 100 random starting tours are generated. The best value found among all trials is denoted in the first column, the average value over all trials is given in the second column. The solutions are compared the optimal values, denoted as **OPT**. Moreover, running times are based on a run of 100 random starting tours and are measured in seconds. Since we will elaborate on the relative running times and growth rates, we could either denote the average running times over 100 trials or the total running time.

[2]retrieved from http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

Instance	LK Solution (min)	LK solution (av)	OPT	Running Time (sec, 100 trials)
Ulysses16	74	88	74	0,3
bayg29	9150	9864	9047	1,0
eil51	428	517	426	4,4
eil76	609	898	538	16.9
KroA100	26363	41602	21285	47,8
ch150	9226	10378	6528	154,1
KroB200	42919	55790	29437	720, 7
lin318	85402	125142	42029	4038,2

Table 2: Lin-Kernighan algorithm on the TSP

An immediate observation is that the algorithm performs well on small instances, and the performance becomes worse with problem size. This is as we expected, since this also happens in the implementation in Lin and Kernighan (1973). As given in the first column, only for the smallest instance the optimum is reached. On the instances up to 51 cities, the differences with optimal values are relatively small: the minimum value obtained from the trials does only deviate by 1% from the best known solution. This increases to a deviation of about 100% for the largest instance. Another observation is that the average solutions differ from the minimum solutions by 10-20% on all instances. This difference implies that the outcomes differ over the trials. This might be because no backtracking is used in this basic implementation, which causes the solution to be affected by the choice of the starting point. In Section 7 we will further elaborate on this bias.

If we compare these results to the original heuristic solutions, a first observation is that the the results are as expected. In both implementations the heuristic performs well on smaller instances and worse on larger instances. The original heuristic performs overall slightly better: optimal solutions were found for problems with up to 50 cities, where this implementation only approaches the optimal solutions (but, as noted before, solutions are significantly close to the optimum). On the largest instance, with 318 cities, the performance of the original solution is better: it reached a solution that only deviates by 1.5% from the optimum, while we obtain a solution which deviates by a 100% from the optimal value. As mentioned before, this might be due to some sophisticated features of the original implementation, such as backtracking. This explains why the results were expected to be slightly worse than the results obtained from the original implementation.

As mentioned before, Helsgaun (2000) has also applied his improved algorithm to these instances. This implementation is shown to be more effective than the original algorithm. On all of the instances we have applied the algorithm to, Helsgaun has found the optimal solution. For most instances, the optimal solution occurred in 100/100 trials, only for the 150-city instance the success rate was 62% and for the 318-city instance this rate was 71%. These results are significantly better than this more basic implementation. An important difference in this algorithm is that instead of using nearest neighbours as measure of nearness when choosing y_i, Helsgaun's implementation uses α-nearness, which is based on sensitivity analysis on spanning 1-trees. This implementation is shown to be a much improved version, which is why a difference in performance was expected.

6.2 2-CCP

6.2.1 Data Resources

As the 2-CCP in this study is defined on a Euclidean graph as well, the modified algorithm can be tested on the same instances as the basic algorithm. However, as for the symmetric TSP the TSP-library exists, this is not the case for the 2-CCP. This is why other benchmark solutions must be found. For the instances we used with up to 150 nodes, not only optimal solutions, but also optimal tours are known. Therefore, as a benchmark solution, we split the optimal solutions in two tours. A further explanation of the splitting method is given below. Also, the Lin-Kernighan tour obtained from the basic algorithm is split similarly and used as benchmark solution.

On the Splitting of Tours

A split tour can be seen as an upper bound for the 2-CCP, since it is a feasible solution for the problem but is not known as the optimal solution. To make this upper bound as tight as possible, the tour is split at least cost. This is done by examining the exact neighbourhood. All possible pairs of edges in the tour are considered to remove. For each pair of edges there is a single way of joining endpoints together such that two separate tours are created. The pair of edges with least cost associated with the split is chosen. An illustration of this splitting method is shown in Figure 5.

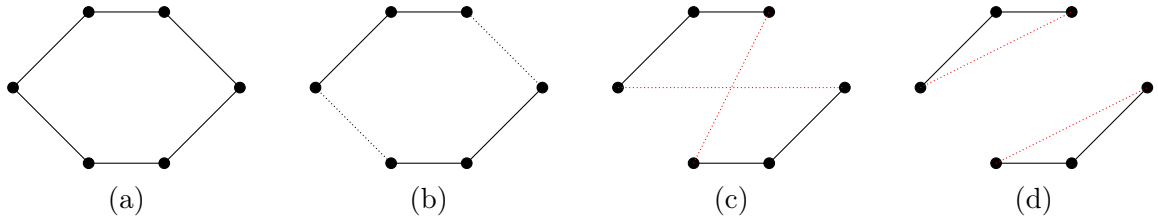


Figure 5: Splitting Tours

In Figure 5 an illustration of the splitting method is given. In (a) the TSP tour is given. In (b) one pair of edges is considered. In (c) and (d) the two possible ways of joining the endpoints together is given, and it can be observed that only one way gives two separate tours.

2-TSP as Upper Bound

As concluded earlier, the 2-CCP is closely related to the 2-TSP. This is why we also use 2-TSP solutions as benchmark solutions. It can be observed that a 2-TSP solution always gives a higher value than a 2-CCP solution and a 2-TSP solution can thus be used as an upper bound for the 2-CCP. An illustration of this fact is given in Figure 6.

(a) (b)

Figure 6: The 2-TSP and the 2-CCP

In Figure 6(a) a solution for the 2-TSP is given, which consists of two separate routes with the depot node connecting both routes. We here show that from every 2-TSP solution, a 2-CCP solution can be derived. This is shown in (b), where the 2-TSP solution is transformed in a 2-CCP solution by disconnecting one route from the depot node. Since the problems we consider are Euclidean, by the triangle-inequality we can conclude that from a 2-TSP always a lower solution for the 2-CCP can be found. Solutions for the 2-TSP can thus be easily transformed into a 2-CCP solution and can be used as an upper bound.

No optimal solutions are known for the 2-TSP on the TSP-library instances. This is why heuristic solutions, obtained from Nuriyeva and Kizilates (2017), are used as benchmark solutions. This heuristic is applied to instances from the TSP-library. Most instances are the same as previously tested on (the instances with respectively 51, 76 and 100 cities) and one instance with 70 cities, **st70** is also included. Computational results obtained by the modified Lin-Kernighan algorithm for the 2-CCP are compared to these 2-TSP heuristic solutions.

6.2.2 Computational Results

In this section the computational results of the modified algorithm are given. First the results are compared to the split tours described previously. Second, the results are compared to the 2-TSP solutions.

Split Tours

First the algorithm is applied to the TSP-library instances and is compared to the split optimal and the split Lin-Kernighan tour. The results are given in Table 3. Solutions are obtained from the algorithm first using a random starting tour, denoted as $\textbf{Solution}_{RAND}$, second using the optimal tour as starting point, $\textbf{Solution}_{OPT}$ and finally the Lin-Kernighan tour is used as starting point, $\textbf{Solution}_{LK}$. Benchmark solutions are the split optimal and the split Lin-Kernighan tour. Moreover, the results are based on 20 starting tours. Running times are averaged over the trials and over the different starting tours.

Instance	Solution_{RAND}	Solution_{OPT}	Solution_{LK}	Split_{OPT}	Split_{LK}	$\text{Running Times}_{AV}$
Ulysses16	71	66	66	68	68	0,01
bayg29	9394	8786	8789	8786	12174	0,03
eil51	513	513	509	431	584	0,06
eil76	793	545	696	643	1034	0,1
kroA100	42696	21070	38280	21175	43335	0,2
ch150	12527	6520	9038	6516	9287	0,5

Table 3: Results on the 2-CCP

Next we will discuss the results. First the results obtained with a random starting tour is discussed, second the variant which uses the optimal starting tour and finally the Lin-Kernighan starting tour is discussed.

Random Starting Tour

If a random tour is used as starting point, the algorithm is most similar to the basic Lin-Kernighan algorithm. However, it can be seen that this does not give the best results. A first observation is that the solution is always worse than the split optimal tour. For the smaller instances, the value is close to the optimal solution split. For the instances with 100 and 150 cities, the performance is poor. The solution is near 90% higher than the split optimal tour. This is as expected, since this decrease in effectiveness also occurs in the original algorithm. In general, for small instances the solutions do not deviate substantially from the optimal solution split. If the results are compared to the split Lin-Kernighan tour, only for the smallest instance, for which the Lin-Kernighan solution coincides with the optimal solution, and for the largest instance, the split tour is shorter than the heuristic solution. Since the Lin-Kernighan algorithm did not perform too well on larger instances, this is not considered a very good upper bound.

Optimal Starting Tour

If the optimal tour is chosen as the starting tour of the algorithm, we observe significantly better results. The solutions are better than the Lin-Kernighan tour split on all instances. The algorithm finds an improvement on the optimal tour split on four instances and it reaches the same value on one instance. We observe that the algorithm here mostly runs only a single LK iteration. If the merging must have positive gain, this would mean that the new starting tour is better than the previous starting tour. As the previous starting tour is the optimal tour, by definition this is not possible. For the algorithm with optimal starting tour, it is thus impossible to run more than one LK iteration if the merging must give positive gain. We use least-cost merging, which may give a slightly negative gain, but we observe that this actually does not happen. Still, one LK iteration involves multiple (k) switches of edges and can thus give an appropriate improvement on the starting tour, as we observe that the optimal tour as starting point gives the best solutions on average.

LK Starting Tour

The Lin-Kernighan starting tour gives solutions which are significantly better than the random starting tour, but mostly worse than the optimal starting tour. Only for the 51-city instance, the solution with this starting tour is best. We have observed that the best Lin-Kernighan tour for this instance is very close to the optimal solution. Apparently the Lin-Kernighan tour is easier to improve on. However, since only 20 random starting points are considered, it might also be due to coincidence. We observe that for smaller instances the solution is close to the lowest benchmark solution. For larger instances again the performance is worse, which is as expected since the Lin-Kernighan solutions are also worse for these instances, which are used as starting tour.

Comparative Analysis

The main findings of these computational results are first that the algorithm performs well compared to the benchmark instances on smaller instances and the performance becomes worse for larger instances. This is expected, since this also happens for the standard algorithm. Second, remarkable for this algorithm is that the solution seems to depend highly on the starting tour. A starting tour that is a better solution for the TSP gives in general also a better solution for the 2-TSP. This effect is caused by the fact that the algorithm mostly runs no more than a single LK edges. This means the resulting 2-CCP solution only differs from the starting solution by a total of k edges.

The algorithm can be considered an effective way of transforming TSP solutions into 2-CCP solutions since in general it is more effective than performing a least-cost split.

Comparison with the 2-TSP

In this section, the heuristic solutions are compared to 2-TSP solutions as benchmark solution. From Nuriyeva and Kizilates (2017), heuristic solutions are found on symmetric TSP-library instances. The solutions are given in Table 4.

Instance	**Solution**$_{RAND}$	**Solution**$_{OPT}$	**Solution**$_{LK}$	**Heuristic 2-TSP Solution**
eil51	513	513	509	584
st70	1322	667	1041	792
eil76	793	545	696	792
kroA100	42696	21070	38280	24911

Table 4: Results compared to 2-TSP

For these instances, as concluded before, the optimal tour as starting point gives the best solutions. If these solutions are compared to the solutions for the 2-TSP, it can be observed that the solution values are better than the 2-TSP solutions on all instances. For two instances the algorithm using Lin-Kernighan starting tour performs even better than this benchmark solution. Since the heuristic 2-TSP solutions form an upper bound for the problem, as desired the algorithm gives solutions which are lower than these upper bounds.

Running Time Analysis

In Lin and Kernighan (1973) it is stated that the running times grow with about $n^{2.2}$. In Table 2 the running times are denoted. It can be observed that for the larger instances these are relatively large. Instead of the $n^{2.2}$ in Lin-Kernighan(1973) and Helsgaun (2000), in this implementation running times grow with $n^{3.4}$ on average. The instance with 200 cities seems to be a bit of an outlier, as the running time for this instance is relatively large. The relatively large growth rate might be due to an implementation that is less efficient than the original implementation.

The running times for the extended problem are in general much smaller than for the basic algorithm. The running times grow with $n^{2.5}$ on average, which is slightly more than the original algorithm, but significantly less than the basic implementation in this study. This small growth compared to the basic algorithm might be due to the fact that the algorithm mostly only runs a single LK iteration. In other words, because of extra restriction, and the merging feature, stopping criteria occur relatively fast. This is why the running times are smaller than for the original algorithm. However, the growth rate is still bigger than in the original algorithm. Since also the basic implementation, which has been shown to be less efficient, is used for the modified algorithm, this is also as expected.

7 Conclusion and Recommendations

In this thesis the focus was first on the TSP, one of the most well studied subjects in combinatorial optimization. Some exact methods for solving the TSP exist, but can only solve problems up to a certain size. This is why many heuristic algorithms have been introduced. In this thesis, in particular the Lin-Kernighan algorithm was the subject of interest. The Lin-Kernighan algorithm dates from 1973 and is, with some improvements, considered one of the best algorithms to find (sub)optimal solutions. The algorithm starts with a feasible solution and by sequentially exchanging pairs of edges, it tries to find better solutions. The algorithm performed well in its original form, but has been improved several times. Nowadays it can solve very large problem instances in relatively small running time. The aim of this thesis was first to implement the basic Lin-Kernighan algorithm and reflect on its performance. The algorithm has been tested on instances of the TSP-library of which optimal values are known. We have observed that the algorithm performs well, especially on smaller instances. On the smallest instance, the optimal value has been reached. As in Lin and Kernighan (1973), effectiveness of the algorithm decreases with problem size. The results are slightly worse than in the original algorithm. This can be explained by the fact that the original algorithm contains some sophisticated features while this implementation only captures the basic steps of the algorithm. Also the results have been compared to the results of Helsgaun (2000) on the same instances. This implementation however is an advanced improvement on the original algorithm, and as expected it performs better than the basic implementation we used.

The second problem of interest was the 2-CCP. This is a problem that is strongly related to the TSP and its extensions. In particular the 2-CCP is closely related to the 2-TSP. The problem consists of finding two disjoint cycles of minimum length which together cover the entire graph. This problem is not as widely-studied as the TSP but some approximation algorithms

exist. The Lin-Kernighan algorithm has been successfully applied to extensions of the TSP, but it has not been adjusted for the 2-CCP yet. In this study, the aim was also to construct a heuristic algorithm for the 2-CCP based on the Lin-Kernighan heuristic and reflect on the results. We have found that an algorithm with a TSP as starting point can find 2-CCP solutions with only a few adjustments. We have concluded that the algorithm performs well, but only if the starting tour is an appropriate solution for the TSP already. First the algorithm has been tested on the same TSP-library instances as the standard algorithm. The algorithm has taken a random tour, the Lin-Kernighan tour and the optimal tour as starting point. For the last-mentioned starting tour the reuslts were best. The results are compared to the optimal tour and the Lin-Kernighan tour split in two parts at least cost. We see that the algorithm gives for most instances better solutions than these benchmark solutions. We can consider this extended algorithm as a way of transforming TSP solutions into 2-CCP solutions. If this algorithm is applied with optimal starting tour, a better solution is found than by splitting this starting tour at least-cost. The same thing holds for the Lin-Kernighan tour. The algorithm with the Lin-Kernighan starting tour gives a better solution than simply splitting this tour at least-cost.

Moreover, the algorithm has been compared to heuristic 2-TSP solutions which form an upper bound for the 2-CCP solutions. The solutions found using the optimal tour as starting point are lower than the benchmark solutions. This is as desired since the 2-TSP solutions are an upper bound on the 2-CCP solutions. Conclusively, the modified algorithm can be considered an effective algorithm for transforming TSP solutions into 2-CCP solutions.

However, the algorithm faces some limitations. In the next section these limitations are described and recommendations for further research are given.

Limitations of this Research and Recommendations for Further Research

The extended algorithm has its limitations. We discuss first the starting point bias and second the limitation regarding the the merging method.

Starting Point Bias

If we run 20 trials with the optimal starting tour, 20 different starting points are considered. It can be observed that the best solutions found do not occur in each trial. In Table 5 the number of times the best solution occurred is denoted as a fraction of all trials.

Instance	Occurrence of Best Solution
Ulysses16	3/20
bayg29	5/20
eil51	3/20
st70	1/20
eil76	1/20
kroA100	1/20
ch150	3/20

Table 5: Occurrence of Best Found Solution

This implies that the results do not only depend on the starting tour, as mentioned before, but

also on the starting point t_1. This phenomenon also occurs in the basic algorithm as implemented in this study. In Lin and Kernighan (1973) this bias is completely removed by the backtracking feature as explained previously. A simple solution could thus be to implement backtracking at least on the first level. In further research this could be done and it could be investigated to what extent this improves solution quality.

Merging

As mentioned before, there are some limitations regarding the merging method. First we desire an improvement in the next LK iteration. To guarantee this, a positive merging method must be implemented. However, we also face that if the merging is done at very low, and negative, cost, it is hard to find any improvement in the next iteration. These are two conflicting observations. In the implentation, a least-cost merging method is used. However we have observed that this often gives a new starting tour that is too tight. Further research might be conducted into exploring more sophisticated ways of merging tours. An example can be to make the merging method forward-looking, that is, check when trying a specific change of edges to merge two tours whether the next step in the following LK iteration can find an improvement already. This might increase running times, but since running times are low for this modified algorithm, such features might be considered. In general, trying different ways of merging could make the algorithm run more than a single LK iteration, which might improve solution quality and makes the solution less dependent on the starting tour.

Besides these improvements on the implementation, further research can also be done in generalizing the algorithm. The 2-CCP is a relative small class of problems. It might be useful to extend the algorithm such that it is feasible for the general class of the m-CCP.

References

Applegate, D., Bixby, D., & Cook, W. (2006). *The Traveling Salesman problem: a computational study.* Princeton, New Jersey, USA: Princeton University Press.

Bender, E. A., & Williamson, S. G. (2005). *A Short Course in Discrete Mathematics.* Dover Publications.

Biggs, N. L., Lloyd, E. K., & Wilson, R. J. (1976). *Graph Theory 1736-1936.* Oxford: Clarendon Press.

Bläser, M., Manthey, B., & Sgall, J. (2006). An improved approximation algorithm for the asymmetric TSP with strengthened triangle inequality. *Journal of Discrete Algorithms, 4(4)*, 623–632.

Croes, G. A.(1958). A Method for Solving Traveling Salesman Problems.*Operations Research*, *6(2)*, 791-812.

Dantzig, G. B., Fulkerson R., & Johnson, S. M. (1954). Solution of a large-scale traveling salesman problem.*Operations Research, 2(4)*, 393–410.

Gimadi, E. Kh., & Rykov, I. A. (2016). Asymptotically optimal approach to the approximate solution of several problems of covering a graph by nonadjacent cycles. *Proceedings of the Steklov Institute of Mathematics, 295(1)*, 57-67.

Held, M., & Karp, R. M. (1962). A Dynamic-Programming Approach to Sequencing Problems. *Journal of the Society for Industrial and Applied Mathematics, 10(1)*, 196-210.

Helsgaun, K. (2000). An Effective Implementation of the Lin-Kernighan Heuristic.*European Journal of Operations Research, 126(1)*, 106-130.

Helsgaun, K. (2017). *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems: Technical report.* Roskilde: Roskilde Universitet.

Kanellakis, P. C., & Papadimitriou, C. H. (1980). Local Search for the Asymmetric Traveling Salesman Problem. *Operations Research, 29(5)*, 1086-1099.

Karapetyan, D., & Gutin, G. (2011). Lin-Kernighan Heuristic Adaptations for the Generalized Traveling Salesman Problem. *European Journal of Operational Research, 208(3)*, 221-232.

Khachay, M., & Neznakhina, K. (2014). Approximation of euclidean k-size cycle cover problem. *Croatian Operational Research Review, 5(2)*, 177-188.

Krolak, P., Felts, W., & Marble, G. (1971). A man-machine approach toward solving the traveling salesman problem. *Communications of the ACM, 14(5)*, 327–334.

Lin, S., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research, 21(2)*, 498-516.

Manthey, B. (2008). On Approximating Restricted Cycle Covers. *Journal of the Society for Industrial and Applied Mathematics for Comuting, 38(1)*, 181-206.

Menger, K. (1932). Botenproblem (Messenger Problem). *Ergebnisse eines Matematischen Kolloquiem*, 11-12.

Nuriyeva, F., & Kizilates, G. (2017). A new Heuristic Algorithm for Multiple Traveling Salesman Problem. *TWMS Journal of Applied and Engineering Mathematics, 7(1)*, 101-109.

Rahman, M. S., & Kaykobad, M. (2005). On Hamiltonian cycles and Hamiltonian paths. *Information Processing Letters, 94(1)*, 37–41.

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M. (1977). An Analysis of Several Heuristics
for the Traveling Salesman Problem. *SIAM Journal on Computing, 6(3)*, 563–581.

Sahni, S., & Gonzalez, T. (1976). P-Complete Approximation Problems. *Journal of the ACM, 23(3)*, 555–565.

Sunil Chandran, L., & Shankar Ram, L. (2007). On the relationship between ATSP and the cycle cover problem. *Theoretical Computer Science, 370(1–3)*, 218–228.

TSPLIB. (n.d.). Retrieved July 1, 2019, from http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

Williamson, D. P., & Shmoys, D. B. (2011). *The Design of Approximation Algorithms.* Cambridge University Press.

Wilson, R. J. (1985). *Introduction to Graph Theory.* Longman Scientific & Technical.

World Traveling Salesman Problem. (n.d.). Retrieved July 1, 2019, from http://www.math.uwaterloo.ca/tsp/world

Yu, W., Liu, Z., & Bao, X. (in press). New approximation algorithms for the minimum cycle cover problem. *Theoretical Computer Science.*

A Appendix

Attached to this thesis is a zipped file which contains the code written for this thesis. In the appendix follows a list of all files included. The programs are all written in java. In the Appendix a list of all files that are included is given. A short explanation of all files is given. First all programs are included and denoted in alphabetical order. The files each contain a java class. Also a test file is included, both the test instance and a file containing the optimal tour are given.

1. **Edge.java** This class defines an Edge object, in which an edge is defined as a combination of its two endpoints. Also the comparable class is implemented.

2. **fileToInstance.java** This class converts a file from the TSP library to a TSP instance which consists of Node objects with node numbers.

3. **LinKernighanAlgorithm.java** This class creates the actual Lin-Kernighan algorithm for a TSP instance. In this class both the basic and the extended algorithm are created and tested.

4. **Node.java** This class creates a node object, in which a Node is defined as a combination of its x- and y-coordinates.

5. **readLine.java** This is an auxiliary class for the fileToInstance class. It converts a line from a file into a node number and a Node object.

6. **st70.tsp** This file contains node numbers and coordinates of the **st70** instance.

7. **st70opt.txt** This file containes a list of node numbers in order of the optimal tour.