A Heuristic for Enhancing the Accuracy of Christofides' Algorithm Based on the Choice of Minimum Spanning Tree*

Dimitrios Rontogiannis
12th Grade High School Student
Athens College, Greece
e-mail: dimronto@gmail.com

Supervised by:
Andreas Karampelas
ICT and Computer Science Teacher
Athens College, Greece
e-mail: akarampelas@haef.gr

January 4, 2019

Abstract

The *Traveling Salesman Problem (TSP)* is one of the most fundamental algorithmic problems in Computer Science. Existing exact algorithms (ie., algorithms that produce the optimal solution) have exponential worst-case time-complexity. Due to the difficulty of TSP, approximation algorithms have been proposed that are much more efficient than existing exact algorithms but whose solutions are not necessarily optimal. One of the best-known approximation algorithms for TSP is Christofides' algorithm. The purpose of our work is to propose a heuristic for enhancing the accuracy of Christofides' algorithm and to provide an experimental evaluation of its performance. More specifically, we consider a number of cities with *integer coordinates* in a restricted square of the Euclidean plane. Such a placement of cities may produce, in the general case, many different *minimum* spanning trees of the complete graph connecting all the cities. We propose a modification of Christofides' algorithm which uses many different minimum spanning trees in order to produce the approximation of the optimal solution. We provide an experimental evaluation of our heuristic. In particular, we implement in C++ three algorithms: (a) a brute-force algorithm that returns the optimal TSP solution and has exponential complexity, (b) Christofides' algorithm which has much better complexity but returns an approximate solution, and (c) our extension of Christofides' algorithm that examines many alternative minimum spanning trees. The experimental results suggest that our heuristic returns an approximate solution that is on the average significantly better than the one produced by the original Christofides' algorithm.

Keywords: Traveling Salesman Problem, Approximation Algorithms, Christofides' Algorithm.

*An earlier version of this paper written in Greek has been accepted for presentation and publication in the proceedings of the *2nd Student Conference for Research and Science*, Athens, Greece, November 2018.

1 Introduction

The *Traveling Salesman Problem (TSP)* (see [Sch][Section 7] for the history of the problem) is one of the most fundamental algorithmic problems in Computer Science. The problem can be intuitively stated as follows: "we are given a set of cities that are all linked together and the distances between them are known. Find the shortest circular route that passes once from each city". The problem has applications in circuit design, bus routing, scheduling of delivery of products, and so on. Unfortunately, the existing exact algorithms (ie., algorithms that produce the optimal solution) have exponential worst-case time-complexity. Due to the difficulty of the TSP, approximation algorithms have been proposed, which are much more efficient than exact algorithms but whose solutions are not necessarily optimal. One of the most well-known approximation algorithms for TSP is Christofides' algorithm which was originally proposed in the article [Chr76].

The purpose of our work is to propose an extension of Christofides' algorithm and to provide an experimental evaluation of its performance. More specifically, we consider a number of cities with *integer coordinates* in a restricted square of the Euclidean plane. Such a placement of cities produces in the general case many different *minimum* spanning trees of the complete graph connecting all the cities. We propose an extension of Cristofides' algorithms which uses many different minimum spanning trees in order to produce the approximation of the optimal solution. We provide an experimental evaluation of our heuristic. In particular, we implement in C++ three algorithms: (a) a brute-force algorithm that returns the optimal TSP solution and has exponential complexity, (b) Christofides' algorithm which has much better complexity but returns an approximate solution, and (c) our extension to Christofides' algorithm that examines many alternative minimum spanning trees. The experimental results suggest that our approach returns an approximate solution that is on the average significantly better than the one produced by the original Christofides' algorithm.

2 Preliminaries

In this section we give some introductory definitions and describe formally the Traveling Salesman Problem. More information can be found in books on graph theory and algorithms (such as [CLRS09] and [Sed02]).

A *graph* G consists of two sets V and E and is denoted by $G = (V, E)$. The set V is the set of *vertices* of the graph and the set E the set of *edges*. Each edge connects two vertices of G. A graph that has multiple edges between vertices is called a *multigraph* and the corresponding edges are called *parallel*. A graph with no parallel edges is said to be *simple*. A graph in which edges carry weights is called a *weighted graph*. The *degree* of a vertex is the number of vertices to which it is adjacent.

A *walk* in a graph $G = (V, E)$ is a sequence of vertices and edges of G of the form $\langle v_0, e_0, v_1, e_1, \ldots, v_{n-1}, e_{n-1}, v_n \rangle$. A *closed walk* is a walk in which the first and the last vertices coincide. A closed walk in which there are no recurring vertices apart from the initial and the final one, is called a *cycle*. A cycle that includes all the vertices of a graph is called a *Hamilton cycle*. An *Euler tour* is a closed walk in which all the vertices and all the edges of the graph appear, and in which every edge of the graph appears exactly once. A graph that does not

contain cycles and in which there exists a walk between any two edges, is called a *tree*. A *spanning tree* of a graph G is a tree that contains all the vertices of G. If G is a weighted graph, a *minimum spanning tree* of G is a spanning tree of G that has the minimum sum of weights among all the spanning trees of G. A *matching* of a graph G is a subset of its edges in which there are no edges that have common vertices. A *perfect matching* is a matching that includes all the vertices of the graph. If G is a weighted graph, a *perfect matching of minimum weight* is a perfect matching in which the sum of weights of all the edges involved is the minimum possible.

Let G be a weighted graph in which all pairs of vertices are connected by edges (such a graph is usually called a *complete graph*). The *Traveling Salesman Problem* (or *TSP* for short) is the problem of finding in G a Hamilton cycle that has the least possible sum of weights. When the vertices of the graph are points in the Euclidean plane and the weight on each edge is equal to the Euclidean distance between the two vertices of the edge, then we have the *Euclidean Traveling Salesman Problem (ETSP)*. It is known that TSP and ETSP are NP-complete problems [CLRS09], which means that the best exact algorithms currently known, need exponential time with respect to the number of vertices in order to determine the solution. In other words, even for relatively small graphs, it is practically impossible to calculate the solution in an acceptable time. Due to the difficulty of TSP and ETSP, approximation algorithms have been proposed, which are much faster than the existing exponential algorithms but whose solutions are not necessarily optimal. The most well-known approximation algorithm for the two above-mentioned problems is *Christofides' Algorithm* originally proposed in [Chr76].

3 Christofides' Algorithm

Christofides' algorithm [Chr76] takes as input a complete weighted graph $G = (V, E)$ and returns a Hamilton cycle of G. The algorithm is described below:

Christofides' Algorithm:

1. Create a minimum spanning tree T of G. Let O be the set of vertices of T that have odd degree. It can be shown that the set O contains an even number of vertices.
2. Find a minimum-weight perfect matching M in the graph consisting of the vertices of O and the edges that connect them in G.
3. Combine the edges of M and T in order to get a multigraph H in which every vertex has even degree.
4. Create an Euler tour P in H. It can be proven that H always contains such a tour.
5. Turn the Euler tour P into a Hamilton cycle by erasing recurring vertices.

Consider the complete graph $G = (V, E)$ depicted in Figure 1. In Figure 2 we give an

example of running Christofides' algorithm on G. We assume that the vertices of the graph are points in Euclidean space and the weights on the edges are their Euclidean distances. In

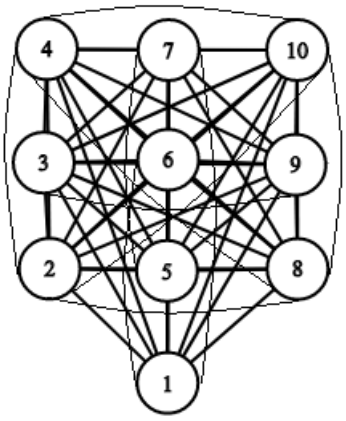

Figure 1: A complete graph on 10 vertices.

its first step, Christofides' algorithm finds a minimum spanning tree T of G and then locates the vertices with odd degrees of T (bold-faced vertices 1, 2, 6, 7, 8 and 10 in Figure 2).

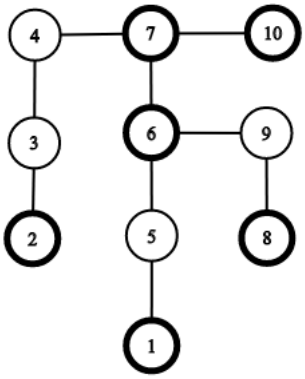

Figure 2: A minimum spanning tree T and its odd-degree vertices.

In the second step the algorithm finds a minimum-weight perfect matching M between these vertices. In our graph, such a matching consists of the edges $(1, 2)$, $(6, 8)$ and $(7, 10)$. In the third step, the algorithm combines the tree T together with the matching M and constructs a multigraph H. Figure 3 shows the combination of the tree T with the matching M (namely the multigraph H).

In its fourth step the algorithm finds an Euler tour in H. One such tour is 2-3-4-7-10-7-6-9-8-6-5-1-2. In its fifth and last step, the algorithm erases the recurring vertices from the walk (where appropriate the algorithm may add edges from the original graph). In this way we eventually get a Hamilton cycle which is shown in Figure 3.

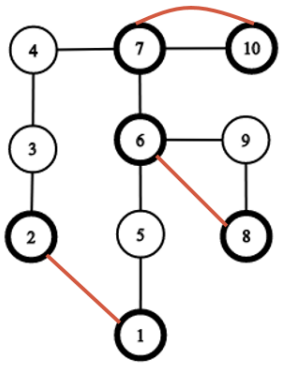

Figure 3: The multigraph H.

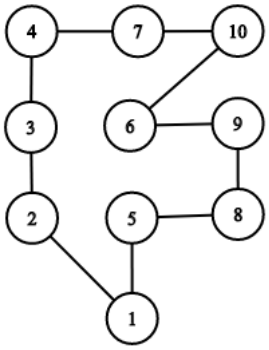

Figure 4: The final Hamilton cycle resulting from the algorithm.

It can be demonstrated that the Hamilton cycle calculated by Christofides' algorithm has a sum of weights less than or equal to the $\frac{3}{2}$ of the optimal Hamilton cycle that exists in the graph [Chr76]. For example, if the optimal Hamilton cycle in G has a cumulative weight of 1000, Christofides' algorithm is guaranteed to find a Hamilton cycle with a sum of weights less than or equal to $\frac{3}{2} \times 1000 = 1500$. In other words, Christofides' algorithm ensures finding a solution to the Traveling Salesman Problem which may not be optimal in the general case but which is close enough to the optimal solution. More importantly, Christofides' algorithm can be executed in a polynomial number of steps, much more efficiently than all existing exact algorithms that produce the optimal solution.

4 An Experimental Study of Christofides' Algorithm

For the purposes of this paper, we have implemented Christofides' algorithm for points in Euclidean space, we have studied it experimentally, and we have proposed an extension of the algorithm, which, on the basis of our experimental results, achieves in the general case a better approximation to the optimal solution. In order to be able to assess the quality of the solutions provided by Christofides' algorithm and the extension we propose, we have also implemented a brute-force algorithm that has exponential complexity but always returns the optimal solution.

Our implementations were performed in C++ and all the relevant code can be retrieved from the page: `https://github.com/rondojim/Travelling-Salesman-Problem`.

For the algorithms we implemented, we considered points in the two-dimensional Euclidean space, and more specifically in the interval $[0, 3] \times [0, 3]$, ie., within the square defined by the points of the plane with coordinates $(0, 0)$, $(0, 3)$, $(3, 3)$ and $(3, 0)$. Within this square we chose 10 points that are the vertices of our graph. For the purposes of this paper, we have limited our attention to points with coordinates that are natural numbers. Given two points, the weight of the edge joining them is the Euclidean distance of the points. We considered all possible 10-point arrangements in the square $[0, 3] \times [0, 3]$. It can be shown that there are altogether 8008 different placements of points within this square: the square has 16 points with integer coordinates, and we want to select 10 of them. The ways to do this are the combinations of 10 points out of 16, that are equal to $\frac{16!}{(10! \times 6!)} = 8008$.

For each different placement of the 10 points, we calculated the optimal Hamilton cycle using a brute-force exponential algorithm based on dynamic programming [CLRS09]. Also, for each different placement we ran Christofides' approximation algorithm. Our implementation of Christofides algorithm is not very efficient because our main goal was to experiment with improving the accuracy of the algorithm and not to create an efficient implementation. In particular, our implementation in order to compute the minimum-weight perfect matching, employs a straightforward but inefficient approach. To obtain a more efficient implementation, one could employ the well-known (but much more difficult to implement) Edmond's Blossom algorithm (see [Edm65, Kol09]).

Let L_{opt} be the optimal length of the Hamilton cycle for a given placement of points, and let L_{Chr} be the length of the Hamilton cycle calculated by Christofides' algorithm. The fraction $ratio = \frac{L_{Chr}}{L_{Opt}}$ shows how good is the approximation that Christofides' algorithm has achieved for this particular placement. This ratio is greater than or equal to 1 and less than or equal to $\frac{3}{2}$. If the ratio is close to 1, it means that Christofides' algorithm gave us a good solution, while if it is close to $\frac{3}{2}$ it means that the approximation is not very satisfactory. The diagram shown in Figure 5 gives the histogram with the distribution of the values of the variable *ratio*, for all 8008 different possible point placements we have examined. We observe

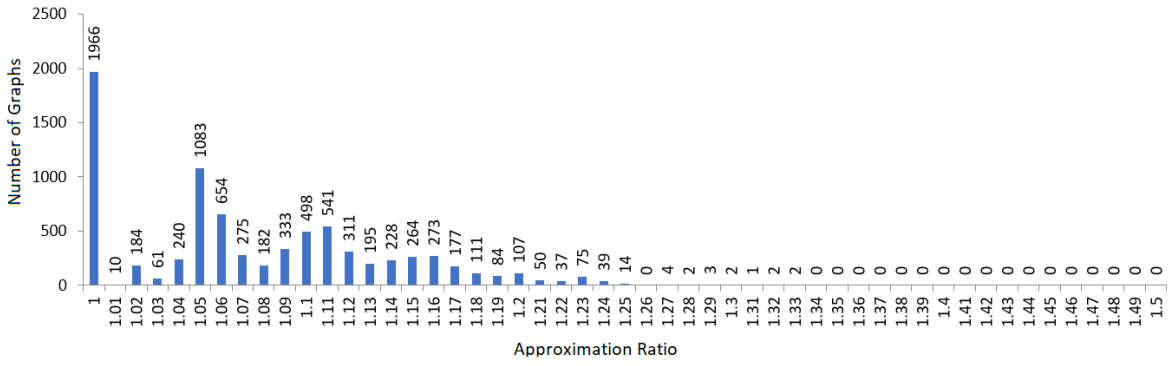

Figure 5: Distribution of approximation ratios achieved by Christofides' algorithm.

that Christofides' algorithm has a very good behavior for the data we examined. For example,

for 1966 different placements (in a total of 8008), Christofides' algorithm gives the optimal solution. Also, the worst approximation that Christofides' algorithm achieves for our data is 1.33, which is much smaller than the theoretical barrier of 1.50. The above statistical results confirm that Christofides' algorithm is indeed a good approximation algorithm in practice. The question we would like to address is whether we can further improve the behavior of the algorithm in practice. We examine this question in the next section.

5 Enhancing the Accuracy of Christofides' Algorithm

We have noticed that a basic step in Christofides' algorithm is the construction of a minimum spanning tree of our original graph G. It is possible, however, that in a graph there exist *many* minimum spanning trees. Especially in the cases we examine (namely, points with integer coordinates in a restricted square of the Euclidean space), the existence of many minimum spanning trees is the rule and not the exception. For this reason, we have modified Christofides' algorithm to examine a fixed number $k \geq 1$ of minimum spanning trees in order to produce its final result:

The modified Christofides' Algorithm:

1. Find a set of $k \geq 1$ minimum spanning trees of G.
2. For every such minimum spanning tree, run Christofides' algorithm.
3. Return the minimum Hamilton cycle resulting from all the minimum spanning trees that have been examined.

The diagram of Figure 6 shows the behavior of the modified algorithm for the same placements of the points as in the previous section of the paper. For the following results, we limited the number of minimum spanning trees we examined in each step of the algorithm to $k = 10$. As can be seen from this diagram, in more than 4700 placements (in a total of 8008), the modified algorithm gives the optimal Hamilton cycle. Moreover, the worst approximation achieved by the modified Christofides' algorithm for our data is 1.23, which is much smaller than the theoretical barrier of 1.50 and also quite smaller than the experimental result of 1.33 mentioned in the previous section of the article. A more detailed experimental comparison, for various values of k, is shown in the following table.

The entries of the table correspond to the following:

- *Average approximation ratio:* It is the average approximation ratio that our heuristic achieves over all the 8008 different 10-point arrangements in the square $[0, 3] \times [0, 3]$.
- *Optimal solution (# of graphs):* It is the number of different 10-point arrangements (out of 8008) for which our heuristic calculates the optimal Hamilton cycle.

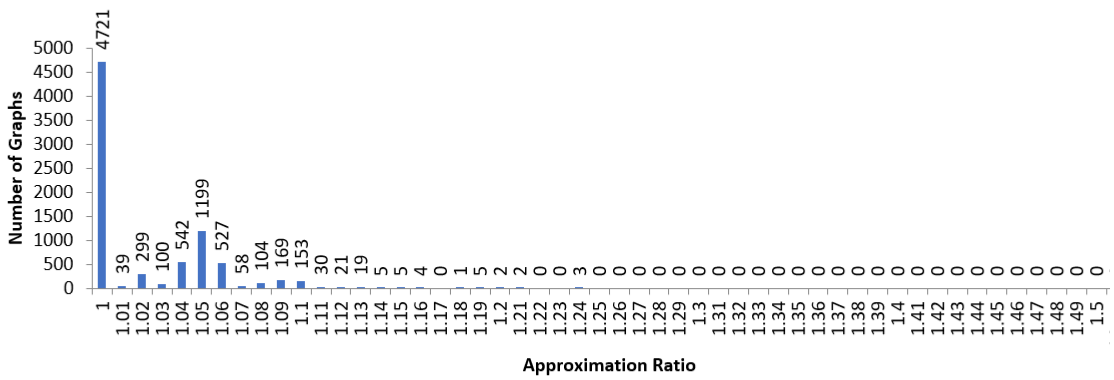

Figure 6: Distribution of approximation ratios achieved by our extension to Christofides' algorithm.

k	Average approximation ratio	Optimal solution (# of graphs)	Execution Time (in sec)
1	1.0710	1966	3285
2	1.0476	2880	4173
3	1.0369	3364	5163
4	1.0318	3735	6253
5	1.0278	4009	7272
6	1.0248	4268	8220
7	1.0236	4378	9273
8	1.0220	4511	10288
9	1.0209	4653	11430
10	1.0201	4721	13703

Table 1: Experimental comparison based on the number k of spanning trees used.

- *Execution time:* It is the total time that our implementation requires in order to calculate the optimal solutions for all 8008 different 10-point arrangements and also to run our heuristic for all these arrangements.

The average approximation ratio as a function of the number k of minimum spanning trees that we use, is depicted in Figure 7. Obviously, the average approximation ratio decreases as k increases. However, this decrease is much more significant in the beginning (for example, when we move from $k = 1$ to $k = 2$); for later values of k the decrease of the average approximation ratio is less impressive.

In Figure 8 we depict the number of graphs for which the proposed heuristic finds the optimal solution. We observe that as k increases, the number of graphs for which the heuristic finds the optimal solution, also increases.

Figure 9 depicts the total execution time of our implementation. Notice that the execution time increases linearly with respect to k, something expected because our heuristic simply

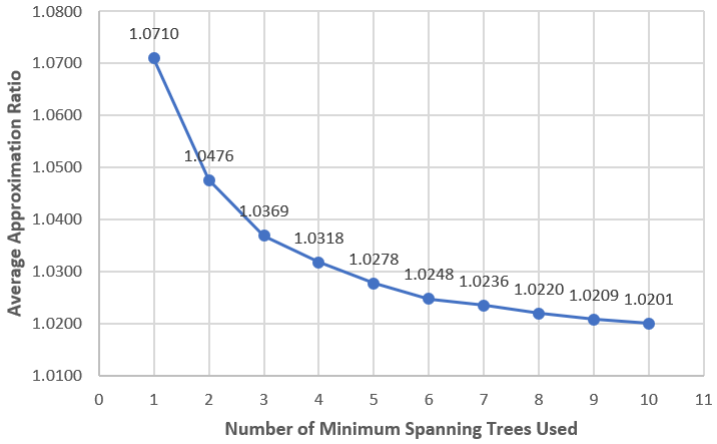

Figure 7: Average approximation ratio as a function of k.

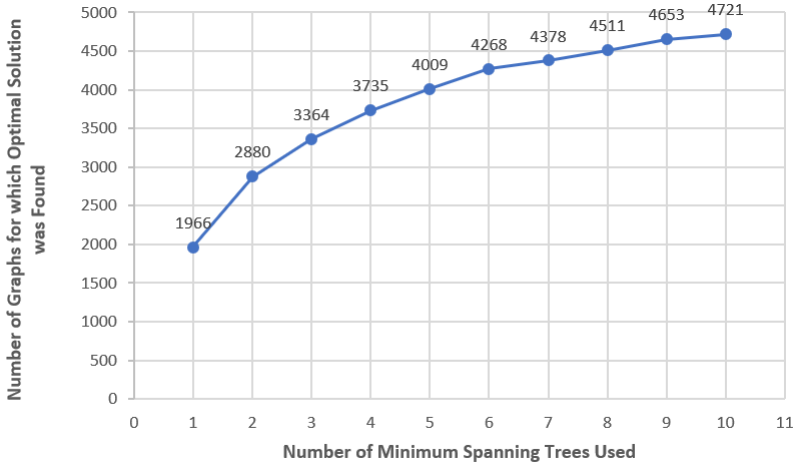

Figure 8: Number of optimal solutions as a function of k.

repeats Christofides' algorithm k times. Notice also that the execution times are quite significant. This is due to three main reasons: (1) For every different k, the time reported is for examining all 8008 different placements of points in the square $[0,3] \times [0,3]$; (2) For every such different placement we first run a brute-force exponential algorithm in order to find an optimal Hamilton cycle and then run Christofides' algorithm k times and select the best solution among the k ones; (3) Our implementation is far from optimal. In particular, as we have already mentioned, in order to find a minimum-weight perfect matching, our implementation uses an inefficient (but easy to implement) approach.

In conclusion, the heuristic proposed in this section has the advantage of giving, in the general case, a better accuracy in practice compared to the original algorithm of Christofides. In order to achieve this, it runs the initial algorithm for a number k of minimum spanning trees of a graph, and at the end selects the best result. We believe that the proposed heuristic can be used in practice in order to improve the accuracy of the solutions obtained by Christofides' algorithm.

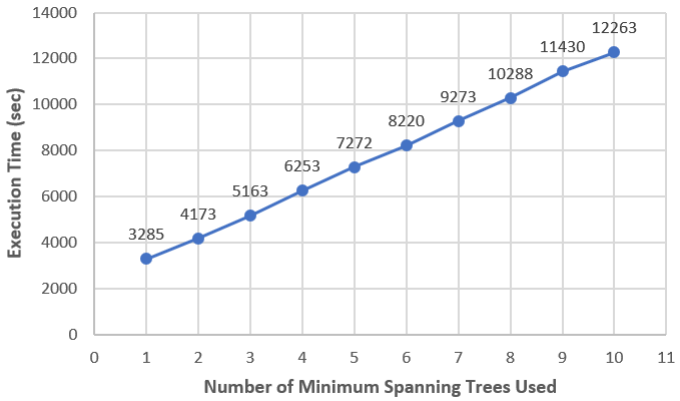

Figure 9: Total execution time as a function of k.

6 Future Work

In the experiments we presented in the previous sections, we examined graphs having 10 vertices. The reason we chose a relatively small number of vertices was because we wanted to run our algorithms on all possible 10-point graphs in the $[0, 3] \times [0, 3]$ region of the space. If we had chosen a larger number of vertices and a more extended segment of the plane, then the total time we would need to investigate all possible graphs would be considerably larger. Also, the runtime of the "exhaustive" (brute-force) exponential algorithm for larger graphs increases very quickly. For these reasons, we did not study graphs with more vertices in this paper. In the future, however, we intend to apply the proposed technique to larger graphs, and examine the improvements it offers.

It is also interesting to extend the technique we proposed to randomly selected points in the Euclidean space with non-integer coordinates. In this case, the probability of finding more than one minimum spanning trees seems to be less than in the case we examined in this paper (the integer coordinates in a restricted square of the plane, increase the likelihood of obtaining many minimum spanning trees). In the case of random coordinates we could look at the k *shortest spanning trees* in the graph. In the literature there have been proposed algorithms for finding the k shortest spanning trees of a graph (see for example [Epp92]). It would be interesting to experimentally examine, in this more general case, how quickly the approximation ratio decreases when k increases. It would also be interesting to study the consequences of choosing different minimum-weight perfect matchings in the construction of the Hamilton cycle.

References

[Chr76] N. Christofides. *Worst-case analysis of a new heuristic for the travelling salesman problem.* Report 388, Graduate School of Industrial Administration, Carnegie Mellon, 1976.

[CLRS09] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.

[Edm65] J. Edmonds. *Paths, trees, and flowers*. Canadian Journal of mathematics 17.3: pages 449-467, 1965.

[Epp92] D. Eppstein. *Finding the k Smallest Spanning Trees.* BIT 32(2): pages 237-248, 1992.

[Kol09] Kolmogorov. V. *Blossom V: a new implementation of a minimum cost perfect matching algorithm*. Mathematical Programming Computation. 1(1): pages 43-67, 2009.

[Sed02] R. Sedgewick. *Algorithms in C++ (Part 5: Graph Algorithms)*. Princeton University, 2002.

[Sch] A. Schrijver. *On the history of combinatorial optimization (till 1960)*. Available from: `https://homepages.cwi.nl/~lex/files/histco.pdf`.