



ELSEVIER

Contents lists available at ScienceDirect

## Journal of Discrete Algorithms

www.elsevier.com/locate/jda



# A new upper bound for the traveling salesman problem in cubic graphs

Maciej Liškiewicz\*, Martin R. Schuster

Institut für Theoretische Informatik, Universität zu Lübeck, Ratzeburger Allee 160, 23538 Lübeck, Germany

## ARTICLE INFO

### Article history:

Received 8 January 2013

Received in revised form 17 October 2013

Accepted 5 February 2014

Available online xxxx

### Keywords:

Exact algorithms

Traveling salesman problem

Cubic graphs

Hamiltonian cycle

## ABSTRACT

We provide a new upper bound for traveling salesman problem (TSP) in cubic graphs, i.e. graphs with maximum vertex degree three, and prove that the problem for an  $n$ -vertex graph can be solved in  $\mathcal{O}(1.2553^n)$  time and in linear space. We show that the exact TSP algorithm of Eppstein, with some minor modifications, yields the stated result. The previous best known upper bound  $\mathcal{O}(1.251^n)$  was claimed by Iwama and Nakashima [Proc. COCOON 2007]. Unfortunately, their analysis contains several mistakes that render the proof for the upper bound invalid.

© 2014 Elsevier B.V. All rights reserved.

## 1. Introduction

It is an outstanding open problem whether the traveling salesman problem (TSP) and the closely related Hamiltonian cycle problem can be solved in  $\mathcal{O}(c^n)$  time for graphs on  $n$  vertices, for some constant  $c < 2$ . Recently Björklund et al. [3] have shown that the classical Bellman–Held–Karp exact algorithms [2,9] for solving TSP can be modified to run in time  $\mathcal{O}((2 - \varepsilon)^n)$ , where  $\varepsilon > 0$  depends only on the maximum vertex degree. This provides the first upper bound on the time complexity of TSP that lies below  $2^n$  for a broad class of graphs such as bounded degree graphs. Particularly, applying the result of [3] for graphs with maximum vertex degree three, also called cubic graphs, one gets that TSP can be solved in time  $2^{3n/4} n^{\mathcal{O}(1)} = \mathcal{O}(1.682^n)$ . On the other hand, the problem of testing whether a cubic graph has a Hamiltonian cycle and consequently the decision version of TSP remain NP-complete even if the graphs are restricted to be planar [7].

Exact algorithms for TSP for special classes of bounded degree graphs, in particular for cubic graphs, have been the subject of separate studies. The motivation for the study comes both from theoretical concerns and from practical applications, e.g. in computer graphics [1,5]. The first exact algorithm for TSP in cubic graphs running faster than in time  $2^n$  was proposed by Eppstein [4]. His algorithm solves the problem in  $2^{n/3} n^{\mathcal{O}(1)} = \mathcal{O}(1.260^n)$  time and linear space and additionally it is easy to implement. Thus, although the technique by Björklund et al. improves the upper bound  $2^n$  for any degree bounded  $\geq 3$ , for specific bounds, like e.g. 3, better methods exist. Eppstein's algorithm is a sophisticated recursive branch-and-bound search, which takes advantage of small vertex degrees in a graph. To speed-up searching, the algorithm uses the fact that in cubic graphs a selection of an edge to a recursively constructed Hamiltonian cycle forces several further edges to be in the cycle or not. In [10] Iwama and Nakashima slightly modify Eppstein's algorithm and provide a new interesting method to bound the number of worst-case branches in any path of the branching tree corresponding to recursive subdivisions of the problem. As a consequence, Iwama and Nakashima claim  $\mathcal{O}(1.251^n)$  to be an upper bound for the run-time of the algorithm. Unfortunately, their paper contains several serious mistakes that render the proof for the upper bound invalid

\* Corresponding author.

E-mail addresses: [liškiewi@tcs.uni-luebeck.de](mailto:liškiewi@tcs.uni-luebeck.de) (M. Liškiewicz), [schuster@tcs.uni-luebeck.de](mailto:schuster@tcs.uni-luebeck.de) (M.R. Schuster).

(for details, see Section 6). After reformulating the key lemma of [10] to be correct and then using the lemma to solve the recurrences derived in [10] in a proper way one could prove the upper bound  $\mathcal{O}(1.257^n)$  that still beats the bound  $\mathcal{O}(1.260^n)$  by Eppstein.

#### Our result

In this article we provide a new upper bound for TSP in cubic graphs. We show that Eppstein's algorithm with some minor modifications, similar to those used in [10], yields the stated result:

**Theorem 1.** *The traveling salesman problem for an  $n$ -vertex cubic graph can be solved in  $\mathcal{O}(1.2553^n)$  time and in linear space.*

Our proof techniques are based on ideas used by Eppstein [4] and Iwama and Nakashima [10] and a new, more careful study of worst-case branches in the tree of recursive subdivisions of the problem performed by the algorithm. Thus, our main contribution is more analytical than algorithmic. Nevertheless, we have implemented our algorithm and verified its easy implementability and good performance for graphs up to 114 vertices (for the experimental results see [12]).

#### Related work

Applying the result by Björklund et al. for an  $n$ -vertex graph with maximum degree four one gets that TSP can be solved in  $\mathcal{O}(1.856^n)$  time and exponential space. Eppstein [4] showed that the problem can be solved in  $\mathcal{O}(1.890^n)$  time but using only polynomial space. Next, Gebauer [8] proposed an algorithm that runs in time  $\mathcal{O}(1.733^n)$ . This algorithm can also list the Hamiltonian cycles but it uses exponential space. Very recently, Cygan et al. [11] have shown a Monte Carlo algorithm with constant one-sided error probability that solves the Hamiltonian cycle problem in  $\mathcal{O}(1.201^n)$  time for cubic graphs and in  $\mathcal{O}(1.588^n)$  time for graphs of maximum degree four. Though the technique presented in [11] works well for the Hamiltonian cycle problem, it is not usable for TSP.


#### Paper organization

In Section 2 we recall Eppstein's algorithm and in Section 3 main ideas of our proof techniques are sketched. In Section 4 we specify our modifications of Eppstein's algorithm and provide its analysis. Section 5 presents the proof of our main technical result on the number of worst-case branches of the branching tree. Finally, Section 6 discusses some issues concerning the paper by Iwama and Nakashima and the last section presents our conclusions.

## 2. Outline of Eppstein's algorithm

Eppstein's TSP algorithm [4] (see also Appendix A) searches *recursively* for a minimum weight Hamiltonian cycle  $H_{\min}$  in a given cubic graph  $G$ . The algorithm constructs successively Hamiltonian cycles which are determined by a set of *forced edges*  $F$ . The goal is to find the set  $F$  which coincides with  $H_{\min}$ . In each recursion step, an edge  $e \in G \setminus F$  is chosen (Step 3). Obviously,  $e$  either belongs to  $H_{\min}$  or not. Thus, the algorithm makes two recursive calls: once  $e$  is added to  $F$  (Step 4), assuming  $e \in H_{\min}$ , and once  $e$  is removed from  $G$  (Step 5), assuming  $e \notin H_{\min}$ . The better solution to these two subproblems will then be returned (Step 6).

At the beginning of each recursive call,  $G$  and  $F$  are simplified iteratively (Step 1). If  $F$  contains two edges meeting at a single vertex  $v$ , the third edge incident to  $v$  cannot be part of  $H_{\min}$ , and therefore it is removed from  $G$ . If  $G$  contains a vertex with degree two, both incident edges have to be in  $H_{\min}$ , thus, they are added to  $F$ . If  $G$  contains parallel edges, one of the edges is removed from  $G$  depending on the weight and on  $F$ . Next, if  $G$  contains a cycle of four unforced edges, two

opposite vertices of which are each incident to a forced edge outside the cycle, like here , then all non-cycle edges that are incident to a vertex of the cycle are added to  $F$ . Furthermore, triangles are contracted to a single vertex by adjusting the weights of attached edges. These techniques reduce the size of the recurrence tree of the algorithm and enforce  $G$  to be simple, cubic and triangle-free. Simultaneously, it is verified whether the current set of forced edges  $F$  leads to a contradiction: e.g.  $F$  might contain three edges meeting at a single vertex or  $F$  contains a non-Hamiltonian cycle. Then the algorithm returns a *None-value*.

Another idea to reduce the size of the recurrence tree is to stop the recursion when  $G \setminus F$  consists of a set of non-connected 4-cycles. Then a solution is found by constructing a minimum spanning tree on some helper graph  $G'$  (Step 2).

The choice of an edge  $e$  (Step 3) which specifies two subproblems for the recursive calls, plays a crucial role in our analysis. Fig. 1 illustrates an example splitting of the problem into two subproblems determined by  $e$ . A newly formed subproblem may induce several further changes to  $G$  and  $F$  by Step 1, as shown in Fig. 1, too. Here, and in the rest of this paper, the bold black line indicates an edge in  $F$ , the bold gray line an edge selected to be added to  $F$  by the current splitting or forced to be added to  $F$  in Step 1 directly after the splitting, and the dotted line indicates an edge removed by the splitting or forced not to appear in a cycle.

Download English Version:

<https://daneshyari.com/en/article/6874793>

Download Persian Version:

<https://daneshyari.com/article/6874793>

[Daneshyari.com](https://daneshyari.com)