THEORY OF COMPUTING, Volume 10 (9), 2014, pp. 217–236
www.theoryofcomputing.org

Improved Inapproximability for TSP

Michael Lampis*

Received November 28, 2012; Revised November 27, 2013; Published September 27, 2014

Abstract: The Traveling Salesman Problem is one of the most studied problems in the theory of algorithms and its approximability is a long-standing open question. Prior to the present work, the best known inapproximability threshold was 220/219, due to Papadimitriou and Vempala. Here, using an essentially different construction and also relying on the work of Berman and Karpinski on bounded-occurrence CSPs, we give an alternative and simpler inapproximability proof which improves the bound to 185/184.

ACM Classification: G.1.6

AMS Classification: 68Q17

Key words and phrases: inapproximability, traveling salesman

1 Introduction

The Traveling Salesman Problem (TSP) is one of the most widely studied algorithmic problems and deriving optimal inapproximability results for it is a long-standing question. On the algorithmic front, there has been much progress recently, at least in the important special case where the metric is derived from an unweighted graph. This case is often referred to as the Graphic TSP. For over 30 years, the $3/2$-approximation algorithm by Christofides [5] (1976) was the best result known. This changed in 2011 when Gharan, Saberi, and Singh gave a slight improvement [8] for Graphic TSP. Subsequently an algorithm with approximation ratio 1.461 was given by Mömke and Svensson [13]. With improved analysis of their algorithm, Mucha obtained a ratio of $13/9$ [14]. The best currently known algorithm has ratio 1.4 and is due to Sebő and Vygen [18].

A conference version of this paper appeared in the Proceedings of the 15th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'2012) [12].

*Supported by ERC Grant 226203.

DOI: 10.4086/toc.2014.v010a009

Nevertheless, there is still a significant gap between the guarantee of the best approximation algorithms we know and the best inapproximability results. The TSP was first shown to be NP-hard to approximate to within $1+\varepsilon$, for some $\varepsilon > 0$ by Papadimitrious and Yannakakis [17] but no explicit inapproximability constant was derived. The work of Engebretsen [6] and Böckenhauer et al. [4] gave inapproximability thresholds of $5381/5380$ and $3813/3812$, respectively. Later this was improved to $220/219$ in [16] by Papadimitriou and Vempala.[1] No further progress has been made on the inapproximability threshold of this problem in the more than ten years since the conference version of the Papadimitriou and Vempala paper appeared ([15]). For the special case of the problem with bounded metrics the works of Engebretsen and Karpinski [7] and more recently Karpinski and Schmied [11] give inapproximability constants only slightly weaker than those known for the general version.

Overview: Our main objective in this paper is to give a different, less complicated inapproximability proof for TSP than the one given in [15, 16]. The proof of [16] is very much optimized to achieve a good constant: the authors reduce directly from MAX-E3-LIN2, a constraint satisfaction problem (CSP) where each constraint is a mod 2 linear equation in exactly 3 variables. For MAX-E3-LIN2, optimal inapproximability results are known, due to Håstad [9]. They take care to avoid introducing extra gadgets for the variables, using only gadgets that encode the equations. Finally they define their own custom expander-like notion on graphs to ensure consistency between tours and assignments. Then the reduction is performed in essentially one step.

Here on the other hand we take the opposite approach, choosing simplicity over optimization. We also start from MAX-E3-LIN2 but go through two intermediate CSPs. The first step in our reduction gives a set of equations where each variable appears at most five times (this property will come in handy in the end when proving consistency between tours and assignments). In this step, rather than introducing something new we rely heavily on machinery developed by Berman and Karpinski to prove inapproximability for bounded-occurrence CSPs [1, 2, 3]. As a second step we reduce to MAX-1-IN-3-SAT, a CSP where each constraint requires exactly one out of three literals to be true. The motivation is that the 1-IN-3 predicate nicely corresponds to the objectives of TSP, since we represent clauses by gadgets and the most economical solution will visit all gadgets once but not more than once. Another way to view this step is that we use MAX-1-IN-3-SAT as an aid to design a TSP gadget for parity. Finally, we give a reduction from MAX-1-IN-3-SAT to TSP.

This approach is (arguably) simpler than the approach of [16], since some of our arguments can be broken down into independent pieces, arguing about the inapproximability of intermediate, specially constructed CSPs. We also benefit from using the amplifier constructed in [3]. Interestingly, putting everything together we end up obtaining a slightly better constant than the one known prior to this work. Though we are still a long way from an optimal inapproximability result, our results indicate that there may still be hope for better bounds with existing tools. Exploring how far these techniques can take us with respect to TSP (and also its variants, see for example [11]) may thus be an interesting question.

The main result of this paper is given below. The result follows directly from the construction in Section 4.1 and Lemmata 4.1 and 4.2.

Theorem 1.1. *There is no polynomial-time* $(92.3/91.8-\varepsilon)$*-approximation algorithm for* TSP *for any* $\varepsilon > 0$ *unless* $\mathsf{P} = \mathsf{NP}$.

[1]The reduction of [16] was first presented in [15], which (erroneously) claimed a better bound.

Let us also note that, since the conference version of this paper appeared, recent work ([10]) has pushed the idea of building a modular reduction further. In particular, this has resulted not only in a further improvement of the inapproximability constant for TSP (to $123/122$), but also in an improvement of the constant for the asymmetric case of the problem (from the $117/116$ bound given in [16] to $75/74$).

2 Preliminaries

We will denote graphs by $G(V,E)$. All graphs are assumed to be undirected, loop-less and edge-weighted, meaning that there is also a function $w : E \to \mathbb{R}^+$. In some cases we will allow E to be a multi-set, that is, we may allow parallel edges. In the case of a multi-set E that contains several copies of some elements, when we write $\sum_{e \in E} w(e)$ we mean the sum that has one term for each copy. A (multi-)graph is Eulerian if there exists a closed walk that visits all its vertices and uses each edge once. It is well known that a (multi-)graph is Eulerian iff it is connected and all its vertices have even degree. We will use $[n]$ to denote the set $\{1,2,\dots,n\}$. We will use $\mathbf{E}[X]$ to denote the expectation of a random variable X.

In the metric Traveling Salesman Problem (TSP) we are given as input an edge-weighted undirected graph $G(V,E)$. Let $d(u,v)$, for $u,v \in V$ denote the shortest-path distance from u to v. The objective is to find an ordering $v_1, v_2, \dots, v_n$ of the vertices such that $\sum_{i=1}^{n-1} d(v_i, v_{i+1}) + d(v_n, v_1)$ is minimized. Note that this is equivalent to the alternative formulation of the problem where we are given all pairwise distances between points and distances satisfy the triangle inequality: shortest path distances on a graph always satisfy the triangle inequality, and the input graph is allowed to be an edge-weighted clique.

Another, equivalent view of the TSP is the following: given an edge-weighted graph $G(V,E)$ we seek to find a multi-set E_T consisting of edges from E such that E_T spans V, the graph $G_T(V,E_T)$ is Eulerian and the sum of the weights of all edges in E_T is minimized. It is not hard to see that the two formulations are equivalent. (The subscript T refers to the Eulerian "tour.") We will make use of this second formulation because it makes some arguments on our construction easier.

We generalize the Eulerian multi-graph formulation as follows. A multi-set E_T of edges from E is a *quasi-tour* if the degrees of all vertices in the multi-graph $G_T(V,E_T)$ are even. In other words, a quasi-tour is a tour that is not required to be connected. The cost of a quasi-tour is defined as $\sum_{e \in E_T} w(e) + 2(c(G_T) - 1)$, where $c(G_T)$ denotes the number of connected components of the multi-graph. The intuition is that we want to allow quasi-tours (because this will simplify some proofs) but penalize them for leaving some components disconnected. It is not hard to see that a TSP tour can also be considered a quasi-tour with the same cost (since for a normal tour $c(G_T) = 1$), but in a weighted graph there could potentially be a quasi-tour that is cheaper than the optimal tour.

One useful property of quasi-tours is the following:

Lemma 2.1. *There exists a quasi-tour of minimum cost such that all edges with weight one are used at most once.*

Proof. If a quasi-tour uses a unit-weight edge twice then we can remove both of these appearances of the edge from the solution without increasing the cost, since the number of components can only increase by one. □

2.1 Forced edges

As mentioned above, we will view TSP as the problem of selecting edges from E to form a minimum-weight multi-set E_T that makes the graph Eulerian. It is easy to see that no edge will be selected more than twice, since if an edge is selected three times we can remove two copies of it from E_T and the graph will still be Eulerian while we have improved the cost.

In our construction we would like to be able to stipulate that some edges are to be used at least once in any valid tour. We can achieve this with the following trick. Suppose that there is an edge (u,v) with weight w that we want to force into every tour. We sub-divide this edge a large number of times, say $p-1$ times, that is, we remove the edge and replace it with a path of p edges going through new vertices of degree two. We then distribute the original edge's weight to the p newly formed edges, so that each has weight w/p. Now, any tour that omits two or more of the newly formed edges must be disconnected. Any tour that omits exactly one of them can be augmented by adding two copies of the unused edge. This only increases the cost by $2w/p$, which can be made arbitrarily small by giving p an appropriately large value. Therefore, we may assume without loss of generality that in our construction we can force some edges to be used at least once. Note that these arguments apply also to quasi-tours.

Another way to summarize the above is the following. Let F-TSP be the version of the problem where the input also contains forced edges. If F-TSP is NP-hard to approximate within some constant c then for all $\varepsilon > 0$, TSP is NP-hard to approximate within $c-\varepsilon$. In other words, the problems are essentially equivalent, and because of this we will focus on the version of the problem where edges can be forced, and simply refer to it as TSP from now on.

3 Intermediate CSPs

In this section we will design a family of instances of MAX-1-IN-3-SAT with some special structure and prove their inapproximability. We will use these instances (and their structure) in the next section where we reduce from MAX-1-IN-3-SAT to TSP.

Let I_1 be a system of m linear equations mod 2, each consisting of exactly three variables. Let n be the total number of variables appearing in I_1 and let the variables be denoted as x_i, $i \in [n]$. Let B be the maximum number of times any variable appears. We will make use of the following seminal result due to Håstad [9]:

Theorem 3.1 (Håstad)**.** *For all $\varepsilon > 0$ there exists an integer B such that given an instance I_1 as above it is NP-hard to distinguish the case when there is an assignment that satisfies at least $(1-\varepsilon)m$ equations from the case when all assignments satisfy at most $(1/2+\varepsilon)m$ equations.*

3.1 Bounded occurences

In I_1 each variable appears at most B times, where the constant B depends on ε. We would like to reduce the maximum number of occurrences of each variable to a small absolute constant. For this, one typically uses some kind of expander or amplifier. Here we will rely on a class of amplifier graphs due to Berman and Karpinski that reduces the number of occurrences to 5. These and other classes of amplifier graphs have been used to give hardness of approximation results for several problems (see [3, 2, 1]).

Theorem 3.2 (Theorem 3 in [3])**.** *Let B be a multiple of 5. Then for all sufficiently large B there exists a bipartite graph $G(L,R,E)$, with $|L| = B$, $|R| = 0.8B$, all vertices of L having degree 4 and all vertices of R having degree 5 that additionally has the following connectivity property: for any $S \subseteq L \cup R$ such that $|S \cap L| \leq |L|/2$ the number of edges in E with exactly one endpoint in S is at least $|S \cap L|$.*

Unfortunately, a full version of [3] has not appeared so far. We therefore provide a (slightly simplified) proof of Theorem 3.2 below just for the sake of completeness. The reader may feel free to skip this proof as only the statement of Theorem 3.2 is needed for the rest of the paper.

Proof. The proof proceeds via the probabilistic method. Let L, R be two sets of vertices such that $|L| = |R| = 4n$, where n is a multiple of 5. First, partition L into n sets of size 4 and R into $0.8n$ sets of size 5 each. Then, select a perfect matching from L to R uniformly at random. Collapse each set from the partitions of L, R into a single vertex. It is easy to see that we have constructed a random bipartite graph that is 4-regular on one side and 5-regular on the other. We will show that with high probability the graph also satisfies the connectivity property. In particular, we will show that for some $\varepsilon > 0$ the probability (over the random matchings) that there exists a "bad" set S violating the property is at most $2^{-\varepsilon n}$, which tends to 0 for large n. More precisely, we will examine various "configurations" that S can have, and find the configuration which has the maximum probability of being bad. We will then show that this is so small that by taking a union bound we can guarantee that with high probability no bad set exists.

If there exists such a bad set S, we can assume without loss of generality that vertices of R belong to S if and only if the majority of their neighbors are in S. In other words, all of the set S can be determined by $S \cap L$, while the placement of vertices of R into S and $V \setminus S$ can be done greedily by counting the number of neighbors each vertex has in S, because doing so can only decrease the size of the produced cut.

Suppose we have a set S such that $|S \cap L| = s$. Let A_i, $i \in \{0, 1, \ldots, 5\}$ be the set of vertices of R that have exactly i neighbors in S. Thus, $S = (S \cap L) \cup A_5 \cup A_4 \cup A_3$. We denote $|A_i| = a_i$ and let the number of edges with exactly one endpoint in S be c. The probability that a specific subset of L of size s will attain the configuration described (that is, a_i vertices of R have i neighbors in S and c edges are cut in total) can be calculated exactly as:

$$P(n,s,c,a_i) = \frac{(0.8n)!}{\prod_{i=0}^{5} a_i!} 5^{a_1+a_4} 10^{a_2+a_3} \frac{(4s)!(4n-4s)!}{(4n)!} . \tag{3.1}$$

Let us explain this. We count the number of matchings that lead to such a configuration. First, we can select a partition of the $0.8n$ vertices of R into the sets A_i, so the first factor is an expansion of

$$\binom{0.8n}{a_0, a_1, a_2, a_3, a_4} .$$

Then, since each vertex of R is actually the result of collapsing 5 vertices, we have to decide which of these are matched to endpoints in S. Thus we have 5 choices for vertices in A_1 and A_4 and $\binom{5}{2} = 10$ choices for vertices in A_2 and A_3. Finally, after having decided which of the endpoints of R will be matched to the $4s$ endpoints in S we select one of the $(4s)!$ possible matchings and do similarly for $V \setminus S$. To obtain a probability we divide by $(4n)!$, the total size of the probability space.

Of course, for the above calculation to make sense the numbers n, s, c, a_i must obey some further constraints. In particular:

$$a_0 + a_1 + a_2 + a_3 + a_4 + a_5 = 0.8n\,, \tag{3.2}$$

$$a_1 + 2a_2 + 2a_3 + a_4 = c\,, \tag{3.3}$$

$$a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 = 4s\,, \tag{3.4}$$

$$c < s \leq 0.5n\,. \tag{3.5}$$

Constraint (3.2) simply states that the A_i partition R. Constraint (3.3) measures the number of cut edges. Constraint (3.4) makes sure that a perfect matching is possible using the selected number of endpoints. Finally, constraint (3.5) states that we are interested in a bad set.

Since all subsets of L of size s are equivalent, by union bound the probability that there exists a set of size s achieving a certain configuration is at most $U(n,s,c,a_i) = P(n,s,c,a_i)\binom{n}{s}$. Expanding equation (3.1) we get

$$U(n,s,c,a_i) = \frac{(0.8n)!}{\prod_{i=0}^{5} a_i!} 5^{a_1+a_4} 10^{a_2+a_3} \frac{(4s)!(4n-4s)!}{(4n)!} \binom{n}{s}\,. \tag{3.6}$$

Our proof strategy consists of showing that the maximum value of $U(n,s,c,a_i)$ subject to constraints (3.2), (3.3), (3.4), (3.5) is at most $2^{-\varepsilon n}$. If we show this, because there are only polynomially (in n) many choices for the parameters s, c, a_i, again by union bound we will have shown that the probability that a bad set exists tends to 0. Thus, the rest of the proof consists of finding the values that maximize U as a function of n and showing an upper bound on this maximum. We do this by proving a series of intuitive (though sometimes tedious) claims.

Before we go on, let us simplify the constraints slightly. From constraints (3.3) and (3.5) we have that $a_1 + a_2 + a_3 + a_4 \leq s$, while from constraint (3.4) we have that $a_1 + a_2 + a_3 + a_4 + 5a_5 \leq 4s$. Multiplying the first inequality with 4 and summing we get (after dividing by 5) $a_1 + a_2 + a_3 + a_4 + a_5 \leq 1.6s$. Using constraint (3.2) this gives

$$a_0 \geq 0.8n - 1.6s\,. \tag{3.7}$$

We will now find an upper bound on the maximum value attainable by U subject to constraints (3.2), (3.3), (3.5), (3.7). Notice that we have added constraint (3.7), which was already implied by the other constraints, so this cannot affect the maximum, while we dropped constraint (3.4), which can only make the maximum larger.

Claim 3.3. *The function $U(n,s,c,a_i)$ when subject to constraints* (3.2), (3.3), (3.5) *and* (3.7) *is maximized when $|n/2 - s| = O(1)$.*

Proof. Intuitively, this claim makes sense, since it states that bisections of L are the most problematic case, as one would expect. Suppose we have some other configuration with s significantly smaller than $n/2$. We will create a configuration where s is closer to $n/2$ and U is increased as follows: decrease n by 5, decrease a_0 by 4 and leave all other parameters unchanged. Observe that all four constraints are still satisfied. Now U is multiplied by a factor of

$$\frac{a_0^4}{(0.8n)^4} \frac{n^{15}}{(n-s)^{15}} (1 \pm o(1))\,.$$

Let us explain this. First, because s is significantly smaller than $n/2$ by constraint (3.7) we know that a_0 is large, so the change due to a_0 can indeed be approximated by a_0^4 (instead of the more accurate $a_0(a_0-1)(a_0-2)(a_0-3)$). In addition, the exponent of n is 15 (and not 20), because U also contains the $\binom{n}{s}$ factor.

Since we only care about sufficiently large n, it is enough to show that the limit of this value is greater than 1, or equivalently that $a_0/(0.8n) > (1-s/n)^{3.75}$.

Observe that from constraints (3.2), (3.3) and (3.5) we can conclude that $a_0 + a_5 \geq 0.3n$. Also, if $a_5 > a_0$, then we can subtract 1 from a_5 and add 1 to a_1. In this case, none of the four constraints is violated and U increases. Therefore, we can conclude that $a_0 \geq 0.15n$.

Now, let us distinguish two cases. If $s/n \geq 0.4$ then $(1-s/n)^{3.75} \leq 0.6^{3.75} \leq 0.148$. Therefore in this case we have $a_0/(0.8n) \geq 0.15n/(0.8n) = 0.1875 > (1-s/n)^{3.75}$.

If $s/n \leq 0.4$ then we use the fact that $(1-s/n)^{3.75}$ is a convex function of s. For $s=0$ the function has of course value 1, while for $s=0.4n$ it has value $0.6^{3.75} \leq 0.148 < 0.2$. The value of a convex function f in an interval $[a,b]$ is upper-bounded by the value of a linear function that coincides with f on points a and b. In this case we will overestimate the value of $(1-s/n)^{3.75}$ at $s=0.4n$ by pretending that it is 0.2, so the linear function that we get is $1-2s/n$. We thus get that $(1-s/n)^{3.75} \leq 1-2s/n$ for $s/n \in [0,0.4]$. But $a_0/(0.8n) \geq 1-2s/n$ by constraint (3.7), and we are done. □

Thanks to the above claim we can assume that essentially $s = n/2$, which makes constraint (3.7) vacuous. We will thus find an upper bound for U subject to constraints (3.2), (3.3) and (3.5).

Claim 3.4. *The function $U(n,s,c,a_i)$ when subject to constraints (3.2), (3.3), and (3.5) is maximized when $|a_i - a_{5-i}| \leq 1$, for $i \in \{0,1,\dots,5\}$.*

Proof. Suppose that $a_i \geq a_{5-i}+2$. Then decrease a_i by 1 and increase a_{5-i} by 1. This does not violate any of the three constraints and it increases U. □

Claim 3.5. *The function $U(n,s,c,a_i)$ when subject to constraints (3.2), (3.3), and (3.5) is maximized when $c = s-1$.*

Proof. Suppose that $c < s-1$. First, observe that it cannot be the case that $a_0 \leq a_1 \leq a_2$ and $a_5 \leq a_4 \leq a_3$, because then $a_0 + a_5 < 0.3n$ and constraint (3.5) will be violated. So suppose we have $a_0 > a_1$ (and the cases $a_1 > a_2$, $a_5 > a_4$ and $a_4 > a_3$ are similar). Then decrease a_0 by 1, increase a_1 by 1 and increase c by 1. This increases U without violating any constraint. □

Note that the claims given above determine the parameters within constant additive terms. Because perturbing any of the parameters by a small additive constant cannot affect U by more than a multiplicative polynomial (in n) factor, and since the upper bound we will prove will be exponential in n, we can disregard these constants. In other words, we will assume, without loss of generality, that because of the above claims we have $c = s = n/2$ and $a_i = a_{5-i}$.

Our problem now becomes

$$\max U(n,s,c,a_i) = \frac{(0.8n)!}{\prod_{i=0}^{2}(a_i!)^2} 5^{2a_1} 10^{2a_2} \frac{(2n)!(2n)!}{(4n)!} \frac{n!}{(n/2)!^2}$$

s.t.

$$a_0 + a_1 + a_2 = 0.4n \tag{3.8}$$

$$a_1 + 2a_2 = 0.25n \tag{3.9}$$

We can use the new constraints to get $a_1 = 0.25n - 2a_2$ and $a_0 = 0.15n + a_2$. We are trying to maximize the part of U that depends on a_0, a_1, a_2, so we want to maximize

$$\frac{5^{a_1} 10^{a_2}}{a_0! a_1! a_2!} = \frac{0.4^{a_2}}{(0.15n + a_2)!(0.25n - 2a_2)! a_2!}.$$

What is the value of a_2 that maximizes the above quantity? Observe that if we increase a_2 by 1, the above quantity changes by a factor of

$$\frac{0.4(0.25n - 2a_2)^2}{a_2(0.15n + a_2)}(1 \pm o(1)).$$

For the maximizing value of a_2 this factor must tend to 1, so that neither increasing nor decreasing a_2 by 1 can improve U. We therefore get $0.4(0.25n - 2a_2)^2 = a_2(0.15n + a_2)$. Solving this for a_2 we get either $a_2 \approx 0.04796n$, or a root that is larger than $0.8n$, which we can ignore. From the first root we get $a_0 \approx 0.19796n$ and $a_1 \approx 0.15407n$.

We can now plug in the above values into U. Using Stirling formula we can approximate $\log(n!)$ with $n \log\left(\frac{n}{\mathrm{e}}\right) + o(n)$, so we get

$$\begin{aligned} \log U &= 0.8n \log\left(\frac{0.8n}{\mathrm{e}}\right) - \sum_{i=0}^{2} 2a_i \log\left(\frac{a_i}{\mathrm{e}}\right) + 2a_1 \log 5 + 2a_2 \log 10 + 4n \log\left(\frac{2n}{\mathrm{e}}\right) - 4n \log\left(\frac{4n}{\mathrm{e}}\right) \\ &\quad + n \log\left(\frac{n}{\mathrm{e}}\right) - n \log\left(\frac{n}{2\mathrm{e}}\right) \\ &= 0.8n \log 0.8 - \sum_{i=0}^{2} 2a_i \log\left(\frac{a_i}{n}\right) + 2a_1 \log 5 + 2a_2 \log 10 - 3n < -0.045n. \end{aligned}$$

Thus, for sufficiently large n the probability that there exists a bad set is at most $2^{-0.045n + o(n)}$, which tends to 0. This completes the proof of Theorem 3.2. □

We now use the above theorem to construct a system of equations where each variable appears exactly 5 times. Our technique is pretty standard (in fact, essentially identical arguments are used in [3]). First, we may assume that in I_1 the number of appearances of each variable is a multiple of 5 (otherwise, repeat all equations five times). Also, by repeating all the equations we can make sure that all variables appear at least B' times, where B' is a sufficiently large number to make Theorem 3.2 hold.

For each variable x_i in I_1 we introduce the variables $x_{(i,j)}, j \in [d(i)]$ and $y_{(i,j)}, j \in [0.8d(i)]$ where $d(i)$ is the number of appearances of x_i in the original instance. We call

$$Z_i = \{x_{(i,j)} \mid j \in [d(i)]\} \cup \{y_{(i,j)} \mid j \in [0.8d(i)]\}$$

the cloud that corresponds to x_i. We will also use M to denote the set of the $x_{(i,j)}$ variables, which we call main variables, and C to denote the set of the $y_{(i,j)}$ variables, which we call checker variables. Construct a bipartite graph with the property described in Theorem 3.2 with $L = [d(i)]$, $R = [0.8d(i)]$ (since $d(i)$ is a constant that depends only on ε this can be done in constant time by brute force). For each edge $(j,k) \in E$ introduce the equation $x_{(i,j)} + y_{(i,k)} = 1$. Finally, for each equation $x_{i_1} + x_{i_2} + x_{i_3} = b$ in I_1, where this is the j_1-th appearance of x_{i_1}, the j_2-th appearance of x_{i_2} and the j_3-th appearance of x_{i_3} replace it with the equation $x_{(i_1,j_1)} + x_{(i_2,j_2)} + x_{(i_3,j_3)} = b$.

Denote this instance by I_2. Since we started with an instance with m equations of size 3, we had in total $3m$ variable appearances, each of which has been replaced by a new variable $x_{(i,j)}$. We have also introduced $0.8 \times 3m = 2.4m$ variables $y_{(i,j)}$. Each $x_{(i,j)}$ variable will now appear in 4 equations of size 2 (because the bipartite amplifier is 4-regular on one side). Thus, $|I_2|$ has $5.4m$ variables and $13m$ equations in total, with $12m$ of these equations having size 2. Let us note that this problem, where equations of different sizes are mixed, is called the Hybrid problem in the work of Berman and Karpinski [3]. A consistent assignment to a cloud Z_i is an assignment that sets all $x_{(i,j)}$ to b and all $y_{(i,j)}$ to $1-b$. By standard arguments using the graph of Theorem 3.2 we can show that an optimal assignment to I_2 is consistent as follows: suppose that we have an inconsistent assignment for a cloud Z_i. Let

$$S_0 = \{x_{(i,j)} \mid x_{(i,j)} = 0\} \cup \{y_{(i,j)} \mid y_{(i,j)} = 1\}$$

and let $S_1 = Z_i \setminus S_0$. In other words, S_0 contains the variables consistent with a 0 assignment to the $x_{(i,j)}$ variables and S_1 those consistent with a 1 assignment. Any size-two equation containing exactly one variable from S_0 and one from S_1 is violated by this assignment. By the properties of the graph, the number of such equations is at least as large as the number of $x_{(i,j)}$ vertices in either S_0 or S_1. Thus, we can flip the assignment of all the variables in one of the two sets and not decrease the number of satisfied equations.

Summary: Let us now summarize the properties of the instance I_2 we have constructed that we will need later.

- The variable set is partitioned into the set M of main variables $x_{(i,j)}$ and the set C of checker variables $y_{(i,j)}$. We have $|M| = 3m$, $|C| = 2.4m$.
- Each variable appears five times.
- There are m equations of size three and $12m$ equations of size two.
- The $12m$ equations of size two can be partitioned into n "clouds." Each cloud consists of equations of the form $x_{(i,j)} + y_{(i,j')} = 1$.
- There is an optimal assignment that satisfies all equations of size two.

Theorem 3.6. *Given a* MAX-LIN2 *instance I_2 as above, for all $\varepsilon > 0$ it is* NP*-hard to distinguish whether the maximum number of satisfiable equations is at least $(13-\varepsilon)m$ or at most $(12.5+\varepsilon)m$.*

3.2 MAX-1-IN-3-SAT

In the MAX-1-IN-3-SAT problem we are given a collection of clauses $(\ell_i \vee \ell_j \vee \ell_k)$, each consisting of at most three literals, where each literal is either a variable or its negation. A clause is satisfied by a truth assignment if exactly one of its literals is set to True. The problem is to find an assignment that satisfies the maximum number of clauses.

We would like to produce a MAX-1-IN-3-SAT instance from I_2. Observe that it is easy to turn the size two equations $x_{(i,j)} + y_{(i,k)} = 1$ to the equivalent clauses $(x_{(i,j)} \vee y_{(i,k)})$. We only need to worry about the m equations of size three.

If the k-th size-three equation of I_2 is $x_{(i_1,j_1)} + x_{(i_2,j_2)} + x_{(i_3,j_3)} = 1$ we introduce three new auxiliary variables $a_{(k,i)}, i \in [3]$ and replace the equation with the three clauses

$$(x_{(i_1,j_1)} \vee a_{(k,1)} \vee a_{(k,2)}), \quad (x_{(i_2,j_2)} \vee a_{(k,2)} \vee a_{(k,3)}), \quad (x_{(i_3,j_3)} \vee a_{(k,1)} \vee a_{(k,3)}).$$

If the right-hand-side of the equation is 0 then we add the same three clauses except we negate $x_{(i_1,j_1)}$ in the first clause. We call these three clauses the cluster that corresponds to the k-th equation.

It is not hard to see that if we fix an assignment to $x_{(i_1,j_1)}, x_{(i_2,j_2)}, x_{(i_3,j_3)}$ that satisfies the k-th equation of I_2 then there exists an assignment to $a_{(k,1)}, a_{(k,2)}, a_{(k,3)}$ that satisfies the whole cluster. Otherwise, at most two of the clauses of the cluster can be satisfied. Furthermore, in this case there exist three different assignments to the auxiliary variables that satisfy two clauses and each leaves a different clause unsatisfied.

We will denote by A the set of (auxiliary) variables $a_{(k,i)}$. Call the instance of MAX-1-IN-3-SAT we have constructed I_3. Note that it consists of $15m$ clauses: the $12m$ equations of size two from I_2 give $12m$ clauses, while the m equations of size three give 3 clauses each. The new instance I_3 contains $8.4m$ variables: the $5.4m$ variables of I_2 plus 3 new auxiliary variables for each size-three equation of I_2.

Summary: Let us now summarize the properties of the instance I_3 we have constructed that we will need later.

- The variable set is partitioned into the set M of main variables $x_{(i,j)}$, the set C of checker variables $y_{(i,j)}$ and the set A of auxiliary variables $a_{(k,i)}$. We have $|M| = 3m$, $|C| = 2.4m$ and $|A| = 3m$.
- Each variable of A appears twice. Each variable of $M \cup C$ appears five times. Each variable appears negated at most once.
- There are $3m$ clauses of size three and $12m$ clauses of size two.
- The $3m$ clauses of size three can be partitioned into m "clusters." Each cluster consists of three clauses of the form

  $$(x_{(i_1,j_1)} \vee a_{(k,1)} \vee a_{(k,2)}), \quad (x_{(i_2,j_2)} \vee a_{(k,2)} \vee a_{(k,3)}), \quad (x_{(i_3,j_3)} \vee a_{(k,1)} \vee a_{(k,3)}),$$

  with the possibility that the first literal of the first of these clauses may be negated.
- The $12m$ clauses of size two can be partitioned into n "clouds." Each cloud consists of clauses of the form $(x_{(i,j)} \vee y_{(i,j')})$.

- There is an optimal assignment that satisfies all clauses of size two.

Theorem 3.7. *Given a* MAX-1-IN-3-SAT *instance* I_3 *as above, for all* $\varepsilon > 0$ *it is* NP*-hard to distinguish if the maximum number of satisfiable clauses is at least* $(15-\varepsilon)m$ *or at most* $(14.5+\varepsilon)m$.

4 TSP

4.1 Construction

We now describe a construction that encodes I_3 into a TSP instance $G(V,E)$. Rather than viewing this as a generic construction from MAX-1-IN-3-SAT to TSP, we will at times need to use facts that stem from the special structure of I_3, as summarized at the end of the previous section.

As mentioned, we assume that in the graph $G(V,E)$ we may include some forced edges, that is, edges that have to be used at least once in any tour. The graph includes a central vertex, which we will call s. For each variable in $x \in M \cup C \cup A$ we introduce two new vertices named x^L and x^R, which we will call the left and right terminal associated with x. We add a forced edge from each terminal to s. For terminals that correspond to variables in $M \cup C$ this edge has weight $7/4$, while for variables in A it has weight $1/2$. We also add two (parallel) non-forced edges between each pair of terminals representing the same variable, each having a weight of 1 (we will later break down at least one from each pair of these, so the graph we will obtain in the end will be simple, that is, it will not have parallel edges). Informally, these two edges encode an assignment to each variable: we arbitrarily label one the True edge and the other the False edge, the idea being that a tour should pick exactly one of these for each variable and that will give us an assignment. We will re-route these edges through the clause gadgets as we introduce them, depending on whether each variable appears in a clause positive or negative.

Now, we add some gadgets to encode the size-two clauses of I_3. The intuition here, which will also apply to size-three clause gadgets, is that we construct a vertex for each appearance of a literal in a clause and use forced edges to turn the vertices from the same clause into a connected component. Then, it will be most economical for the tour to visit each such component exactly once, thus simulating the 1-in-3 predicate. Let us now give more details.

Let $(x_{(i,j_1)} \vee y_{(i,j_2)})$ be a clause of I_3 and suppose that this is the k_1-th clause that contains $x_{(i,j_1)}$ and the k_2-th clause that contains $y_{(i,j_2)}$, $k_1, k_2 \in [5]$. Then we add two new vertices to the graph, call them $x^{k_1}_{(i,j_1)}$ and $y^{k_2}_{(i,j_2)}$. Add two forced edges between them, each of weight $3/2$ (recall that forced edges represent long paths, so these are not really parallel edges). Finally, re-route the True edges incident on $x^L_{(i,j_1)}$ and $y^L_{(i,j_2)}$ through $x^{k_1}_{(i,j_1)}$ and $y^{k_2}_{(i,j_2)}$ respectively. More precisely, if the True edge incident on $x^L_{(i,j_1)}$ connects it to some other vertex u, remove that edge from the graph and add an edge from $x^L_{(i,j_1)}$ to $x^{k_1}_{(i,j_1)}$ and an edge from $x^{k_1}_{(i,j_1)}$ to u. All these edges have weight one and are non-forced (see Figure 1).

We use a similar gadget for clauses of size three (see Figure 2). Consider a cluster

$$(x_{(i_1,j_1)} \vee a_{(k,1)} \vee a_{(k,2)}), \quad (x_{(i_2,j_2)} \vee a_{(k,2)} \vee a_{(k,3)}), \quad (x_{(i_3,j_3)} \vee a_{(k,1)} \vee a_{(k,3)}),$$

and suppose for simplicity that this is the fifth appearance for all the main variables of the cluster. Then we add the new vertices $x^5_{(i_1,j_1)}, x^5_{(i_2,j_2)}, x^5_{(i_3,j_3)}$ and also the vertices $a^1_{(k,1)}, a^2_{(k,1)}, a^1_{(k,2)}, a^2_{(k,2)}$ and $a^1_{(k,3)}, a^2_{(k,3)}$.

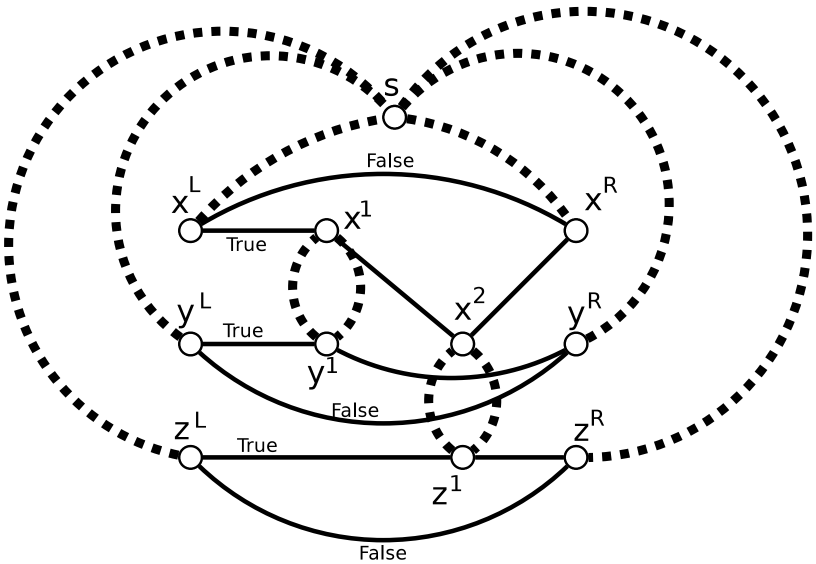

Figure 1: Example construction for the clause $(x \vee y) \wedge (x \vee z)$. Forced edges are denoted by dashed lines. There are two terminals for each variable and two gadgets that represent the two clauses. The True edges incident on the terminals are re-routed through the gadgets where each variable appears positive. The False edges connect the terminals directly since no variable appears anywhere negated. Forced edges incident on s have weight $7/4$. All other forced edge in this figure have weight $3/2$. All non-forced edges have weight 1.

To encode the first clause we add two forced edges of weight $5/4$, one from $x^5_{(i_1,j_1)}$ to $a^1_{(k,1)}$ and one from $x^5_{(i_1,j_1)}$ to $a^1_{(k,2)}$. We also add a forced edge of weight 1 from $a^1_{(k,1)}$ to $a^1_{(k,2)}$, thus making a triangle with the forced edges (see Figure 2). We re-route the True edge from $a^L_{(k,1)}$ through $a^1_{(k,1)}$ and $a^2_{(k,1)}$. We do similarly for the other two auxiliary variables and the main variables. Finally, for a cluster where $x_{(i_1,j_1)}$ is negated, we use the same construction except that rather than re-routing the True edge that is incident on $x^L_{(i_1,j_1)}$ we re-route the False edge. This completes the construction.

4.2 From assignment to tour

Let us now prove one direction of the reduction and in the process also give some intuition about the construction. Call the graph we have constructed $G(V,E)$.

Lemma 4.1. *If there exists an assignment to the variables of I_3 that leaves at most k equations unsatisfied, then there is a tour of G with cost at most $T = L + k$, where $L = 91.8m$.*

Proof. Observe that by construction we may assume that all the unsatisfied clauses of I_3 are in the clusters and that at most one clause in each cluster is unsatisfied, otherwise we can obtain a better assignment. Also, if an unsatisfied clause has all literals set to False we can flip the value of one of the auxiliary variables without increasing the number of violated clauses, since each auxiliary variable appears only

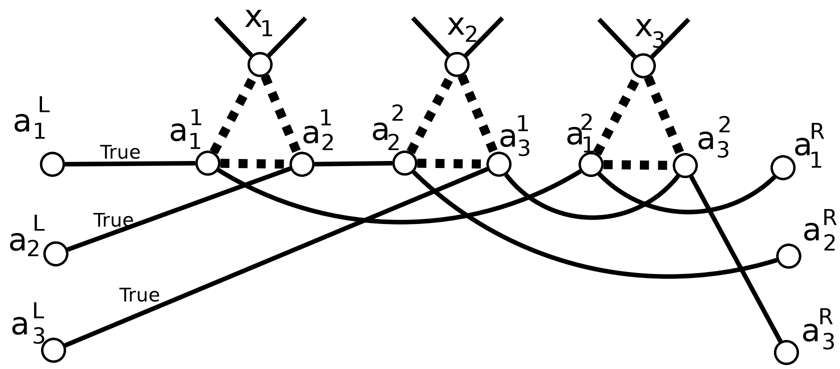

Figure 2: Example construction fragment for the cluster $(x_1 \vee a_1 \vee a_2) \wedge (x_2 \vee a_2 \vee a_3) \wedge (x_3 \vee a_1 \vee a_3)$. The False edges which connect each pair of terminals and the forced edges that connect terminals to s are not shown. The forced edges connecting a_i^j to x_k have weight $5/4$, while the third forced edge of each triangle has weight 1.

in two clauses. Thus, we may assume that all clauses have a True literal. Also, we may assume that no clause has all literals set to True: suppose that a clause does, then both auxiliary variables of the clause are True. We set them both to False, gaining one clause. If this causes the two other clauses of the cluster to become unsatisfied, set the remaining auxiliary variable to True. We conclude that all clauses have either one or two True literals.

Our tour uses all forced edges exactly once. For each variable x set to True in the assignment the tour selects the True edge incident on the terminal corresponding to x. If the edge has been re-routed all its pieces are selected, so that we have selected edges that make up a path from x^L to x^R. Otherwise, if x is set to False in the assignment the tour selects the corresponding False path.

Observe that this is a valid quasi-tour because all vertices have even degree (for each terminal we have selected the forced edge plus one more edge, for gadget vertices we have selected the two forced edges and possibly the two edges through which True or False was re-routed). Also, observe that the tour must be connected, because each clause contains a True literal, therefore for each gadget two of its external edges have been selected and they are part of a path that leads to the terminals.

The cost of the tour is at most $F+N+M+k$, where F is the total cost of all forced edges in the graph and N,M are the total number of variables and clauses respectively in I_3. To see this, notice that there are $2N$ terminals, and there is one edge incident on each and there are M clause gadgets, $M-k$ of which have two selected edges incident on them and k of which have four. Summing up, this gives $2N+2M+2k$, but then each unit-weight edge has been counted twice, meaning that the non-forced edges have a total cost of $N+M+k$.

Finally, we have $N = 8.4m$, $M = 15m$ and $F = 3\times 12m + (7/2)\times 3m + (7/2)\times 5.4m + 1\times 3m = 68.4m$, where the terms are respectively the cost of size-two clause gadgets, the cost of size-three clause gadgets, the cost of edges connecting terminals to s for the main variables, the checker variables and for the auxiliary variables. We have $F+N+M = 91.8m$. □

4.3 From tour to assignment

We would like now to prove the converse of Lemma 4.1, namely that if a tour of cost $L+k$ exists then we can find an assignment that leaves at most k clauses unsatisfied. Let us first give some high-level intuition and in the process justify the weights we have selected in our construction.

Informally, we could start from a simple base case: suppose that we have a tour such that all edges of G are used at most once. It is not hard to see that this then corresponds to an assignment, as in the proof of Lemma 4.1. So, the problem is how to avoid tours that may use some edges twice.

To this end, we first give some local improvement arguments that make sure that the number of problematic edges, which are used twice, is limited. However, arguments like these can only take us so far, and we would like to avoid having too much case analysis.

We therefore try to isolate the problem. For variables in $M \cup C$ which the tour treats honestly, that is, variables which do not appear in clauses whose corresponding gadgets have edges used twice, we directly obtain an assignment from the tour. For the other variables in $M \cup C$ we pick a random value and then extend the whole assignment to A in an optimal way. We want to show that the expected number of unsatisfied clauses is at most k.

The first point here is that if a clause containing only honest variables turns out to be violated, the tour must also be paying an extra cost for it. The difficulty is therefore concentrated on clauses with dishonest variables.

By using some edges twice the tour is paying some cost on top of what is accounted for in L. We would like to show that this extra cost is larger than the number of clauses violated by the assignment. It is helpful to think here that it is sufficient to show that the tour pays an additional cost of $5/2$ for each dishonest variable. This cost is enough since main and checker variables appear 5 times. By setting dishonest variables randomly we will satisfy half the clauses that contain them in expectation: clauses of size two are satisfied with probability $1/2$ if at least one of their variables is randomly set, while for clauses of size three we assign the auxiliary variables *after* randomly assigning the main variables, thus a whole cluster is satisfied if and only if the parity equation from which it was derived is satisfied and this happens with probability $1/2$.

A crucial point now is that, by a simple parity argument, there has to be an even number of violations (that is, edges used twice) for each variable (Lemma 4.4). This explains the weights we have picked for the forced edges in size-three gadgets ($5/4$) and for edges connecting terminals to s ($7/4 = 5/4 + 1/2$ or $5/4$ extra to the cost already included in L for fixing the parity of the terminal vertex). Two such violations give enough extra cost to pay for the expected number of unsatisfied clauses containing the variable.

At this point, we could also set the weights of forced edges in size-two gadgets to $5/2$, which would be split among the two dishonest variables giving $5/4$ to each. Then, any two violations would have enough additional cost to pay for the expected unsatisfied clauses. However, we are slightly more careful here: rather than setting all dishonest variables in $M \cup C$ independently at random, we pick a random but consistent assignment for each cloud. This ensures that all size-two clauses with violations will be satisfied. Thus, it is sufficient for violations in them to have a cost of $3/2$: the amount “paid” to each variable is now $3/4 = 5/4 - 1/2$, but the expected number of unsatisfied clauses with this variable is also decreased by $1/2$ since one clause is surely satisfied.

Let us now proceed to give the full details of the proof. Recall that if a tour of a certain cost exists, then there exists also a quasi-tour of the same cost. It suffices then to prove the following:

Lemma 4.2. *If there exists a quasi-tour of G with cost at most $L+k$ then there exists an assignment to the variables of I_3 that leaves at most k clauses unsatisfied.*

In order to prove Lemma 4.2 it is helpful to first make some easy observations. First, recall Lemma 2.1, which states that in an optimal quasi-tour all unit-weight edges are used at most once. All non-forced edges in our construction have unit weight and are therefore used at most once.

Second, if both forced edges of a gadget of size two are used twice then we can remove one appearance of each from the solution, decreasing the cost. Similarly, in a gadget of size three if two forced edges are used twice then we can drop one copy of each and use the third edge twice, making the tour cheaper. Therefore, in each gadget there is at most one forced edge that is used twice.

Third, if both forced edges that connect the terminals x^L, x^R to s are used twice, then we can remove one appearance of each from the solution and replace them by the shortest path from x^L to x^R that uses only non-forced unit weight edges. This has weight at most one for the auxiliary variables and two for the rest (since no variable appears negated more than once), which in both cases is at most as much as the weight of the removed edges. Therefore, for each variable x, at least one of the forced edges that connect x^L, x^R to s is used exactly once.

Given a tour E_T, we will say that a variable x is honestly traversed in that tour if all the forced edges that involve it are used exactly once (this includes the forced edges incident on x^L, x^R and x^i, $i \in [5]$).

Let us now give two more useful facts.

Lemma 4.3. *There exists an optimal tour where all forced edges between two different vertices that correspond to two variables in A are used exactly once.*

Proof. We refer the reader again to Figure 2. Suppose we have an optimal tour that does not satisfy this property. We will transform it into a tour of the same or lower cost that does. Assume that the edge (a_1^1, a_2^1) is used twice (the other cases are equivalent by symmetry since all vertices a_i^j are connected to one terminal and one other such vertex).

First, suppose that at least one of the edges that connect one of these two endpoints to a terminal is selected, say the edge (a_1^L, a_1^1). Then modify the solution by removing that edge and a copy of the duplicate forced edge and adding a copy of (a_2^L, a_2^1), (s, a_2^L) and (s, a_1^L). This does not increase the cost.

Second, suppose that both (s, a_1^L) and (s, a_2^L) are used twice in the tour. Then we can modify the tour by dropping one copy of each and a copy of the duplicate gadget edge and adding (a_1^L, a_1^1) and (a_2^L, a_2^1).

Finally, suppose that none of the previous two cases is true. Thus, neither of (a_1^L, a_1^1), (a_2^L, a_2^1) is used in the tour. This means that (a_1^1, a_1^2) and (a_2^1, a_2^2) are both used to ensure that a_1^1 and a_2^1 have even degree. Also, one of the edges connecting a terminal to s is used once, say (s, a_1^L). This means that the False edge incident to a_1^L must be used to make the degree of a_1^L even. Remove the False edge and the edge (a_1^1, a_1^2) from the tour and add the edges (a_1^L, a_1^1) and (a_1^R, a_1^2). This reduces to the first case. □

Lemma 4.4. *In an optimal tour, if a variable is dishonest then it must be dishonest twice. More precisely, the number of forced edges that involve the variable (either inside gadgets or connecting terminals to s) and are used twice must be even.*

Proof. Consider a variable x and first suppose that neither of the forced edges connecting s to the terminals is used twice, but there is a single forced edge in a gadget that is used twice. It follows that the vertex that corresponds to x in that gadget has an odd number of unit-weight edges incident to it selected. The two terminals have a single selected unit-weight edge incident on them and all other vertices that belong to x have an even number of incident unit-weight edges selected, since their total degree is even. Thus, summing the number of selected unit-weight edges incident on all the vertices that belong to x we get an odd number, which is a contradiction since we counted each such edge exactly twice. A similar argument applies if one assumes that one of the forced edges incident on the terminals is used twice and all other forced edges are used once. □

Observe that it follows from Lemmata 4.3 and 4.4 that if all the main variables involved in a cluster are honest then the auxiliary variables of that cluster are also honest. This holds because if the main variables are honest then by Lemma 4.3 no forced edge inside the gadgets of the cluster is used twice, so by Lemma 4.4 and the fact that at least one of the forced edges incident on the terminals is used once, the auxiliary variables are honest.

We would like now to be able to extract a good assignment even if a tour is not honest, thus indirectly proving that honest tours are optimal.

Proof of Lemma 4.2. Consider the following algorithm to extract an assignment from the tour: first, for each variable in $M \cup C$ that was traversed honestly give it the same truth-value as in the tour, that is, if the tour selects the True edge incident on the corresponding terminal, set the variable to True, otherwise to False. To decide on the value of the dishonest variables from $M \cup C$ produce n random bits $b_i, i \in [n]$ (recall that n is the number of variables of I_1, or the number of clouds in I_2). For each i set all dishonest variables $x_{(i,j)}$ to be equal to b_i and all dishonest $y_{(i,j)}$ to be equal to $1 - b_i$. This ensures that size-two clauses that contain two dishonest variables are always satisfied, since these clauses are always between two variables of the same cloud.

Let us also assign the auxiliary variables. If there is an assignment to the auxiliary variables of a cluster that satisfies all three clauses select it. Otherwise, select an assignment that violates the clause of a dishonest variable from M, if such a variable exists, and satisfies the other two. If all main variables are honest, as we have argued the auxiliary variables are also honest, so pick the corresponding assignment.

We now have a randomized assignment for I_3, so let us upper-bound the expected number of unsatisfied clauses. Let U be a random variable equal to the set of unsatisfied clauses and let $U = U_1 \cup U_2$ where U_1 contains all the unsatisfied clauses that involve only honest variables from $M \cup C$ and U_2 the rest. (Note that U_1 is not random.)

The cost T of the quasi-tour we have is $\leq F + N + M + k$, where we remind the reader that F is the cost of forced edges, N is the number of variables of I_3 and M the number of clauses. Let E_G be the set of forced gadget edges that the tour uses twice (each of these is included once in E_G). Let E_S be the set of forced edges incident on s that the tour uses twice. Let E_1 be the set of unit-weight edges that the tour uses (recall that each is used once). Let U_1' be the set of clauses that correspond to gadgets the tour visits at least twice (meaning they have at least four incident edges selected). Let U_1'' be the set of clauses that correspond to gadgets the tour does not visit (meaning that each forms its own connected component).

We have

$$T = \sum_{e \in E_T} w(e) + 2(c(G_T) - 1) = F + \sum_{e \in E_1} w(e) + \sum_{e \in E_G} w(e) + \sum_{e \in E_S} w(e) + 2(c(G_T) - 1)\,.$$

By definition $\sum_{e \in E_1} w(e) = |E_1|$. Let us try to lower-bound this quantity using arguments similar to the proof of Lemma 4.1. After the selection of the forced edges there are $2N - |E_S|$ terminals with odd degree, so each has a selected unit-weight edge incident to it. There are $|U_1'|$ gadgets with at least four selected incident edges and $M - |U_1'| - |U_1''|$ gadgets with two selected incident edges. Summing up we get $2N - |E_S| + 2M + 2|U_1'| - 2|U_1''|$, but each edge is counted twice, so we have

$$|E_1| \geq N - \frac{1}{2}|E_S| + M + |U_1'| - |U_1''|\,.$$

Using this fact we get

$$T \geq F + N + M + \sum_{e \in E_G} w(e) + \sum_{e \in E_S} \left(w(e) - \frac{1}{2} \right) + |U_1'| + 2(c(G_T) - 1) - |U_1''|\,.$$

Now, observe that $|U_1''| \leq c(G_T) - 1$, because each element of U_1'' forms a component and there is one component that is not an element of U_1'' (the one that contains s). Thus, $2(c(G_T) - 1) - |U_1''| \geq |U_1''|$. Combining this with the above we get

$$T \geq F + N + M + \sum_{e \in E_G} w(e) + \sum_{e \in E_S} \left(w(e) - \frac{1}{2} \right) + |U_1'| + |U_1''|\,.$$

Given the known upper-bound on the cost of the tour we have that

$$k \geq \sum_{e \in E_G} w(e) + \sum_{e \in E_S} \left(w(e) - \frac{1}{2} \right) + |U_1'| + |U_1''|\,.$$

We now need to argue two facts and we are done. First $|U_1| \leq |U_1'| + |U_1''|$. Recall that U_1 is the set of unsatisfied clauses that involve honest variables. Since the variables are traversed honestly their corresponding gadgets are either visited at least twice or not at all, so they are counted in $|U_1'|$ or in $|U_1''|$.

Second, we would like to show that

$$\mathbf{E}[|U_2|] \leq \sum_{e \in E_G} w(e) + \sum_{e \in E_S} \left(w(e) - \frac{1}{2} \right)\,.$$

Before we do that, observe that if we show this then it follows that $\mathbf{E}[|U|] = \mathbf{E}[|U_2|] + |U_1| \leq k$, so there must exist an assignment that leaves no more than k clauses unsatisfied and we are done.

So, let us try to upper-bound $\mathbf{E}[|U_2|]$, which is the expected number of unsatisfied clauses that contain a dishonest variable. First, observe that if there are dishonest auxiliary variables in a cluster by the construction of the assignment we have ensured that any unsatisfied clause must contain a dishonest main variable. Therefore, it suffices to count the expected number of unsatisfied clauses that contain a dishonest main or checker variable.

Let us define a credit $\mathrm{cr}(x)$ for each dishonest main or checker variable x. If a forced edge connecting a terminal to s is used twice we give x a credit of $5/4$ (which is equal to $w(e)-1/2$, since these edges have weight $7/4$). If a forced edge in a gadget that involves x and another main or checker variable is used twice we give x a credit of $3/4$ (which is equal to $w(e)/2$). Finally, if a forced edge in a gadget that involves x and an auxiliary variable is used twice we give x a credit of $5/4$ (which is equal to $w(e)$). We define $\mathrm{cr}(x)$ to be the sum of credits given to x in this process.

If $D \subseteq M \cup C$ is the set of dishonest main and checker variables then it is not hard to see that

$$\sum_{x\in D} \mathrm{cr}(x) \le \sum_{e\in E_G} w(e) + \sum_{e\in E_S} (w(e)-1/2)\,.$$

All edges are counted once in the sum of credits, except for those from E_G that involve two main variables, for which each is credited half the weight.

We will now argue that the expected number of unsatisfied clauses that contain a variable x is at most $\mathrm{cr}(x)$. Recall that clauses containing x and another dishonest main or checker variable are by construction satisfied, while clauses made up of x and one honest variable are satisfied with probability $1/2$. Also, clauses of size 3 that contain x are satisfied with probability at least $1/2$, since with probability $1/2$ the equation from which the cluster was obtained is satisfied. Thus, if $\mathrm{cr}(x) \ge 5/2$ we are done, because x appears in 5 clauses and by the random assignment we will in expectation satisfy at least $5/2$ clauses. We know that x received at least two credits by Lemma 4.4, so $\mathrm{cr}(x) \ge 3/2$, as the smallest credit is $3/4$. If $\mathrm{cr}(x) = 3/2$ then x must have received two credits that were shared with other dishonest variables. Therefore, there are two clauses containing x which are surely satisfied, and out of the other three the expected number of unsatisfied clauses is $3/2 \le \mathrm{cr}(x)$. Similarly, if $\mathrm{cr}(x) = 2$, then x shared a credit with another variable at least once, so one clause is surely satisfied and the expected number of unsatisfied clauses out of the other four is 2.

We therefore have

$$\mathbf{E}[|U_2|] \le \sum_{x\in D} \mathrm{cr}(x) \le \sum_{e\in E_G} w(e) + \sum_{e\in E_S} (w(e)-1/2)$$

and this concludes the proof. □

5 Conclusions

We have given an alternative and (we believe) simpler inapproximability proof for TSP, also modestly improving the known bound. We believe that the approach followed here where the hardness proof goes explicitly through bounded-occurrence CSPs, is more promising than the somewhat ad-hoc method of [16], not only because it is easier to understand but also because we stand to gain almost "automatically" from improvements in our understanding of the inapproximability of bounded-occurrence CSPs. In particular, though we used the 5-regular amplifiers from [3], any such amplifier would work essentially "out of the box," and any improved construction could imply an improvement in our bound. In fact, in recent work [10] the hardness bound for TSP has been further improved, in part by substituting the 5-regular amplifiers with a new 3-regular amplifier construction.

Nevertheless, the distance between the upper and lower bounds on the approximability of TSP remains quite large and it seems that some major new idea will be needed to close it.

References

[1] PIOTR BERMAN AND MAREK KARPINSKI: On some tighter inapproximability results (extended abstract). In *Proc. 26th Internat. Colloq. on Automata, Languages and Programming (ICALP'99)*, volume 1644 of *LNCS*, pp. 200–209. Springer, 1999. [doi:10.1007/3-540-48523-6_17] 218, 220

[2] PIOTR BERMAN AND MAREK KARPINSKI: Efficient amplifiers and bounded degree optimization. *Electron. Colloq. on Comput. Complexity (ECCC)*, 8(53), 2001. ECCC. 218, 220

[3] PIOTR BERMAN AND MAREK KARPINSKI: Improved approximation lower bounds on small occurrence optimization. *Electron. Colloq. on Comput. Complexity (ECCC)*, 10(8), 2003. ECCC. 218, 220, 221, 224, 225, 234

[4] HANS-JOACHIM BÖCKENHAUER, JURAJ HROMKOVIČ, RALF KLASING, SEBASTIAN SEIBERT, AND WALTER UNGER: An improved lower bound on the approximability of metric TSP and approximation algorithms for the TSP with sharpened triangle inequality. In *Proc. 17th Ann. Symp. on Theoretical Aspects of Comp. Sci. (STACS'00)*, volume 1770 of *LNCS*, pp. 382–394. Springer, 2000. [doi:10.1007/3-540-46541-3_32] 218

[5] NICOS CHRISTOFIDES: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon University, 1976. 217

[6] LARS ENGEBRETSEN: An explicit lower bound for TSP with distances one and two. *Algorithmica*, 35(4):301–319, 2003. Preliminary version in STACS'99. [doi:10.1007/s00453-002-1001-6] 218

[7] LARS ENGEBRETSEN AND MAREK KARPINSKI: TSP with bounded metrics. *J. Comput. System Sci.*, 72(4):509–546, 2006. Preliminary version in ICALP'01. [doi:10.1016/j.jcss.2005.12.001] 218

[8] SHAYAN OVEIS GHARAN, AMIN SABERI, AND MOHIT SINGH: A randomized rounding approach to the traveling salesman problem. In *Proc. 52nd FOCS*, pp. 550–559. IEEE Comp. Soc. Press, 2011. [doi:10.1109/FOCS.2011.80] 217

[9] JOHAN HÅSTAD: Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. Preliminary version in STOC'97. [doi:10.1145/502090.502098] 218, 220

[10] MAREK KARPINSKI, MICHAEL LAMPIS, AND RICHARD SCHMIED: New inapproximability bounds for TSP. In *Proc. 24th Internat. Symp. on Symbolic and Algebraic Computation (ISAAC'13)*, volume 8283 of *LNCS*, pp. 568–578. Springer, 2013. See also at ECCC. [doi:10.1007/978-3-642-45030-3_53] 219, 234

[11] MAREK KARPINSKI AND RICHARD SCHMIED: On approximation lower bounds for TSP with bounded metrics. *Electron. Colloq. on Comput. Complexity (ECCC)*, 19(8), 2012. ECCC. See also at arXiv. 218

[12] MICHAEL LAMPIS: Improved inapproximability for TSP. In *Proc. 15th Internat. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'12)*, volume 1770

of *LNCS*, pp. 243–253. Springer, 2012. See also at arXiv. [doi:10.1007/978-3-642-32512-0_21] 217

[13] TOBIAS MÖMKE AND OLA SVENSSON: Approximating graphic TSP by matchings. Technical report, 2011. Preliminary version in FOCS'11. [arXiv:1104.3090] 217

[14] MARCIN MUCHA: $\frac{13}{9}$-approximation for graphic TSP. *Theory Comput. Syst.*, 2012. Preliminary version in STACS'12. [doi:10.1007/s00224-012-9439-7] 217

[15] CHRISTOS H. PAPADIMITRIOU AND SANTOSH VEMPALA: On the approximability of the traveling salesman problem (extended abstract). In *Proc. 32nd STOC*, pp. 126–133. ACM Press, 2000. [doi:10.1145/335305.335320] 218

[16] CHRISTOS H. PAPADIMITRIOU AND SANTOSH VEMPALA: On the approximability of the traveling salesman problem. *Combinatorica*, 26(1):101–120, 2006. Preliminary version in STOC'00. [doi:10.1007/s00493-006-0008-z] 218, 219, 234

[17] CHRISTOS H. PAPADIMITRIOU AND MIHALIS YANNAKAKIS: The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18(1):1–11, 1993. [doi:10.1287/moor.18.1.1] 218

[18] ANDRÁS SEBŐ AND JENS VYGEN: Shorter tours by nicer ears: 7/5-approximation for graphic TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. Technical report, 2012. To appear in Combinatorica. [arXiv:1201.1870] 217

AUTHOR

Michael Lampis
KTH Royal Institute of Technology
mlampis@kth.se
http://www.csc.kth.se/~mlampis

ABOUT THE AUTHOR

MICHAEL LAMPIS received his Ph. D. from the City University of New York in 2011 advised by Amotz Bar-Noy. His favorite research area is parameterized complexity, but it is not hard to get him interested in other branches of theoretical computer science. The two procrastination activities he most excels in are reading the New York Times and playing blitz chess matches online. When he grows up he wants to be a computer science professor or an astronaut.