Long Tours and Short Superstrings

(extended abstract)

S. Rao Kosaraju*
Dept. of Computer Science
The Johns Hopkins University
Baltimore, MD 21218-2694

James K. Park
Dept. of Computer Science
Dartmouth College
Hanover, NH 03755-3510

Clifford Stein†
Dept. of Computer Science
Dartmouth College
Hanover, NH 03755-3510

Abstract

This paper considers weight-maximizing variants of the classical symmetric and asymmetric traveling-salesman problems. Like their weight-minimizing counterparts, these variants are MAX SNP-hard. We present the first nontrivial approximation algorithms for these problems. Our algorithm for directed graphs finds a tour whose weight is at least $\frac{38}{63} \approx 0.603$ *times the weight of a maximum-weight tour, and our algorithm for undirected graphs finds a tour whose weight is at least* $\frac{5}{7} \approx 0.714$ *times optimal. These bounds compare favorably with the* $\frac{1}{2}$ *and* $\frac{2}{3}$ *bounds that can be obtained for undirected and directed graphs, respectively, by simply deleting the minimum-weight edge from each cycle of a maximum-weight cycle cover. Our algorithm for directed graphs can be used to improve several recent approximation results for the shortest-superstring problem.*

1 Introduction

The *traveling-salesman problem* (TSP) is one of the most famous and most fundamental problems in computer science. It has many practical applications, and it is also (presumably) quite difficult to solve, being one of the original NP-hard problems [10]. For these reasons, considerable research effort has been focused on designing good approximation algorithms for the TSP. Most of these approximation algorithms are concerned with the problem of finding a shortest (i.e., minimum-weight) tour, but in this paper, we will propose approximation algorithms for the problem of finding a longest tour. We call this latter problem the *maximum-weight traveling-salesman problem* (Max TSP). This problem has received some attention recently, as it has been shown that a good approximation algorithm for the directed Max TSP implies a good approximation algorithm for the *shortest-superstring problem*, which arises in DNA sequencing and data compression. We will also show that for certain minimum-weight traveling-salesman problems, a good approximation to a longest tour in a derived graph gives the best-known approximation to a shortest tour in the original graph.

Given a graph $G = (V, E)$ and a function $w : E \to \mathbf{R}^+$ assigning a nonnegative weight to each edge, even approximating a shortest tour is NP-hard [12]; thus, it is traditional to consider instances with edge weights satisfying the *triangle inequality*, which requires that for any three vertices i, j, and k, $w(i, j) \leq w(i, k) + w(k, j)$. Even with the triangle inequality, however, both the undirected (i.e., symmetric) and directed (i.e., asymmetric) versions of the problem are MAX SNP-hard [11], which means there are likely to be approximation bounds strictly greater than 1 that no (polynomial-time) approximation algorithm can achieve. For undirected graphs with edge weights satisfying the triangle inequality, the best positive result known is Christofides' algorithm [4], which guarantees a tour whose weight is at most $\frac{3}{2}$ times the weight of a minimum-weight tour. For directed graphs, the best algorithm [7] guarantees a tour whose weight is at most an $O(\log n)$ factor times optimal.

We are aware of very little work on the problem of finding a longest tour. On the negative side, the reduction in [11] also implies that both the undirected and directed variants of the Max TSP are MAX SNP-hard. The best positive results are due to Fisher, Nemhauser, and Wolsey [6]. They analyzed several different heuristics for both undirected and directed graphs. For directed graphs, the best of these heuristics (at least in terms of its worst-case approximation

*Supported by NSF Award CCR-9107293.
†Partly supported by NSF Award CCR-9308701, a Walter Burke Research Initiation Award, and a Dartmouth College Research Initiation Award.

bound) works by first finding a maximum-weight *cycle cover* for the graph in question. (A cycle cover is a set of cycles such that each vertex is in exactly one cycle.) It then discards the minimum-weight edge in each cycle and arbitrarily connects the resulting paths to form a tour. Since the weight in the cycle cover is an upper bound on the weight in an optimal tour, and the minimum-weight edge in each cycle contains at most half the total weight in the cycle, this algorithm yields a tour containing at least $\frac{1}{2}$ of the weight in an optimal tour. For undirected graphs, Fisher et al. use a maximum-weight 2-*matching* in place of the previous algorithm's cycle cover. Since a 2-matching is just a cycle cover containing only cycles of size three or more, the same analysis yields a $\frac{2}{3}$-approximation for the undirected case. We are also aware of a recent result by Barvinok [2] for graphs whose edge weights are distances between points in a Euclidean metric space. He describes an algorithm that, for any fixed ϵ, produces an estimate ℓ_{est} for the length of a longest tour; this estimate is guaranteed to satisfy $(1-\epsilon)\ell_{\mathrm{opt}} \leq \ell_{\mathrm{est}} \leq (1+\epsilon)\ell_{\mathrm{opt}}$, where ℓ_{opt} is the length of a longest tour. This algorithm does not actually produce a tour, however.

In this paper, we give the first non-trivial approximation algorithms for the Max TSP. Specifically, we give an algorithm that achieves a $\frac{38}{63}$-approximation in directed graphs and a related algorithm that achieves a $\frac{5}{7}$-approximation in undirected graphs. As in the simple approximation algorithms mentioned above, we always begin with a cycle cover. We then employ various graph coloring and matching techniques that allow us to trade edges in the cycle cover for edges outside the cycle cover such that the resulting set of edges form vertex-disjoint paths containing a significant fraction of an optimal tour's weight.

A motivating application of our algorithm for directed graphs is the shortest-superstring problem: given a set of strings $s_1, ..., s_m$ over an arbitrary alphabet, find a minimum-length string containing each s_i as a substring. As we mentioned earlier, this problem has applications in DNA sequencing and data compression, and it has recently attracted considerable attention [1, 3, 5, 13, 14, 15]. Testing whether there exists a superstring of a given length is NP-complete [8], and the optimization problem is MAX SNP-hard [3]. Blum, Jiang, Li, Tromp, and Yannakakis [3] established that the classical greedy heuristic achieves an approximation factor of 4. They also gave an algorithm that can guarantee a $(3-2\epsilon)$-approximation to the shortest superstring given a $(\frac{1}{2}+\epsilon)$-approximation algorithm for the directed Max TSP problem. Thus, our $\frac{38}{63}$-approximation algorithm implies a $176/63 \approx 2.793$ bound for the shortest-superstring problem. This bound is better than the previously published $26/9 \approx 2.889$ and $17/6 \approx 2.833$ bounds of Teng and Yao [14] and Czumaj, Gąsieniec, Piotrów, and Rytter [5], respectively, and it is almost as good as a recently discovered 2.75 bound of Armen and Stein [1]. Furthermore, our bound is achieved without exploiting any additional structure of the shortest-superstring problem beyond that used in Blum et al.'s reduction (whereas the results of [1, 5, 14] do rely on additional structure of this kind).

Recently, Papadimitriou and Yannakakis [11] gave a $\frac{7}{6}$-approximation algorithm for the problem of finding a minimum-length tour in an undirected graph where the edge weights are all either 1 or 2. If we apply our $\frac{5}{7}$-approximation algorithm for finding a maximum-length tour to their problem, we achieve a $\frac{9}{7}$-approximation, which is not quite as good. However, our techniques generalize to handle the case where all the edge weights are in a range between x and $2x$, for some positive x, while their techniques do not appear to generalize. Even more generally, consider an instance of the undirected TSP with edge weights between 1 and some constant a (that do not necessarily satisfy the triangle inequality). If we wish to approximate a minimum-length tour, we can replace each edge of weight w by one of weight $a - w$, and then apply our $\frac{5}{7}$-approximation algorithm for the undirected Max TSP. It is not difficult to see that this approach yields a $(\frac{5}{7} + \frac{2}{7}a)$-approximation for the original problem. If $a \leq \frac{11}{4}$, then this bound represents an improvement over Christofides' $\frac{3}{2}$ bound, and in general it compares quite favorably with the trivial bound of a. Furthermore, the result generalizes to the case where the weights are in some interval between x and ax. It also works for directed graphs; in particular, using our approximation algorithm for the directed Max TSP yields a $(\frac{38}{63} + \frac{25}{63}a)$-approximation for the weight-minimizing problem.

2 Approximating the Longest Tour in a Directed Graph

In this section, we present our $\frac{38}{63}$-approximation algorithm for the directed Max TSP. For simplicity, we will assume that the number of vertices in the graph is even throughout this extended abstract. Our algorithm will make use of two standard polynomial-time algorithms for graphs. A *cycle cover* for a graph is a set of cycles such that each vertex is included in exactly one cycle. The weight of a cycle cover is the sum of the weights of its edges. The first algorithm finds a maximum-weight cycle cover for an n-vertex directed graph in $O(n^3)$ time, using a well-known reduction of the cycle-cover problem to the assignment problem. The second algorithm finds a maximum-weight (though not necessarily perfect) matching for an n-vertex undirected graph in $O(n^3)$ time.

Our algorithm for the Max TSP has three versions, each using the following general framework, which is the same framework used by Fisher et al. [6] to obtain a $\frac{1}{2}$-approximation algorithm.

Generic Directed Max TSP Algorithm

Step 1. Compute a maximum-weight cycle cover $\mathcal{C}$ for G.

Step 2. Use $\mathcal{C}$ to obtain a set P of vertex-disjoint paths.

Step 3. Construct a tour by patching together the paths of P.

We will give three different implementations of Step 2. The first will be Fisher et al.'s approach of deleting the smallest edge in each cycle. The second and third approaches will be much more involved and use non-cycle edges in an attempt to find a set of paths with more weight.

To compare the three approaches, we need the following definitions. We call any cycle through exactly 2 vertices a *2-cycle* and any cycle through 3 or more vertices a *3^+-cycle*. Now let W denote the total weight of the edges in $\mathcal{C}$, let bW denote the weight of those edges that are the heavier (i.e., larger-weight) edge in some 2-cycle of $\mathcal{C}$, and let cW denote the weight of those edges that are the lighter (i.e., smaller-weight) edge in some 2-cycle. We will show that the first implementation of Step 1 yields a $(\frac{2}{3}+\frac{1}{3}(b-2c))$-approximation, the second yields a $(\frac{7}{12}-\frac{1}{12}(b-2c))$-approximation, and the third yields a $(\frac{2}{3}+\frac{4}{15}(b-2c))$-approximation. (We express each bound as a linear function of $b-2c$ in order to simplify the balancing of the bounds.) Simple algebra shows that the best of the three is always at least a $\frac{38}{63}$-approximation.

We remark that obtaining a lower bound on $w(P)$, the weight of the paths found in Step 2, is sufficient for bounding the weight of the tour we obtain, since our assumption of nonnegative edge weights implies that no matter how we connect the paths of P together, the resulting tour is always of weight at least $w(P)$. We thus ignore Step 3 in the remaining analysis.

2.1 Version 1

Version 1 is the simple $\frac{1}{2}$-approximation algorithm described in the introduction. We obtain P from $\mathcal{C}$ by just deleting the smallest edge in each cycle. Though $w(P)$ can be a small as $\frac{1}{2}W$ if $b=c$ and $\mathcal{C}$ contains only 2-cycles, it is often significantly better. Specifically, if we let pW denote the weight of the set consisting of the lightest edge from each 3^+-cycle, then $w(P)=(1-c-p)W$. Combining this quantity with the bound $p\leq\frac{1}{3}(1-b-c)$, which follows from the definition of p, we obtain the following lemma.

Lemma 1 Version 1 of the generic algorithm returns a $(\frac{2}{3}+\frac{1}{3}(b-2c))$-approximation to the Max TSP. ■

2.2 Version 2

We begin by giving a weaker variant of version 2 that contains some of the basic ideas. We will then give our stronger version that achieves the improved bound.

Version 2, Simple Variant

Step 2a. Construct an undirected graph G' from G and $\mathcal{C}$ as follows:

- Replace each 2-cycle $\{(u,v),(v,u)\}$ of $\mathcal{C}$ with an undirected edge (u,v) of weight 0.
- For all other pairs of vertices u and v, create an undirected edge (u,v) with weight $\max\{w(u,v),w(v,u)\}$.

Step 2b. Find a maximum-weight matching M in G'.

Step 2c. To the directed counterparts of the undirected edges of M, add one edge from each 3^+-cycle (taking care not to create any cycles in the corresponding undirected graph).

Step 2d. Contract the 2-cycles.

Step 2e. Two-color the remaining graph to obtain two sets of directed paths. Let P consist of those edges in the color class with more weight, together with one edge from each 2-cycle.

Let $T_{\rm opt}$ denote the optimal Max TSP tour, and let $w(T_{\rm opt})$ denote its weight. First we consider the weight of the edges of $T_{\rm opt}$ in the graph G'. Since we have eliminated 2-cycles, but not decreased the weight on any other edges, this weight is at least $w(T_{\rm opt})-bW$, as each 2-cycle can contribute at most one edge to $T_{\rm opt}$. Furthermore, the edges of $T_{\rm opt}$ can be partitioned into two matchings with average weight $\frac{1}{2}(w(T_{\rm opt})-bW)$.

Since the edges of $T_{\rm opt}$ in the graph G' are a subset of the edges in G', the matching we find in Step 2b must then have weight at least $\frac{1}{2}(w(T_{\rm opt}) - bW)$.

We next augment the graph with an edge from each 3^+-cycle. (It needs to be shown that we can always add such an edge without creating any cycles in the graph containing undirected versions of these edges; however, we omit the proof of this claim, as we will prove the much stronger Path-Coloring Lemma below.) We thus have a set of edges that connect the 2-cycles of $\mathcal{C}$ and have weight at least $\alpha = (w(T_{\rm opt}) - bW)/2 + pW$. We then contract all the 2-cycles. Since they have zero weight in G', this contraction does not affect the total weight. We are now left with a collection of paths and cycles. We claim that the edges in these paths and cycles can be two-colored (i.e., partitioned into two color classes) such that each color class corresponds to a collection of vertex-disjoint directed paths. Furthermore, one of these color classes must contain edges whose total weight is at least $\alpha/2$.

Now we uncontract the two cycles. We claim that it is always possible to add one edge from each 2-cycle to the edges in our color class so that we still have collection of disjoint paths. To see why, assume it were not possible. Then some 2-cycle must have two incident edges that have the same color and are both directed either towards the cycle or away from the cycle. Thus, in the contracted graph, these two edges were both directed either toward some vertex or away from it, which means they could not have received the same color, a contradiction.

Adding the compatible edge from each 2-cycle, we obtain a set of directed paths whose weight is at least

$$\alpha/2 + cW \; = \; (w(T_{\rm opt}) - bW)/4 + pW/2 + cW \; .$$

Plugging in $p \le \frac{1}{3}(1-b-c)$ and $W \ge w(T_{\rm opt})$ then gives a bound of $(5/4)(c+p)w(T_{\rm opt})$. Balancing this bound with the $(1-c-p)w(T_{\rm opt})$ bound obtained by version 1 of the generic algorithm shows we can obtain a respectable 5/9-approximation algorithm for the Max TSP.

In order to obtain a stronger bound, we need to make two improvements to the simple version above. In the procedure above, we are placing zero weight on the 2-cycle edges and then guaranteeing that we will get the lighter edge in the 2-cycle. However, if the heavier edge in a 2-cycle has significantly more weight than the lighter edge, we may wish to allow an edge from the 2-cycle to be placed in the matching. To reflect this desire, we place a weight of $2(b_i - c_i)$ on the undirected edge corresponding to a 2-cycle C_i, where b_i is the weight of the heavier edge in C_i and c_i the weight of the lighter edge. We will also use a strong graph-coloring lemma that will allow us to include more than the lightest weight in each 3^+-cycle. In fact, it will allow us to include every edge except the lightest in each 3^+-cycle. We will then show that this collection of edges can be manipulated into a collection that satisfies the conditions of our lemma, while retaining the same amount of total weight. This approach will yield our improved result.

We begin by stating our coloring lemma.

Lemma 2 (Path-Coloring Lemma) Let G be a directed graph such that (1) each vertex has indegree at most 2, outdegree at most 2, and total degree at most 3, and (2) the graph does not contain any 2-cycles. Then the edges of G can be colored with 2 colors such that the edges in each color class form a set of directed paths.

Proof Omitted. ■

We call a coloring such as the one produced by the lemma a *path coloring*.

We will now analyze our improved algorithm, which is summarized in Figure 1. Let b_1W be the total weight of the edges in $T_{\rm opt}$ that are the heavier edge in a 2-cycle of $\mathcal{C}$, and let c_1W be the total weight of the corresponding light 2-cycle edges. Similarly, let c_2W be the total weight of the edges in $T_{\rm opt}$ that are the lighter edge in a 2-cycle of $\mathcal{C}$, and let b_2W be the total weight of the corresponding heavy 2-cycle edges. Then the weight of the optimal tour edges that are not in a 2-cycle in $\mathcal{C}$ is $w(T_{\rm opt}) - b_1W - c_2W$. Thus, we can add back in the modified weight of the 2-cycle edges to find that the total weight of the optimal tour in G' is at least

$$w(T_{\rm opt}) - b_1W - c_2W + 2(b_1 - c_1)W + 2(b_2 - c_2)W \; .$$

Since each 2-cycle C_i has $c_i \le b_i \le 2c_i$, we must have

$$(b_1 - 2c_1 + 2b_2 - 3c_2) \; \ge \; (b_1 + b_2) - 2(c_1 + c_2) \; \ge \; b - 2c \; ,$$

which implies the weight of the optimal tour in G' is at least $w(T_{\rm opt}) + (b-2c)W$. Thus, the matching found in Step 2b will have weight at least $\frac{w(T_{\rm opt}) + (b-2c)W}{2}$.

In Step 2c, we add back in weight $(1-p-b-c)W$ from the 3^+-cycles, so at the beginning of Step 2d, we have a graph with total weight at least

$$\frac{w(T_{\rm opt}) + (b-2c)W}{2} + (1-p-b-c)W \; . \qquad (1)$$

To compute the weight remaining at the end of Step 2d, let A be the set of 2-cycles that have corresponding

Version 2, Real Variant

Step 2a. Construct an undirected graph G' from G and $\mathcal{C}$ as follows:

- Reclassify each 2-cycle with $b_i > 2c_i$ as a 3^+-cycle.
- Replace each 2-cycle $\{(u,v),(v,u)\}$ of $\mathcal{C}$ with an undirected edge (u,v) of weight $2(b_i - c_i)$.
- For all other pairs of vertices u and v, create an undirected edge (u,v) with weight $\max\{w(u,v), w(v,u)\}$.

Step 2b. Find a maximum-weight matching M in G'.

Step 2c. To the directed counterparts of the undirected edges of M, add all but the lightest edge from each 3^+-cycle.

Step 2d. Contract the 2-cycles of $\mathcal{C}$.

Step 2e. Two-color the remaining graph using Lemmas 2 and 3 to obtain two sets of directed paths. Let P consist of those edges in the color class with more weight, together with one edge from each 2-cycle.

Figure 1: Version 2, Real Variant

edges in the matching. Furthermore, let $\hat{b}W$ be the sum of the weights of the heavier edges in A, and let $\hat{c}W$ be the sum of the weights of the lighter edges in A. Simplifying (1) and subtracting the $2(\hat{b} - \hat{c})W$ of weight that is contracted in Step 2d, we get a collection of edges of weight at least

$$\frac{w(T_{\mathrm{opt}})}{2} + \left(1 - 2c - \frac{b}{2} - p - 2(\hat{b} - \hat{c})\right) W \qquad (2)$$

at the end of Step 2d.

Now consider the collection of edges with which we begin Step 2e. They consist of a set of paths that came from the 3^+-cycles and a set of edges from the matching. This graph does not satisfy the conditions of Lemma 2, but it can still be path colored with two colors, as the following lemma shows.

Lemma 3 Let E be the set of edges after Step 2d. E can be path-colored with two colors (as suggested in Figure 2).

Proof By the construction, each vertex has at most one incoming cycle edge, one outgoing cycle edge, and one additional matching edge, which means it satisfies the degree requirements of Lemma 2. Furthermore, when you contract the 2-cycles of $\mathcal{C}$, two vertices of degree at most 3 with 2 common edges are contracted to form a vertex of degree at most 2. Thus, we have satisfied all the conditions of Lemma 2 except that there may be 2-cycles in our graph. However, the only 2-cycles possible are those formed by a cycle edge and a matching edge. To distinguish these from the 2-cycles in $\mathcal{C}$, we will call these *induced 2-cycles*.

Now consider such a 2-cycle $\{(u,v),(v,u)\}$. By the degree-three constraint and the fact that one of the

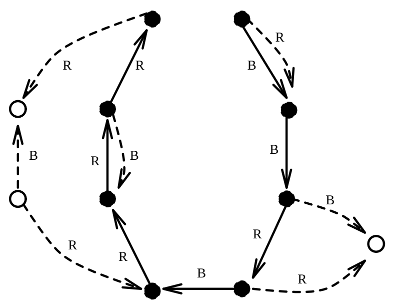

Figure 2: The set of edges after Step 2d can be path-colored with two colors. The hollow nodes represent contracted two cycles. The solid arrows represent cycle-cover edges and the dotted arrows represent matching edges.

edges between u and v is a matching edge, we know that there is at most one other edge incident to v (call it (v,x)) and at most one other edge incident to u (call it (u,y)) and that neither of these edges is a matching edge. (We will assume that there are edges incident to both u and v; the case where one or both edges are missing is similar.) To eliminate the 2-cycle, we contract the path (x,v,u,y) down to one edge (x,y). It is straightforward to verify that we now have a graph with the same properties and one fewer 2-cycle. We iterate this process until there are no 2 cycles remaining. After we color the contracted graph, it is simple to color the original graph. If (x,y) is colored with particular color (call it red), then color (x,v), (v,u), and (u,y) red and color (u,v) with the other color (call it blue). Since (u,v) cannot be incident to another blue edge, it is safe to color it this way. ■

Since the weight of the edges remaining after contracting 2-cycles in Step 2d is at least the quantity in (2), we can obtain a collection of directed paths of weight at least

$$\frac{1}{2}\left(\frac{w(T_{\text{opt}})}{2}+\left(1-2c-\frac{b}{2}-p-2(\hat{b}-\hat{c})\right)W\right) \quad (3)$$

using the preceding lemma. All that remains to analyze is the addition of 2-cycle edges at the end of Step 2e. This addition increases the quantity in (3) by at least $(\hat{b}+(c-\hat{c}))W$, since for each 2-cycle in $\mathcal{C}$, we can add the 2-cycle's heavier edge if the corresponding undirected edge was included in M and an edge whose weight is at least that of the lighter edge if not. Thus, we have achieved a collection of paths of weight at least

$$\begin{aligned}
&\frac{1}{2}\left(\frac{w(T_{\text{opt}})}{2}+\left(1-2c-\frac{b}{2}-p-2(\hat{b}-\hat{c})\right)W\right)\\
&\quad+(\hat{b}+(c-\hat{c}))W\\
&\quad\geq \left(\frac{3}{4}-\frac{b}{4}-\frac{p}{2}\right)w(T_{\text{opt}})\\
&\quad\geq \left(\frac{7}{12}-\frac{1}{12}(b-2c)\right)w(T_{\text{opt}})\ .
\end{aligned}$$

The last line is obtained by plugging in the bound $p \leq \frac{1}{3}(1-b-c)$ (which still holds, we claim, after reclassifying 2-cycles with $b_i > 2c_i$ as 3^+-cycles). Thus, we have the following lemma.

Lemma 4 Version 2 of the generic algorithm returns a $(\frac{7}{12}-\frac{1}{12}(b-2c))$-approximation to the Max TSP.

Note that the above bound is in terms of the values of b and c after reclassifying 2-cycles with $b_i > 2c_i$ as 3^+-cycles; thus, we need to be careful in comparing this bound with the bounds for versions 1 and 3, which are in terms of the original values of b and c. Fortunately, the aforementioned reclassification can only reduce the value of $b-2c$, so the bound of Lemma 4 must also hold for the original values of b and c.

2.3 Version 3

We will begin our description of our third version of the generic algorithm by first presenting a weaker variant that illustrates several of the techniques used to obtain our third version's bound. The intuition behind this weaker variant is that we are (again) attempting to pick up some combination of the weight in the cycle cover and the weight that was missed by the cycle cover. To this end, we first form a new graph $G' = (V', E')$ by contracting the cycles of $\mathcal{C}$. We then find a matching in G' whose weight is maximized with respect to a new weight metric designed to reflect the "compatibility" of edges outside the cycle cover with the cycle cover's edges. Given a cycle C and an edge e with exactly one endpoint incident to C, we define $\text{cw}(e, C)$, the *compatible weight of* C *with respect to* e, to be $w(C) - w(f_{e,C})$, where $f_{e,C}$ is the one edge in C that is incompatible with e, i.e., it shares a head or tail with e. We then define the weight $w'(e)$ of an edge $e \in E'$ to be $w'(e) = w(e) + \text{cw}(e, C_{\text{head}(e)}) + \text{cw}(e, C_{\text{tail}(e)})$, where $C_{\text{head}(e)}$ and $C_{\text{tail}(e)}$ are the cycles corresponding to the endpoints of e. (See Figure 3.) Note that a matching M in G' corresponds in a natural way to a set P of paths in G such that $w(P) = w'(M)$.

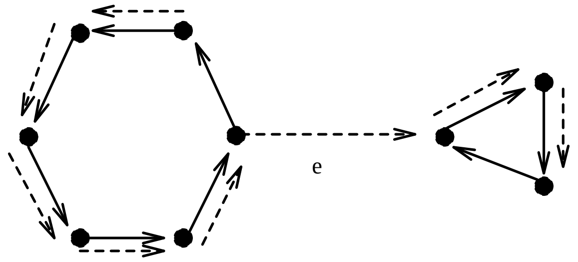

Figure 3: The new weight $w'(e)$ of edge e is the sum of the original weights of the dotted edges.

VERSION 3, SIMPLE VARIANT

Step 2a. Construct an auxiliary directed (multi-) graph $G' = (V', E')$ by contracting each cycle $C \in \mathcal{C}$ to a single vertex. Define edge weights for G' as above.

Step 2b. Find a maximum-weight matching M in G'. Let P be the set of paths in G corresponding to the edges of M.

To analyze the weight of P, let the maximum number of vertices in any cycle of $\mathcal{C}$ be denoted by γ. Furthermore, let T_{opt} be the optimal Max TSP tour, and let T_{ext} denote the subset of edges from T_{opt} with endpoints in distinct cycles of $\mathcal{C}$. We begin by assuming that $T_{\text{opt}} = T_{\text{ext}}$, i.e., all the tour edges go between cycles in the cycle cover. In this case, we can bound $w'(T_{\text{opt}})$ by observing that it consists of $w(T_{\text{opt}})$ plus some weight from the cycles. More precisely,

$$
\begin{aligned}
w'(T_{\mathrm{opt}}) & \\
&= \sum_{e \in T_{\mathrm{opt}}} w'(e) \\
&= \sum_{e \in T_{\mathrm{opt}}} \left(w(e) + \mathrm{cw}(e, C_{\mathrm{head}(e)}) + \mathrm{cw}(e, C_{\mathrm{tail}(e)}) \right) \\
&= w(T_{\mathrm{opt}}) \\
&\quad + \sum_{C \in \mathcal{C}} \left(\sum_{\substack{e \in T_{\mathrm{opt}} \\ \text{s.t. head}(e) \in C \\ \text{or tail}(e) \in C}} w(C) - w(f_{e,C}) \right) ,
\end{aligned}
$$

where the last inequality follows by substituting the definition of compatible weight and regrouping. Since each cycle C has $2|C|$ incident tour edges (where $|C|$ denotes the number of vertices in C) and since each edge of C will be incompatible with exactly two tour edges, we conclude that

$$
w'(T_{\mathrm{opt}}) \; = \; w(T_{\mathrm{opt}}) + \sum_{C \in \mathcal{C}} (2|C| - 2) w(C) \; .
$$

Since no vertex of G' has more than 2γ edges from T_{opt} incident to it, Vizing's theorem [16] tells us that it is possible to $(2\gamma + 1)$-color the edges of T_{opt} in G'. Thus, these edges can be partitioned into $(2\gamma + 1)$ matchings with an average weight of at least $\frac{1}{2\gamma+1} w'(T_{\mathrm{opt}})$. In Step 2b, we are finding a maximum-weight matching in a supergraph of T_{opt}, so the maximum-weight matching must have weight at least $\frac{1}{2\gamma+1} w'(T_{\mathrm{opt}})$.

In fact, we can make a significantly stronger claim. Consider a particular color class, say red. It is quite possible that some of the vertices will not have *any* incident red edges. Let v_C be one such vertex, and consider the set of paths P in G corresponding to the red edges. Since v is not incident to any red edges in G', and since v_C represents a cycle C in G, we know that none of the vertices in that cycle have any incident edges in P. Thus, we can augment P by adding, say, all but one of C's edges. In particular, if we let $w'(C)$ denote the weight of all the edges in C except the lightest, then we can use a matching M, to find a set of paths P of weight $\sum_{e \in M} w'(e) + \sum_{v_C \notin M} w'(C)$.

Now observe that the vertex v_C of G' corresponding to cycle C of G is unmatched in exactly $2\gamma + 1 - |C|$ of the $2\gamma + 1$ matchings obtained from T_{opt}. Thus, summing over all $2\gamma + 1$ matchings, we see that for each C, we can add

$$
(2\gamma + 1 - |C|) w'(C) \; \geq \; (2\gamma + 1 - |C|) \frac{|C| - 1}{|C|} w(C)
$$

to the total weight of the matchings, which implies

$$
\begin{aligned}
w'(M) & \\
&\geq \frac{1}{2\gamma + 1} \Bigg(w(T_{\mathrm{opt}}) \qquad (4) \\
&\qquad + \sum_{C \in \mathcal{C}} (2|C| - 2) w(C) \\
&\qquad + \sum_{C \in \mathcal{C}} (2\gamma + 1 - 2|C|) \frac{|C| - 1}{|C|} w(C) \Bigg) \\
&\geq \frac{1}{2\gamma + 1} \Bigg(w(T_{\mathrm{opt}}) \qquad (5) \\
&\qquad + \sum_{C \in \mathcal{C}} (2\gamma + 1) \frac{|C| - 1}{|C|} w(C) \Bigg) \; .
\end{aligned}
$$

Although we are ultimately interested in the weight of matchings, as the previous analysis has suggested, it is convenient to focus on how much weight each individual cycle contributes to the union of the $2\gamma + 1$ matchings. We define the total weight that a particular cycle C contributes as a result of colors that *are* assigned to incident tour edges (the term inside the first summation of (4)) to be C's *colored contribution* and the total weight that C contributes as a result of colors that *are not* assigned to incident tour edges (the term inside the second summation of (4)) to be C's *uncolored contribution*.

We note that the notion of an uncolored contribution is only for the purpose of analysis. It is easy to see that there is never any advantage to leaving two cycles unmatched, as we could just take the two unmatched pieces and join them by an edge. Since that edge must have non-negative (actual) weight, it is always at least as good to match two vertices as it is to leave them unmatched. Thus, our algorithm will return a complete matching (assuming an even number of vertices in G') and hence does not have to consider the case of unmatched vertices.

We now remove the assumption that $T_{\mathrm{opt}} = T_{\mathrm{ext}}$ and consider how (5) changes. Since the only edges of T_{opt} that appear in G' are those of T_{ext}, we must replace $w(T_{\mathrm{opt}})$ with $w(T_{\mathrm{ext}})$. Furthermore, the vertex of G' corresponding to a cycle of degree $|C|$ in the original graph may no longer have $2|C|$ T_{opt} edges incident to it, which can make its colored contribution decrease. However, this decrease is always offset by an increase in the uncolored contribution. Specifically, for each edge of T_{opt} that is internal to C, the colored contribution decreases by $2|C| - w(e_1) - w(e_2)$ where e_1 and e_2 are edges of C, as C is incident to two fewer color classes. However, it is now uncolored two more times, so the uncolored contribution increases by $2w'(C) = 2|C| - 2w(e_{\mathrm{lightest}})$, where e_{lightest} is the minimum-weight edge of C. Since every edge $e_i \in C$ has $w(e_i) \geq w(e_{\mathrm{lightest}})$, the net gain is non-negative. Thus, we can conclude that

$$\begin{aligned} & w'(M) \\ \geq \quad & \frac{1}{2\gamma+1}\left(w(T_{\mathrm{ext}}) \right. \qquad (6) \\ & \left. + \sum_{C\in\mathcal{C}} (2\gamma+1)\frac{|C|-1}{|C|} w(C) \right) . \end{aligned}$$

To complete our analysis of the simple version of the generic algorithm's third variant, let $T_{\mathrm{int}}(C)$ be the set of T_{opt} edges internal to C, i.e., the set of edges in $T_{\mathrm{opt}} - T_{\mathrm{ext}}$ that have both endpoints in C. We observe that $w(T_{\mathrm{int}}(C)) \leq w(C)$, as otherwise we could obtain a cycle cover with strictly more weight than $\mathcal{C}$ by replacing C with a new cycle obtained by patching together the edges of $T_{\mathrm{int}}(C)$. Thus, $w(T_{\mathrm{ext}}) \geq w(T_{\mathrm{opt}}) - \sum_{C\in\mathcal{C}} w(C)$, which means we can rewrite (6) as

$$\begin{aligned} & w'(M) \\ \geq \quad & \frac{1}{2\gamma+1}\left(w(T_{\mathrm{opt}}) \right. \qquad (7) \\ & \left. + \sum_{C\in\mathcal{C}} \left((2\gamma+1)\frac{|C|-1}{|C|} - 1 \right) w(C) \right) . \end{aligned}$$

If we plug a lower bound of $|C| \geq 2$ into (7), we find only that $w'(M) \geq \frac{1}{2}w(T_{\mathrm{opt}})$, which is no improvement over the bound of version 1. However, we can obtain a better bound using a slightly modified algorithm and a more careful analysis of the second term in (7). For each cycle C, this term depends not only on C's colored and uncolored contributions, but also on the *internal weight* that we must subtract off. (In the preceding analysis, we bounded this latter quantity using the inequality $w(T_{\mathrm{int}}(C)) \leq w(C)$.) If we add C's colored contribution to its uncolored contribution and subtract off its internal weight, we obtain what we call C's *net contribution*. The key to better analysis will be to give tighter bounds on the net contribution of each cycle C based on the size of C.

Another important feature of our modified algorithm is that, prior to constructing G', we break the larger cycles of $\mathcal{C}$ into smaller cycles of constant size. By reducing the size of the largest cycle, we will be able to color the edges of T_{ext} with fewer colors and thus replace the fraction $\frac{1}{2\gamma+1}$ in (7) with a larger fraction that is independent of the cycle cover. However, we do lose some weight in breaking up the large cycles, weight that we will have to account for below.

A third modification that we make involves using a somewhat different calculation of the compatible weight of an edge and also the uncolored weight of a cycle. Specifically, we redefine $\mathrm{cw}(e, C)$ to be $w(H_{e,C})$, where $H_{e,C}$ is a maximum-weight Hamiltonian path on the vertices of C that is compatible with e. Note that under this new metric, it is still possible to obtain a set P of paths in G from a matching M in G' such that $w(P) = w'(M)$, and that $w(H_{e,C})$ is always at least $w(C) - w(f_{e,C})$. We similarly redefine the uncolored weight of a cycle to be $w(H_C)$, where H_C is a maximum-weight Hamiltonian path on the vertices of the cycle (that is not restricted to be compatible with any edge). Note that all of the above Hamiltonian paths can be computed in constant time per path once all cycles are of constant size.

<u>VERSION 3, REAL VARIANT</u>

Step 2a. Let x be a constant to be determined later. Break each cycle C such that $|C| > x$ into $\lceil |C|/x \rceil$ pieces, all but at most one of size x. We denote the new set of cycles by $\mathcal{C}'$.

Step 2b. Construct an auxiliary directed graph $G' = (V', E')$ by contracting each cycle $C \in \mathcal{C}'$ to a single vertex. Define edge weights w' for G' as described above.

Step 2c. Find a maximum-weight matching M in G'. Let P be the set of paths in G corresponding to the edges of M.

We now show that splitting a cycle into pieces does not cause us to throw away too much weight.

Lemma 5 In Step 2a, each cycle C can be broken into pieces P_i such that

$$\begin{aligned} \sum_{P_i \in C} w(P_i) \quad & \geq \quad \frac{|C| - \left\lceil \frac{|C|}{x} \right\rceil}{|C|} w(C) \\ & \geq \quad \frac{x-1}{x+1} w(C) . \end{aligned}$$

Sketch of Proof If we must delete k edges out of ℓ in C, only weight $(k/\ell)w(C)$ need be lost, since the weight of the minimum-weight set of k edges in any fixed configuration can be no worse than the average weight of such a set. The worst case, in terms of the fraction of weight lost, comes when we lose 2 edges from a cycle with $x+1$ edges. ■

Note that for the purposes of our analysis, there is an important difference between an unsplit cycle (i.e., one whose original size was at most x) and a piece P_i from a split cycle. The piece P_i is really just a path (rather than a cycle), which means we can pick up all of its weight when it is unmatched, i.e.,

$$w'(P_i) \;\geq\; w(P_i)\;. \tag{8}$$

The corresponding inequality is *not* necessarily true for unsplit cycles.

We can now bound the net contribution of cycles of size greater than x by combining the previous lemma and discussion with the techniques used to analyze the simple variant.

Lemma 6 Let $C \in \mathcal{C}$ be such that $|C| > x$. Then the sum of the net contributions of the pieces that comprise C is no less than

$$(2x-2)\frac{x-1}{x+1}w(C)\;.$$

Proof Since each edge's weight is at least as large as it was for the simple variant of our algorithm, the colored and uncolored contributions of a piece P_i sum to at least

$$(2|P_i|-2)w(P_i)+(2x+1-2|P_i|)w'(P_i)\;.$$

Furthermore, since $w(H_{P_i})$ is an upper bound on the weight of those edges of T_{opt} that are internal to a piece P_i, so is $w'(P_i)$. Thus, the net contribution of all the pieces of C is at least

$$\sum_{P_i\in C}(2|P_i|-2)w(P_i)+(2x+1-2|P_i|)w'(P_i)-w'(P_i)\;.$$

Using (8) and combining terms yields the lemma. ■

The other key will be a more careful handling of the 2-cycles. For a particular 2-cycle C_i, let b_i be the weight of the heavier edge, and let c_i be the weight of the lighter edge. Then there are three cases for how the maximum-weight tour can traverse the cycle. These cases are analyzed in the following lemma.

Lemma 7 A 2-cycle C_i whose heavier edge has weight b_i and whose lighter edge has weight c_i makes a net contribution of at least $(2x-1)b_i + c_i$ to (7).

Proof Omitted. ■

We can now improve the bound of (7). Specifically, we use Lemma 6 to bound the net contribution of the large cycles and Lemma 7 to bound the net contribution of the 2-cycles. For cycles of size 3 to x, we use the original bound together with $|C| > 3$. Combining all these bounds yields

$$\begin{aligned}
&w'(M)\\
&\geq \frac{1}{2x+1}\Bigg(w(T_{\mathrm{opt}})\\
&\qquad+\sum_{\substack{C_i\in\mathcal{C}\\|C|=2}}((2x-1)b_i+c_i)\\
&\qquad+\sum_{\substack{C\in\mathcal{C}\\3\leq|C|\leq x}}\left((2x+1)\frac{2}{3}-1\right)w(C)\\
&\qquad+\sum_{\substack{C\in\mathcal{C}\\|C|>x}}(2x-2)\frac{x-1}{x+1}w(C)\Bigg)\\
&\geq \frac{1}{2x+1}\Bigg(w(T_{\mathrm{opt}})+(2x-1)bW+cW\\
&\quad+\min\left\{\frac{4}{3}x-\frac{1}{3},(2x-2)\frac{x-1}{x+1}\right\}(1-b-c)W\Bigg)\;.
\end{aligned}$$

Setting $x = 7$ and performing some algebraic manipulations yields the following lemma.

Lemma 8 Version 3 of the generic algorithm returns a $(\frac{2}{3}+\frac{4}{15}(b-2c))$-approximation to the Max TSP. ■

Combining Lemmas 1, 4, and 8, we obtain the following theorem.

Theorem 1 If we run versions 1, 2, and 3 of the generic algorithm and take the best tour, we obtain a $\frac{38}{63}$-approximation to the Max TSP. ■

3 Approximating the Longest Tour in an Undirected Graph

Techniques similar to those used in version 3 of the previous section's generic algorithm can also be used to obtain improved approximation results for the undirected Max TSP. The best previous algorithm for the problem, due to Fisher, Nemhauser, and Wolsey

[6], begins by finding a maximum-weight 2-matching, which is simply a cycle cover containing only cycles of size three or more. Given an n-vertex undirected graph, such a cycle cover can be found in $O(n^3)$ time, as the maximum-weight 2-matching problem is easily reduced to the standard maximum-weight matching problem for an $O(n)$-vertex undirected graph. Fisher et al.'s algorithm then deletes the minimum-weight edge from each cycle and connects the resulting paths to obtain a 2/3-approximation to the graph's maximum tour. In this section, we will improve this approximation bound to 5/7.

Like the algorithms of the previous section, our improved approximation algorithm for the undirected Max TSP starts by computing a maximum-weight cycle cover $\mathcal{C}$ for the input graph G, though here we use a 2-matching to insure $\mathcal{C}$ contains no 2-cycles. We then attempt to pick up some weight outside of $\mathcal{C}$ by finding a maximum-weight matching in a related graph G'. As in version 3 of the previous section's generic algorithm, this new graph is obtained by contracting some of the original graph's vertices. Rather than contracting all of the vertices in each cycle, however, we will contract only some of them. Specifically, for each cycle C, we will chose (arbitrarily) three adjacent vertices, which we will call the *representatives* of C. It is these 3-sets of vertices that we will contract before applying the matching algorithm.

We also use a new edge weight metric to measure the worth of edges in G'. To define this metric, we first orient the edges of each cycle to obtain a directed cycle (which of the two possibilities we choose is irrelevant). Given an edge e with endpoint x in cycle C, we then define $f_{x,C}$ to be the edge of C whose head is incident to x. If x is a representative of C, we also define $H_{x,C}$ to be a maximum-weight Hamiltonian path on the representatives of C that has x as an endpoint. Next, we define $\mathrm{cw}(x, C)$, the *compatible weight of C with respect to x*. (This quantity is similar but not identical to quantity $\mathrm{cw}(e, C)$ defined in the last section.) If x is a representative of C, then $\mathrm{cw}(x, C) = \max\{w(C) - w(f_{x,C}), H_{x,C}\}$. On the other hand, if x is not a representative of C, then $\mathrm{cw}(x, C) = -w(f_{e,x})$. Finally, given an edge e with endpoints x and y on cycles C_x and C_y, respectively, we define

$$w'(e) \;=\; w(e) + \mathrm{cw}(x, C_x) + \mathrm{cw}(y, C_y) \;.$$

Note that C_x and C_y may be the same cycle, but in this case, at most one of x and y can be a representative of the cycle.

To see why these new edge weights are useful, consider the following lemma.

Lemma 9 Corresponding to any matching M in G' is a set P of vertex-disjoint paths in G with total weight at least $w'(M)$.

Proof Each edge $e \in M$ has a weight $w'(e)$ that is computed by adding the weights of a collection of edges and subtracting the weights of another collection. Letting $e_1, \ldots, e_m$ of G' in some order, we define an m-dimensional vector $V'(e)$ whose ith component $V'_i(e)$ is $+1$ if $w(e_i)$ is added to $w'(e)$, -1 if $w(e_i)$ is subtracted from $w'(e)$, and 0 otherwise. Just as $w'(M)$ is the sum of $w'(e)$ over all $e \in M$, we define $V'(M)$ to be the vector sum of $V'(e)$ over all $e \in M$. The quantity $V'_i(M)$ then denotes the (net) number of times e_i contributes to the sum $w'(M)$. We define P to be the multiset of edges obtained by including e_i in the multiset $\max\{V'_i(M), 0\}$ times. This definition ensures that

$$w(P) \;\geq\; \sum_{i=1}^{m} V'_i(M) w(e_i) \;=\; w'(M) \;.$$

It remains be shown that P is indeed a set of vertex-disjoint paths. To do so, it will be sufficient to establish the following three properties of P:

1. P contains no edge more than once,
2. the subgraph of G corresponding to P has no vertex of degree three or more, and
3. the subgraph of G corresponding to P is acyclic.

In the following proof, we ignore the possibility that there exists a representative x of a cycle C such that $\mathrm{cw}(x, C) = H_{x,C}$. It is easy to verify that all of the properties still hold in the presence of such x and C.

We first prove Property 1. We will show that $V'_i(M) \leq 1$ for all i. Consider a particular edge e_i. By the definition of our edge weight metric, $V'_i(f) > 0$ implies either (1) $f = e_i$ or (2) e_i is contained in some cycle C and f is incident to a representative of C. Since M contains at most one f incident to a representative of C and since $V'_i(f)$ is at most one, we need only show that the existence of such an $f \in M$ implies $V'_i(e_i) = 0$ to complete our proof that P satisfies Property 1. To see why this claim follows, let x and y denote the endpoints of e_i, and suppose that C orients e_i from x to y. The weight $w'(e_i)$ assigned to e_i is $w(e_i) + \mathrm{cw}(x, C) + \mathrm{cw}(y, C)$. Since the head of e_i is incident to y, the quantity $\mathrm{cw}(y, C)$ contains a $-w(f_{y,C}) = -w(e_i)$ term. Furthermore, no other term contains a term involving e_i, which implies $V'_i(e_i) = 0$.

We now prove Property 2. Since P is a subgraph of the union of a matching and a cycle cover, we know that each vertex has at most two incident cycle-cover edges and at most one incident matching edge (and no other incident edges). Thus, the only way to have a degree-3 vertex is to have exactly two incident cycle-cover edges and one incident matching edge. Now consider a vertex v with two incident cycle-cover edges. From the definition of $\mathrm{cw}(v, C)$, we see that adding a matching edge incident to v effectively removes one of the two cycle-cover edges. Thus, v's degree will never exceed two.

We prove Property 3 by contradiction. Assume that P does contain a cycle C'. First observe that C' cannot have two consecutive edges from M, as the existence of such edges would contradict the definition of a matching. Also observe that $P - M$ is acyclic, as it is a subgraph of a collection of disjoint "near-cycles," where a near-cycle is a cycle with one edge deleted. Thus, C' must contain at least one edge from the matching. Call this edge a, and consider either of the two cycle-cover edges that must be adjacent to a in C'. This edge, which we will call z_1, must have its tail incident to a, as otherwise it would be "incompatible" with a. Continuing to traverse C', we will encounter zero or more additional cycle edges $z_2, \ldots, z_k$ and then another matching edge b (which may be identical to a). By the definition of z_k, its head must be incident to an endpoint of b. Thus, z_k is "incompatible" with b and could not be part of C', a contradiction. ■

We can now summarize our algorithm.

<u>Undirected MAX TSP Algorithm</u>

Step 1. Compute a maximum-weight 2-matching $\mathcal{C}$ for G.

Step 2. From each cycle of $\mathcal{C}$, select three adjacent vertices to be the cycle's representatives. Construct an auxiliary graph $G' = (V', E')$ by contracting each cycle's representatives to a single vertex. Define edge weights for G' as above (which will require solving a maximum-weight Hamiltonian-path problem for each cycle C).

Step 3. Compute a maximum-weight matching M for G'.

Step 4. Construct a tour for G by patching together the paths corresponding to M.

The analysis of our algorithm is similar to the analysis for version 3 of the previous section's generic algorithm. We again start by assuming that all of the edges in the optimal tour T_{opt} appear in G', i.e., none of them connect representatives of the same cycle. In this case, it is not hard to see that the *colored contribution* of a cycle $C \in \mathcal{C}$ is $4w(C)$, since the weight of each cycle C is added six times (twice for each representative) and the weight of each edge of C is subtracted twice, as it is incompatible with exactly two T_{opt} edges. As in the directed case, we can also talk about the *uncolored weight* of a vertex v in G'. The vertex of G' corresponding to the representatives of a cycle C in G will be uncolored once, so in this case, we can add the maximum of the weight of the $|C| - 1$ largest edges in C and the weight of a maximum-weight Hamiltonian path on the representatives of C. (Note that when the other vertices of G' are unmatched, we pick up no additional weight.) Letting H_C denote a maximum-weight Hamiltonian path on the representatives of C, we obtain

$$\begin{aligned} w'(M) & \\ \geq \ & \frac{1}{7}\Bigg(w(T_{\mathrm{opt}}) \qquad\qquad (9) \\ & + \sum_{C \in \mathcal{C}} \left(4w(C) + \max\left\{\frac{|C|-1}{|C|}w(C), w(H_C)\right\}\right)\Bigg) \ . \end{aligned}$$

Now consider what happens when there are edges of T_{opt} between representatives of the same cycle. Let T_{ext} denote the subset of T_{opt} edges that do appear in G', and for each cycle C of G, let $T_{\mathrm{int}}(C)$ denote the subset of T_{opt} edges both of whose endpoints are representatives of C. The first term in (9) is now only $w(T_{\mathrm{ext}})$, so we will need to transfer some weight from the second term to the first in order to bring the first back up to $w(T_{\mathrm{opt}})$ (much as we did in the last section). For the cycles that no longer have six distinct representative edges, the colored contribution decreases; however, this decrease is no more than the corresponding increase in the uncolored contribution, since the uncolored contribution is at least $\frac{|C|-1}{|C|}w(C)$ per matching. We will use the fact that the uncolored contribution is also at least H_C per matching to offset the loss for T_{opt} edges that are internal to the representatives of a cycle. Thus, we can obtain

$$\begin{aligned} w'(M) & \\ \geq \ & \frac{1}{7}\left(w(T_{\mathrm{ext}}) + \sum_{C \in \mathcal{C}} \left(4w(C) + w(T_{\mathrm{int}}(C))\right)\right) \\ \geq \ & \frac{1}{7}\left(w(T_{\mathrm{opt}}) + 4w(\mathcal{C})\right) \\ \geq \ & \frac{5}{7} w(T_{\mathrm{opt}}) \ . \end{aligned}$$

Theorem 2 It is possible to find a tour of an undirected graph whose total weight is at least 5/7 of the weight of a maximum-weight tour for the graph. ■

Hartvigsen [9] has shown that in unweighted undirected graphs, it is possible to find a triangle-free cycle cover if one exists. If the analogous result held for weighted undirected graphs, our approach would yield a $\frac{7}{9}$-approximation for the undirected Max TSP.

4 Concluding Remarks

If we could improve our bound for the directed max TSP to $\frac{5}{8}$, which would represent an improvement of about $\frac{2}{100}$, we would have a $2\frac{3}{4}$-approximation for the shortest-superstring problem. Such a result would match the best bound previously known [1]. We believe that such a bound is attainable using methods similar to those presented in this paper.

Acknowledgements

We would like to thank Chris Armen, Perry Fizzano, and David Shmoys for helpful discussions.

References

[1] C. Armen and C. Stein. A $2\frac{3}{4}$-approximation algorithm for the shortest superstring problem, May 1994. Unpublished manuscript.

[2] A. I. Barvinok. Two algorithmic results for the traveling salesman problem, April 1994. Unpublished manuscript.

[3] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, LA, May 6–8, 1991*, pages 328–336. ACM Press, New York, 1991.

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

[5] A. Czumaj, L. Gąsieniec, M. Piotrów, and W. Rytter. Parallel and sequential approximation of shortest superstrings. In E. M. Schmidt and S. Skyum, editors, *SWAT 94: Proceedings of the 4th Scandinavian Workshop on Algorithm Theory, Aarhus, Denmark, July 6–8, 1994*, volume 824 of *Lecture Notes in Computer Science*, pages 95–106. Springer-Verlag, Berlin, 1994.

[6] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for finding a maximum weight hamiltonian circuit. *Operations Research*, 27(4):799–809, 1979.

[7] A. M. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[8] J. Gallant, D. Maier, and J. Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50–58, 1980.

[9] D. B. Hartvigsen. *Extensions of Matching Theory.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1984.

[10] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[11] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.

[12] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[13] J. Tarhio and E. Ukkonen. A greedy approximation algorithm for constructing shortest common superstrings. *Theoretical Computer Science*, 57(1):131–145, 1988.

[14] S.-H. Teng and F. Yao. Approximating shortest superstrings. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, Palo Alto, CA, November 3–5, 1993*, pages 158–165. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[15] J. Turner. Approximation algorithms for the shortest common superstring problem. *Information and Computation*, 83(1):1–20, 1989.

[16] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskretnyi Analiz*, 3:25–30, 1964. In Russian.