Operations Research Letters 55 (2024) 107133

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Circulant TSP special cases: Easily-solvable cases and improved approximations

Austin Beal, Yacine Bouabida, Samuel C. Gutekunst *, Asta Rustad

ARTICLE INFO

Article history:
Received 24 August 2023
Received in revised form 18 April 2024
Accepted 23 May 2024
Available online 31 May 2024

Keywords:
Traveling Salesman Problem
Circulant
Complexity

ABSTRACT

Circulant TSP is an intriguing special case of the Traveling Salesman Problem, whose complexity remains an often-cited open problem. In this note, we present three results: We show that circulant TSP can be efficiently solved whenever the input number of vertices is a prime-squared; we show that the $\{1,2\}$-TSP can be easily and efficiently solved when specialized to circulant instances; and we present a substantially-improved approximation factor for finding a minimum-cost Eulerian connected sub-(multi)graph on two-stripe circulant instances.

1. Introduction and circulant TSP

The Symmetric Traveling Salesman Problem (TSP) is a fundamental problem in combinatorial optimization and a canonical NP-hard problem. An input consists of a set of n vertices $[n] := \{1, 2, \ldots, n\}$ and edge costs $c_{ij} = c_{ji}$ (for $1 \leq i, j \leq n$), indicating the costs of traveling between vertices i and j. The TSP is then to find a minimum-cost Hamiltonian cycle, visiting each vertex exactly once.

With just this set-up, the TSP is well known to be NP-hard. An algorithm that could approximate TSP solutions in polynomial time to within any factor $\alpha > 1$ would imply P=NP (see, e.g., Theorem 2.9 in Williamson and Shmoys [31]). Thus it is common to consider special cases that restrict the edge costs. For instance, requiring costs to be *metric* (so that $c_{ij} + c_{jk} \geq c_{ik}$ for all $i, j, k \in [n]$), to correspond to distances in an underlying graph on $[n]$, to correspond to Euclidean distances, or to be restricted as $c_{ij} \in \{1, 2\}$ for all i, j (the *(1, 2)-TSP*). See, e.g., [2,3,8,17,18,22–26,28,29] among many others.

One special case that is particularly intriguing, but where relatively little is known, is *circulant TSP*. Circulant TSP instances are those whose edge costs can be described by a *circulant matrix*, which imposes substantial symmetry: the cost of edge $\{i, j\}$ can only depend on $(i - j) \bmod n$. Our implicit assumption that the edge costs are also symmetric means that the cost of an edge c_{ij} is a function of $\min\{(i - j) \bmod n, (j - i) \bmod n\}$, which we interpret as the **length** of the edge $\{i, j\}$. For instance, edges $\{1, 2\}$, $\{3, 2\}$, and $\{n, 1\}$ all have the same length 1; we will denote the cost of any such edge as c_1. We can thus describe (symmetric) circulant TSP instances with a symmetric, circulant cost matrix with $\lfloor \frac{n}{2} \rfloor$ parameters $c_1, c_2, \ldots, c_{\lfloor \frac{n}{2} \rfloor}$:

$$C := (c_{ij})_{i,j=1}^{n} = \begin{pmatrix} 0 & c_1 & c_2 & c_3 & \cdots & c_2 & c_1 \\ c_1 & 0 & c_1 & c_2 & \cdots & c_3 & c_2 \\ c_2 & c_1 & 0 & c_1 & \ddots & c_4 & c_3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_1 & c_2 & c_3 & c_4 & \cdots & c_1 & 0 \end{pmatrix},$$

where $c_i = c_{n-i}$ for $i = 1, \ldots, \lfloor \frac{n}{2} \rfloor$ denotes the cost of a length-i edge. That is, the cost of traveling between vertices i and j is

$$c_{ij} = c_{\min\{(i-j) \bmod n, (j-i) \bmod n\}}.$$

See Fig. 1. Importantly, in circulant TSP we do not make the prototypical assumption that edge costs are metric.

Circulant TSP was first studied in the 70's, motivated by waste minimization ([10]) and reconfigurable network design ([21]). Intriguingly, in the 70's Garfinkel [10] showed that circulant TSP can be easily and efficiently solved whenever the number of vertices n is prime (see Section 3). In general, circulant symmetry imposes just enough structure to sometimes – but by no means always – make a formally hard problem tractable. It is not known if this is the case for circulant TSP, and circulant TSP's complexity has been often cited as a significant open problem (e.g., Burkard [6], Burkard, Deĭneko, Van Dal, Van der Veen, and Woeginger [7], and Lawler, Lenstra, Rinnooy Kan, and Shmoys [19]).

Since the 70s, most work on circulant TSP's complexity has been on the simplest non-trivial case of circulant TSP: the *two-stripe symmetric circulant TSP*. This is the special case where exactly

* Corresponding author at: One Dent Drive, Lewisburg PA, 17837, United States of America.
E-mail address: sg040@bucknell.edu (S.C. Gutekunst).

https://doi.org/10.1016/j.orl.2024.107133

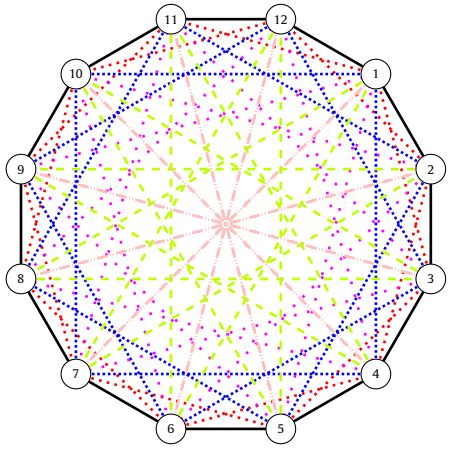

Fig. 1. Circulant symmetry. Edges of a fixed length are indistinguishable and have the same cost. E.g. all edges of the form $\{v, v+1\}$ (where $v+1$ is taken mod n) have length 1, and thus cost c_1.

two of the edge costs $c_1, c_2, ..., c_{\lfloor \frac{n}{2} \rfloor}$ are finite. Greco and Gerace [13] and Gerace and Greco [11] made progress on this case, and recently, Gutekunst, Jin, and Williamson [14] resolved it and showed that the two-stripe symmetric circulant TSP problem is solvable in polynomial time. In parallel, substantial number theoretic work has gone into understanding what collections of edge lengths can constitute a Hamiltonian cycle and/or path (see, e.g., Buratti and Merola [4], Costa, Morini, Pasotti, and Pellegrini [9], and McKay and Peters [20], and Horak and Rosa, Pasotti and Pellegrini [27]).

In this paper, we present three results that, while motivated by recent work on the two-stripe symmetric circulant TSP, apply more generally to circulant TSP:

- Our first result, in Section 3, is the first complexity result for circulant TSP based on the factorization of n since Garfinkel's 70's result [10] that circulant TSP can be efficiently solved when n is prime. Specifically, we show that circulant TSP is also efficiently solvable when the number of vertices n is a prime-squared, continuing to flesh out connections between circulant TSP and number theory.
- In Section 4, we study the *two-class circulant TSP*, which specializes the $(1,2)-$TSP to circulant instances: this is the circulant TSP when the edge costs $c_1, c_2, ..., c_{\lfloor \frac{n}{2} \rfloor} \in \{1,2\}$. Perhaps counter-intuitively, it turns out that the two-class problem is considerably easier than the two-stripe circulant TSP.
- Finally, in Section 5, we return to the two-stripe setting and present a 10/9-approximation algorithm for finding an Eulerian, connected sub-(multi)graph (using all vertices) of minimum cost on two-stripe instances (or equivalently, finding a minimum-cost Hamiltonian cycle on the metric completion of a two-stripe instance). This substantially improves the best previous result for this setting, which is Gerace and Irving [12]'s (4/3)-approximation algorithm for general circulant TSP instances that are also metric (and therefore for finding minimum-cost Eulerian, connected sub-(multi)graphs on any circulant instance).

We begin in Section 2 by briskly introducing background about circulant graphs and their Hamiltonicity, which we will repeatedly make use of.

2. Preliminaries: circulant graphs and hamiltonicity

Let $S \subseteq \{1, 2, ..., \lfloor \frac{n}{2} \rfloor\}$ be a set of edge lengths. We consider *circulant graphs*, which are graphs on vertex set $[n]$ with exactly the edges whose lengths are in S.

Definition 2.1. Let $S \subseteq \{1, ..., \lfloor \frac{n}{2} \rfloor\}$. The **circulant graph** $C\langle S \rangle$ is the (simple, undirected, unweighted) graph on vertex set $[n]$ including exactly the edges whose lengths are in S. I.e., the graph with adjacency matrix $A = (a_{ij})_{i,j=1}^{n}$, where

$$a_{ij} = \begin{cases} 1, & (i-j) \bmod n \in S \text{ or } (j-i) \bmod n \in S \\ 0, & \text{else.} \end{cases}$$

Burkard and Sandholzer [5] studied Hamiltonicity in circulant graphs, and deduced the following:

Proposition 2.2 *(Burkard and Sandholzer [5]). Let* $\{a_1, ..., a_t\} \subseteq \left[\lfloor \frac{n}{2} \rfloor\right]$ *and let* $\mathcal{G} = \gcd(n, a_1, ..., a_t)$. *The circulant graph* $C\langle\{a_1, ..., a_t\}\rangle$ *has* $\mathcal{G}$ *components. The* i*th component, for* $1 \le i \le \mathcal{G}$, *consists of* $n/\mathcal{G}$ *nodes*

$$\left\{ i + \lambda\mathcal{G} \bmod n : \ 0 \le \lambda \le \frac{n}{\mathcal{G}} - 1, \lambda \in \mathbb{Z} \right\}.$$

$C\langle\{a_1, ..., a_t\}\rangle$ *is Hamiltonian if and only if* $\mathcal{G} = 1$.

A complete proof can be found in [5], showing how to recursively construct Hamiltonian cycles whenever $\mathcal{G} = 1$ and giving rise to an $O(n \log(n))$-time algorithm for finding Hamiltonian cycles whenever $\mathcal{G} = 1$. Because many of our results will lean on this proposition, we sketch the idea below, taking care to emphasize details pertinent to our results. We also adopt two notational conventions: First, all vertex labels are implicitly taken modulo n (e.g. $v + a_1$ is shorthand for $(v + a_1) \bmod n$, so that $\{v, v + a_1\}$ is a length-a_1 edge). Second, we use $\equiv_n$ to denote congruence modulo n.

The idea of the proof is to start with a cycle on the vertices

$$\{v : 1 \le v \le n, v \equiv_{\gcd(n,a_1)} 1\};$$

given our notational conventions, we write this more succinctly as $\{v : v \equiv_{\gcd(n,a_1)} 1\}$. Getting such a cycle is straightforward: we start at vertex 1, and follow length-a_1 edges until returning to 1. Then from $s = 2$ to $s = t$, we extend a cycle visiting all vertices in the set $\{v : v \equiv_{\gcd(n,a_1,...,a_{s-1})} 1\}$ to one visiting all vertices in $\{v : v \equiv_{\gcd(n,a_1,...,a_s)} 1\}$. We do so by "copying and translating" our original cycle by multiples of a_s, and then using circulant symmetry to merge these cycles. The proof sketch below makes these ideas more precise. While the proof is not particularly technical, it is notationally cumbersome; we thus begin with an example that captures the process and role of symmetry.

Example 2.3. Suppose $n = 72$, and $S = \{a_1, a_2, a_3, a_4\}$ with $a_1 = 12, a_2 = 24, a_3 = 9$, and $a_4 = 16$. The graph $C\langle\{a_1\}\rangle$ is a cycle cover with $\gcd(n, a_1) = 12$ cycles each of length $n/\gcd(n, a_1) = 6$. We start with the cycle containing vertex 1: we start at vertex 1 and follow length-a_1 edges until we return, yielding the cycle $1, 13, 25, 37, 49, 61, 1$. This cycle visits every vertex in the set $\{1 + \lambda \gcd(n, a_1) : 0 \le \lambda \le \frac{n}{\gcd(n,a_1)} - 1\}$, which is exactly the same as $\{v : v \equiv_{\gcd(n,a_1)} 1\} = \{v : v \equiv_{12} 1\}$.

Then we start our iterative process at $s = 2$, and extend our cycle visiting each vertex in $\{v : v \equiv_{\gcd(n,a_1)} 1\}$ to $\{v : v \equiv_{\gcd(n,a_1,a_2)} 1\}$. Since $\gcd(n, a_1, a_2) = 12 = \gcd(n, a_1)$, we already have such a cycle so we move on to $s = 3$. Since $\gcd(n, a_1, a_2, a_3) = 3$, we will "grow" our cycle on $\{v : v \equiv_{12} 1\}$ to a cycle visiting every vertex in $\{v : v \equiv_3 1\}$. This process is illustrated in Fig. 2. We start by *copying and translating* our cycle $1, 13, 25, 37, 49, 61, 1$ by multiples $a_3 = 9$, so that we have a cycle cover on $\{v : v \equiv_3 1\}$. The *translates* give us a cycle cover consisting of four cycles:

$$1 + \lambda a_3, 13 + \lambda a_3, 25 + \lambda a_3, 37 + \lambda a_3, 49 + \lambda a_3, 61 + \lambda a_3, 1 + \lambda a_3$$

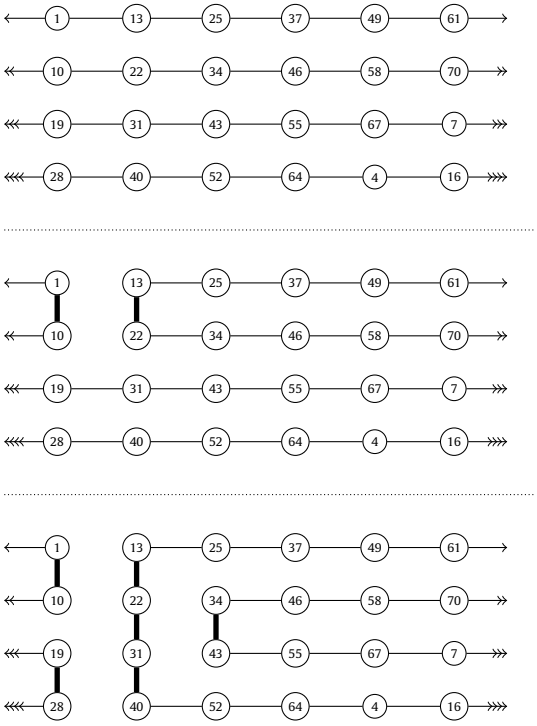

Fig. 2. Example of the iterative process for Proposition 2.2 when $n = 72$ and going from $C\langle\{12, 24\}\rangle$ to $C\langle\{12, 24, 9\}\rangle$. We begin with a cycle visiting every vertex in $\{v : v \equiv_{12} 1\}$ using just length-12 edges and "translate" by multiples of 9 it to form a cycle cover on $\{v : v \equiv_3 1\}$ (top pane). We then use circulant symmetry to patch these cycles together (middle and bottom pane), yielding a cycle visiting every vertex in $\{v : v \equiv_3 1\}$ (bottom pane). Edges with matching numbers of arrowheads "wrap around" and are connected to each other.

for $0 \le \lambda \le 3$, as shown in the top pane of Fig. 2.

Next, we use circulant symmetry to merge these cycles into one cycle on $\{v : 1 \le v \le 72, v \equiv_3 1\}$. To merge the first two cycles, we pick an edge (here, $\{1, 13\}$) in the first cycle, delete it and its *translate* $\{1 + a_3, 13 + a_3\} = \{10, 22\}$ in the second cycle, then add the length-a_3 edges $\{1, 10\}$ and $\{13, 22\}$. We then continue patching in successive cycles, picking an edge in the second cycle, deleting it and its translate in the third cycle, adding length-a_3 edges to merge the third cycle in, and repeating again to merge in the fourth cycle. The end result of this process is shown in the bottom pane of Fig. 2, yielding a cycle visiting every vertex in $\{v : v \equiv_3 1\}$. Since $\gcd(n, a_1, a_2, a_3, a_4) = 1$, the final $s = 4$ step of the algorithm would take our cycle $1, 61, 49, \ldots, 70, 10, 1$, "copy and translate" it by multiples a_4 to get a cycle cover on $\{1, \ldots, 72\}$, and merge these together by deleting pairs of an edge and its translate, then patching them together with length-a_4 edges.

Proof (sketch). The proof of Proposition 2.2 proceeds as in Example 2.3: We begin with a cycle visiting every vertex in $\{v : v \equiv_{\gcd(n,a_1)} 1\}$ attained by starting at vertex 1 and following length-a_1 edges until we return: $1, 1 + a_1, 1 + 2a_1, \ldots, 1 + \left(\frac{n}{gcd(n,a_1)} - 1\right)a_1, 1$. Then we proceed iteratively from $s = 2$ to $s = t$. At each step, we "grow" our cycle visiting every vertex in $\{v : v \equiv_{\gcd(n,a_1,\ldots,a_{s-1})} 1\}$ to a cycle visiting every vertex in $\{v : v \equiv_{\gcd(n,a_1,\ldots,a_{s-1},a_s)} 1\}$. Note that if $\gcd(n, a_1, \ldots, a_{s-1}) = \gcd(n, a_1, \ldots, a_{s-1}, a_s)$, then these two vertex sets are the same, so we can move on to the next value of s without having to do anything (as in the $s = 2$ step in Example 2.3).

Otherwise, say that our cycle on $\{v : v \equiv_{\gcd(n,a_1,\ldots,a_{s-1})} 1\}$ is $v_1, v_2, \ldots, v_{n/\gcd(n,a_1,\ldots,a_{s-1})}, v_1$. We translate this by multiples of a_s to get a cycle cover of $\{v : 1 \le v \le n, v \equiv_{\gcd(n,a_1,\ldots,a_s)} 1\}$. In particular, our cycle cover consists of the $\gcd(n, a_1, \ldots, a_{s-1})/\gcd(n, a_1, \ldots, a_{s-1}, a_s)$ cycles $v_1 + \lambda a_s, v_2 + \lambda a_s, \ldots, v_{n/\gcd(n,a_1,\ldots,a_{s-1})} + \lambda a_s, v_1 + \lambda a_s$, for $0 \le \lambda \le (\gcd(n, a_1, \ldots, a_{s-1})/\gcd(n, a_1, \ldots, a_{s-1}, a_s)) - 1$. We merge the cycles together one-at-a-time, first picking some edge $\{v_1, v_2\}$ in the original cycle, deleting it and its translate $\{v_1 + a_s, v_2 + a_s\}$, and patching them together with two length-a_s edges: $\{v_1, v_1 + a_s\}$ and $\{v_2, v_2 + a_s\}$. This patches together the cycle with $\lambda = 0$ and $\lambda = 1$, and we repeat: from $\lambda = 2$ to $(\gcd(n, a_1, \ldots, a_{s-1})/\gcd(n, a_1, \ldots, a_{s-1}, a_s)) - 1$, we set $j = 1 + ((\lambda + 1) \bmod 2)$, delete $\{v_j + (\lambda - 1)a_s, v_{j+1} + (\lambda - 1)a_s\}$ and its translate $\{v_j + \lambda a_s, v_{j+1} + \lambda a_s\}$, and add $\{v_j + (\lambda - 1)a_s, v_j + \lambda a_s\}$ and $\{v_{j+1} + (\lambda - 1)a_s, v_{j+1} + \lambda a_s\}$. Here, j alternates the pairs of edges we delete (e.g. in the $s = 3$ case of Example 2.3, we deleted translates of $\{1, 13\}$ in our first and third merge, and of $\{13, 25\}$ in our second merge).

In the language of Proposition 2.2, we have thus shown that $C\langle\{a_1, \ldots, a_t\}\rangle$ is Hamiltonian whenever $\mathcal{G} = 1$; whenever $\mathcal{G} > 1$, length $a_1, a_2, \ldots$, and a_t edges will never allow us to leave the component $\{v : v \equiv_{\mathcal{G}} 1\}$ and the graph is not Hamiltonian (and is, in fact, disconnected). However, our algorithm will terminate with a cycle on $\left\{1 + \lambda\mathcal{G} \bmod n : \ 0 \le \lambda \le \frac{n}{\mathcal{G}} - 1\right\}$, which can readily be translated to a cycle on $\left\{i + \lambda\mathcal{G} \bmod n : \ 0 \le \lambda \le \frac{n}{\mathcal{G}} - 1\right\}$ by adding $i - 1$ to each vertex in the cycle. □

3. Circulant TSP and primes

That circulant TSP can be solved in polynomial-time whenever n is prime follows from Proposition 2.2 immediately: Let ℓ denote the length of a cheapest edge in an input to circulant TSP (i.e. $c_\ell = \min\{c_1, c_2, \ldots, c_{\lfloor \frac{n}{2} \rfloor}\}$). Then by Proposition 2.2, the graph $C\langle\{\ell\}\rangle$ is Hamiltonian, and starting at vertex 1 and following edges of length ℓ until you return to vertex 1 yields a minimum-cost Hamiltonian cycle of cost $n \times c_\ell$. This result was first shown in the 70's [10], but since then, no other results relating the complexity of circulant TSP to the factorization of n have been shown.

Our first result extends these connections and shows that, when n is a prime-squared, it is also easy to determine the cost of the circulant TSP solution. In this case, the cost of an optimal solution depends on up to two edge-lengths: the cheapest edge-length ℓ, and (if ℓ is not relatively prime to n), the cheapest edge-length that is relatively prime to n. Note that, if $n = p^2$ for a prime p, the only edge-lengths that are not relatively prime to n are those that are multiples of p. I.e., for $1 \le i \le \lfloor n/2 \rfloor$,

$$\gcd(i, n) = \begin{cases} p, & i \text{ is a multiple of } p, \\ 1, & \text{else.} \end{cases}$$

Theorem 3.1. *Let $n = p^2$ where $p \ge 3$ is a prime. Let ℓ denote the length of a cheapest edge in an input to circulant TSP (i.e. $c_\ell = \min\{c_1, c_2, \ldots, c_{\lfloor \frac{n}{2} \rfloor}\}$). Let $S = \{c_i : 1 \le i \le \lfloor \frac{n}{2} \rfloor, \gcd(n, i) = 1\}$ denote the set of edge-lengths relatively prime to n, and let s denote the length of a cheapest edge in that set (i.e. $c_s = \min\{S\}$). A minimum-cost Hamiltonian cycle costs $(n - p) \times c_\ell + p \times c_s$.*

Note that if $c_\ell = c_s$ (e.g., if $\gcd(\ell, n) = 1$), then the circulant graph $C\langle\{\ell\}\rangle$ is Hamiltonian and $(n - p) \times c_\ell + p \times c_s = nc_\ell$. Otherwise, by Proposition 2.2, $C\langle\{\ell\}\rangle$ has $n/\gcd(n, p) = n/p = p$ components, each of which has p vertices (and all vertices in a component will be congruent mod p). We adopt a convention for plotting $C\langle\{\ell, s\}\rangle$ in terms of these components, shown in Fig. 3: we start by drawing the first component of $C\langle\{\ell\}\rangle$, consisting of all vertices congruent to 1 mod p, connected by length-ℓ edges in a cycle that "wraps around" vertically. Then we translate this component by s, plotting all vertices congruent to $(1 + s)$ mod p in the next

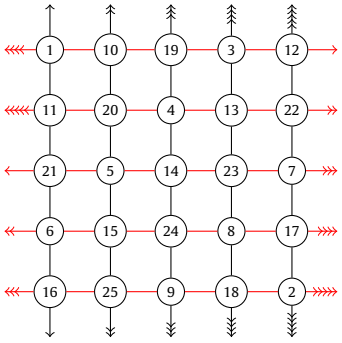

Fig. 3. Convention for plotting $C\langle\{\ell, s\}\rangle$ when $n = 25$, $\ell = 10$, and $s = 9$. Length-ℓ edges are vertical (and in black) and length-s edges are horizontal (and in red). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

column, arranged so that horizontally adjacent vertices are connected by length-s edges. We repeat this process, forming a grid, until we reach the component consisting of all vertices congruent to $(1 + (p-1)s) \bmod p$ in the rightmost column. The vertices in this last column are then connected back to vertices in the first column by length-s edges (i.e. $1 + (p-1)s + s \equiv_p 1$, so a length-$s$ edge from a vertex in the last column wraps back around to a vertex in the first column). However, these length-s edges between the last and first column do not necessarily wrap around to the same row. See Fig. 3, for instance, and see [15] for more general results on the structure of circulant graphs.

Before proving Theorem 3.1, we need one basic fact about linear congruences. See, e.g., Theorem 57 of [16].

Proposition 3.2. *The linear congruence*

$$ax \equiv_n b$$

has a solution x if and only if $\gcd(a, n)$ divides b. Moreover, there are exactly $\gcd(a, n)$ solutions which take the form

$$x_0 + \lambda \frac{n}{\gcd(a, n)}, \ \lambda = 0, 1, \ldots, \gcd(a, n) - 1$$

for some $0 \le x_0 < \frac{n}{\gcd(a,n)}$.

Proof (of Theorem 3.1). If $\gcd(\ell, n) = 1$, then starting at vertex 1 and following edges of length ℓ yields a Hamiltonian cycle of cost $n \times c_\ell$ (and since any Hamiltonian cycle must use n edges all of which cost at least c_ℓ, this is optimal).

Otherwise, $C\langle\{\ell\}\rangle$ has p components and any Hamiltonian cycle must cost at least $(n-p) \times c_\ell + p \times c_s$: Such a cycle must use at least p edges of cost at least c_s, as all edges cheaper than c_s stay within components of $C\langle\{\ell\}\rangle$, and at least p other edges are needed to connect these components in a cycle.

Thus it suffices to show that a Hamiltonian cycle using $(n-p)$ length-ℓ edges and p length-s edges exists. Note that such a cycle will use exactly one length-s edge between each pair of adjacent columns (adjacent in the sense of our drawing convention from Fig. 3), and then one final length-s edge wrapping from the last column to the first. We will first construct a Hamiltonian path, starting at vertex 1 and traversing through the columns one-at-a-time. In doing so, the Hamiltonian path will use $(p-1)$ edges of length s (one between the first and second column, one between the second and third, ..., and one between the $(p-1)$th and pth). We will traverse the columns so that the Hamiltonian path ends at vertex $1 - s$. Then, we can extend it to a Hamiltonian cycle by taking a final length-s edge to vertex 1. In constructing the Hamiltonian path, the only choice we have is how we traverse each column: moving "up" (i.e. from vertex v to $(v - \ell) \bmod n$) or "down" (i.e. from vertex v to $(v + \ell) \bmod n$). We need to show that we can choose a sequence of "up" and "down" columns so that the last vertex visited in the final column is $1 - s$.

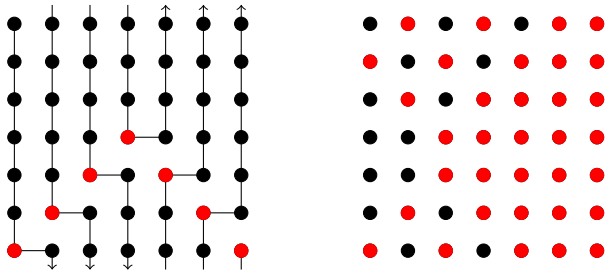

Fig. 4. One possible sequence of up- and down-moves with the final vertex visited in each column marked in red (left). All possible final vertices marked in each column when $n = 7^2$ (right).

The intuition for this process is shown in Fig. 4, which first shows an example sequence of "ups" and "downs" where we traverse down in the first four columns and then up in the last three. Note that, each time we go down in a column, the final vertex visited in that column is one row above where we entered that column (modulo the number of rows); each time we go up, the final vertex visited in that column is one row below where we entered that column (modulo the number of rows). The red vertices trace out the last vertex visited in each column. On the right of Fig. 4, we trace out (in red) the final vertices we can reach in a column by any sequence of ups and downs: there are two red vertices in the first column (based on whether we go up or down), then three red vertices in the second column (corresponding to going down twice, going down once and up once, or going up twice), and so on. We see that we can choose a sequence of ups and downs to reach every vertex in the last column: no matter what row $1 - s$ is in, we will be able to reach it.

More formally, suppose that vertex $1 - s$ is in row r, with $0 \le r < p$ (indexing the top row as row 0). We need to show that, regardless of r, we can choose a sequence of ups and downs to end our Hamiltonian path at the vertex in row r of the last column (i.e., at $1 - s$). If we choose to go down k times and up $p - k$ times, we end in row $((-k) + (p-k)) \bmod p$. It thus suffices to show that we can choose k, with $0 \le k \le p$, such that $p - 2k \equiv_p r$. That is, $2k \equiv_p p - r$. Since $p \ne 2$, we have that $\gcd(2, p) = 1$ and by Proposition 3.2, there is a unique solution $0 \le k < p$. This value of k gives rise to a Hamiltonian path from 1 to $1 - s$ using exactly $p - 1$ edges of length-s; taking one final length-s edge from $1 - s$ to 1 yields the desired Hamiltonian cycle. □

Algorithmically, note that we can find the row r of $1 - s$ by solving the following congruence for r:

$$1 - s \equiv_n (p-1) \times s + r \times \ell.$$

We can then solve $2k \equiv_p p - r$ to attain the desired value of k; both congruences can be solved using the extended Euclidean algorithm in $O(\log^2(n))$ time. See, for example, Theorem 4.4 in [30].

4. The two-class TSP

One particularly well-studied variant of symmetric TSP is the $(1, 2)$-TSP, where all edges have cost 1 or 2 (these instances are not necessarily circulant). This variant is NP-hard and is a special case of the more general metric TSP, but better approximation algorithms are known for the $(1, 2)-$TSP than for the metric TSP (see, e.g., [1,3,18,26]).

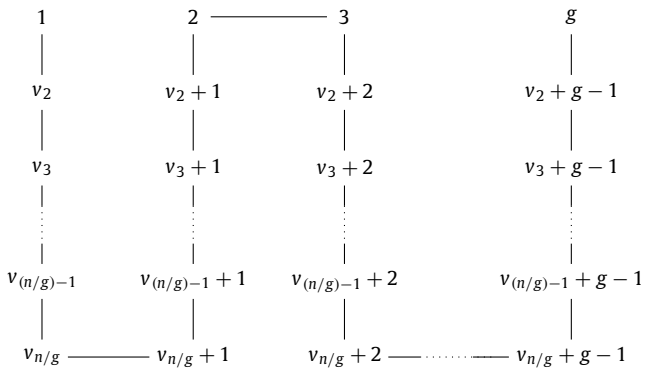

Fig. 5. Extending a Hamiltonian path on one component of $C\langle S\rangle$ to a Hamiltonian path on all vertices in the proof of Theorem 4.1 when g is even. When g is odd, the Hamiltonian path ends at $v_{n/g}+g-1$.

In this section, we consider the $(1,2)-$TSP specialized to circulant instances, the two-class circulant TSP: Here, the c_i can take on exactly two distinct values; without loss of generality, these values are 1 ("cheap") or 2 ("expensive"). That is, $c_i \in \{1,2\}$ for all $1 \le i \le \lfloor \frac{n}{2} \rfloor$. We show that two-class circulant TSP instances can be efficiently solved.

Theorem 4.1. *Consider an instance of the two-class circulant TSP where $S := \{i : c_i = 1\}$ denotes the set of cheap edge-lengths, and let $g := \gcd(n, S)$ denote the GCD of n and all edge-lengths in S. Then the optimal solution to this instance has cost:*

$$\begin{cases} n, & g = 1 \\ n+g, & g > 1. \end{cases}$$

Proof. First, suppose that $g = 1$. Then by Proposition 2.2, the graph $C\langle S\rangle$ is Hamiltonian, so Proposition 2.2 gives a Hamiltonian cycle just using cost-1 edges.

Otherwise, $g > 1$, which implies that $1 \notin S$ and the graph $C\langle S\rangle$ has g components. Any Hamiltonian cycle must thus use at least g expensive edges and hence cost at least $n+g$. To construct such a Hamiltonian cycle, we start by building a path $v_1 = 1, v_2, \ldots v_{n/g}$ starting at vertex 1 and visiting all vertices in the component of $C\langle S\rangle$ including vertex 1; this can be done using the algorithm in Proposition 2.2 (obtaining a Hamiltonian cycle, say, on the subgraph consisting of $\{v : 1 \le v \ne n, v \equiv_g 1\}$ and deleting one of the edges incident to vertex 1). We translate this Hamiltonian path to the other components of $C\langle S\rangle$ by length-1 edges, obtaining Hamiltonian paths

$$v_1 + k, v_2 + k, \ldots v_{n/g} + k$$

for $k = 0, 1, \ldots, (g-1)$. We join these paths with length-1 edges as in Fig. 5, adding the edges $\{v_{n/g}+k, v_{n/g}+k+1\}$ for k odd and $\{k, k+1\}$ for k even. This yields a Hamiltonian path from vertex 1 to either $v_1 + (g-1) = g$ (if g is even) or $v_{n/g} + (g-1)$ (if g is odd). Note, however, that both are adjacent to 1 by a cost-2 edge: otherwise, they would be in the same component of $C\langle S\rangle$. Thus, we can extend this Hamiltonian path to a Hamiltonian cycle, and we used exactly g cost-2 edges. □

5. Minimum-cost Eulerian tours

In our final section, we consider a problem related to the two-stripe circulant TSP: finding an Eulerian, connected sub-(multi)graph (including all vertices) of minimum cost on a two-stripe circulant instance. That is, a minimum-cost tour visiting every vertex, potentially repeating edges, on a two-stripe circulant instance. Classic algorithms for the metric TSP (such as the double-tree and Christofides–Serdyukov algorithm [8,29,31]) begin by finding such a tour; our problem is equivalent to finding a Hamiltonian tour on the metric completion of a two-stripe circulant TSP instance. Gerace and Irving [12] give a (4/3)-approximation algorithm for general circulant TSP instances that are also metric (and therefore for finding minimum-cost Eulerian, connected sub-(multi)graphs on any circulant instance). Here, we show that the ratio can be improved to (10/9) when considering a two-stripe instance.

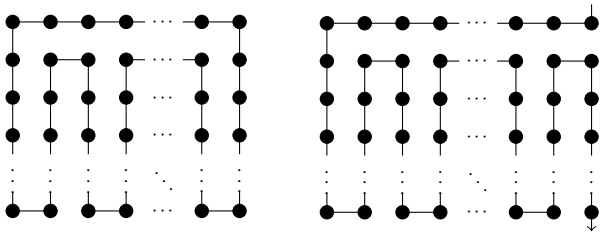

Fig. 6. Feasible Hamiltonian cycles when g is even (left) and odd (right), which are optimal Eulerian tours when $c_i = c_j$.

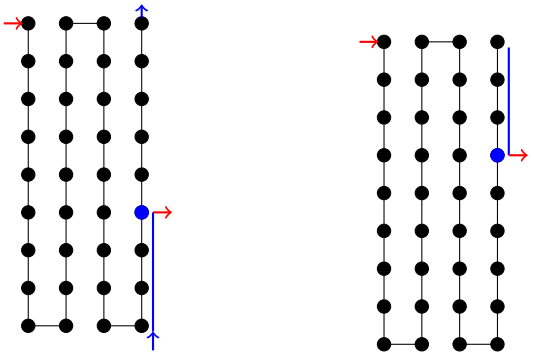

Fig. 7. Optimal Eulerian tour when $c_i = 0$, depending on the row of the vertex $(1-j) \bmod n$ within the last column.

More specifically, consider a circulant instance with two finite edge costs $0 \le c_i \le c_j < \infty$. We assume that $\gcd(n, i, j) = 1$: otherwise, by Proposition 2.2, the graph $C\langle\{i, j\}\rangle$ will not be connected. Let $g = \gcd(n, i)$. If $g = 1$, then the length-i edges form a Hamiltonian cycle, and form an optimal Eulerian tour of cost $n \times c_i$. Otherwise, $C\langle\{i\}\rangle$ consists of g components, and an Eulerian tour costs at least $(n-g) \times c_i + g \times c_j$.

To motivate our result, consider two extremal cases. First, if $c_i = c_j$, then any Hamiltonian cycle is optimal. Fig. 6 sketches such a Hamiltonian cycle, based on the parity of g. As in the drawing convention from Fig. 3, we plot components of $C\langle\{i\}\rangle$ as columns. With this convention, "cheap" length-i edges are vertical and "expensive" length-j edges are horizontal. Conversely, if $c_i = 0$, we can frivolously use length-i (vertical) edges. We consider an Eulerian tour as shown in Fig. 7: There will be some vertex $(1-j) \bmod n$ in the last column (highlighted in blue) connected to vertex 1 by a length-j edge. We follow a zig-zagging Hamiltonian path ending at either the bottom or top vertex in that column, which has n/g vertices in it. We then take extra length-i edges to reach $(1-j) \bmod n$: either "wrapping around" (as in the left of Fig. 7) or "turning around" the same column (as in the right), depending on whichever uses fewer length-i edges. Since there are n/g length-i edges in this column, one direction will use at most $n/(2g)$ extra length-i edges; when $c_i = 0$ these tours cost exactly $g \times c_j$, which is optimal.

Our final result gives rise to a 10/9-approximation algorithm: it shows that at least one of these extremal tours will always be

within 10/9 of the minimum-cost Eulerian tour. To make the analysis more clean, we scale all edge costs by $1/c_j$ (if $c_j = 0$, then $c_i = c_j = 0$ and our tour from Fig. 6 is optimal) and define $c := \frac{c_i}{c_j}$. Then our cheaper edges cost $0 \le c \le 1$, and our expensive edges cost 1. Thus our tours from Fig. 6 cost

$$(n - 2(g-1))\,c + 2(g-1).$$

Our tours from Fig. 7 use $(n/g) - 1$ length-i edges in each of the g columns in the original Hamiltonian path, at most $n/(2g)$ additional length-i edges in the last column, and exactly g length-j edges. All together, they thus cost at most

$$g\left(\frac{n}{g} - 1\right)c + \frac{n}{2g}c + g = \left(n - g + \frac{n}{2g}\right)c + g.$$

Theorem 5.1. *Consider a two-stripe circulant input where* $\gcd(n, i, j) = 1$ *and* $g := \gcd(n, i) > 1$. *At least one of the tours shown in Figs. 6 and 7 has cost within (10/9) of the minimum-cost Eulerian tour.*

Proof. First we note that any Eulerian tour must use at least n edges, and at least g of these must be length-j to fully connect the components of $C\langle\{i\}\rangle$ and return to vertex 1. Thus, any Eulerian tour costs at least $(n-g)c + g$. Note also that this implies that, if $g = 2$, the Eulerian tours from Fig. 6 are optimal.

Hence, we assume that $g \ge 3$. We will benchmark each tour against the lower bound, and thus want to show that

$$\min\left\{\frac{(n-2(g-1))\,c + 2(g-1)}{(n-g)c+g}, \frac{\left(n - g + \frac{n}{2g}\right)c + g}{(n-g)c+g}\right\} \le \frac{10}{9}.$$

Equivalently, that

$$\min\left\{1 + \frac{(g-2)(1-c)}{(n-g)c+g}, 1 + \frac{\frac{n}{2g}c}{(n-g)c+g}\right\} \le \frac{10}{9},$$

or that

$$\min\left\{\frac{(g-2)(1-c)}{(n-g)c+g}, \frac{\frac{n}{2g}c}{(n-g)c+g}\right\} \le \frac{1}{9}.$$

Straightforward calculus shows that $\frac{(g-2)(1-c)}{(n-g)c+g}$ is decreasing in c for $c \in [0,1]$ (and decreases from $\frac{g-2}{g}$ to 0), while $\frac{\frac{n}{2g}c}{(n-g)c+g}$ is increasing in c for $c \in [0,1]$ (and increases from 0 to $1/(2g)$). Thus, the worst-case value of $\min\{\frac{(g-2)(1-c)}{(n-g)c+g}, \frac{\frac{n}{2g}c}{(n-g)c+g}\}$ occurs when both arguments are equal: when

$$(g-2)(1-c) = \frac{n}{2g}c.$$

See, for instance, Fig. 8.

Solving for c, we find

$$c = \frac{2g(g-2)}{n + 2g(g-2)}.$$

Plugging in for c, we find that

$$\begin{aligned}
\frac{\frac{n}{2g}c}{(n-g)c+g} &= \frac{\frac{n}{2g}\frac{2g(g-2)}{n+2g(g-2)}}{(n-g)\frac{2g(g-2)}{n+2g(g-2)} + g} \\
&= \frac{\frac{n(g-2)}{n+2g(g-2)}}{(n-g)\frac{2g(g-2)}{n+2g(g-2)} + g} \\
&= \frac{n(g-2)}{(n-g)2g(g-2) + g(n+2g(g-2))} \\
&= \frac{n(g-2)}{2ng(g-2) - 2g^2(g-2) + gn + 2g^2(g-2)} \\
&= \frac{g-2}{2g^2 - 3g}.
\end{aligned}$$

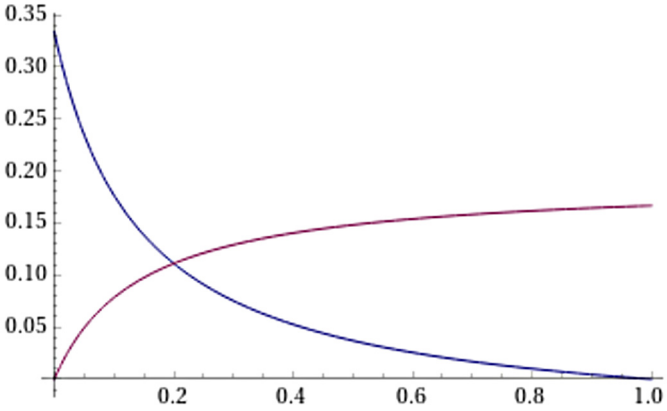

Fig. 8. Example of $\frac{(g-2)(1-c)}{(n-g)c+g}$ (blue) and $\frac{\frac{n}{2g}c}{(n-g)c+g}$ (red) when $n = 24$ and $g = 3$.

Hence,

$$\min\left\{\frac{(g-2)(1-c)}{(n-g)c+g}, \frac{\frac{n}{2g}c}{(n-g)c+g}\right\} \le \frac{g-2}{2g^2-3g}.$$

Finally, we note that $\frac{g-2}{2g^2-3g}$ is decreasing in g for $g \ge 3$, and at $g = 3$, $\frac{g-2}{2g^2-3g} = 1/9$. This completes our proof. □

CRediT authorship contribution statement

Austin Beal: Investigation, Software, Writing – review & editing. **Yacine Bouabida:** Investigation, Software, Writing – review & editing. **Samuel C. Gutekunst:** Conceptualization, Investigation, Supervision, Writing – original draft, Writing – review & editing. **Asta Rustad:** Investigation, Software, Writing – review & editing.

Acknowledgements

The paper was supported by the National Science Foundation, Grant No. CCF-2153331. We thank the referee for thoughtful feedback. Special thanks also to Billy Jin for careful feedback on a draft.

References

[1] A. Adamaszek, M. Mnich, K. Paluch, New approximation algorithms for (1, 2)-TSP, in: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
[2] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, J. ACM 45 (5) (1998) 753–782.
[3] P. Berman, M. Karpinski, 8/7-approximation algorithm for (1, 2)-TSP, in: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, 2006, pp. 641–648.
[4] M. Buratti, F. Merola, Dihedral Hamiltonian cycle systems of the cocktail party graph, J. Comb. Des. 21 (1) (2013) 1–23.
[5] R. Burkard, W. Sandholzer, Efficiently solvable special cases of bottleneck travelling salesman problems, Discrete Appl. Math. 32 (1) (1991) 61–76.
[6] R.E. Burkard, Efficiently solvable special cases of hard combinatorial optimization problems, Math. Program. 79 (1) (1997) 55–69.
[7] R.E. Burkard, V.G. Deĭneko, R. van Dal, J.A.A. van der Veen, G.J. Woeginger, Well-solvable special cases of the traveling salesman problem: a survey, SIAM Rev. 40 (3) (1998) 496–546.
[8] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Technical report, Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.
[9] S. Costa, F. Morini, A. Pasotti, M.A. Pellegrini, A problem on partial sums in Abelian groups, Discrete Math. 341 (3) (2018) 705–712.
[10] R.S. Garfinkel, Minimizing wallpaper waste, part 1: a class of traveling salesman problems, Oper. Res. 25 (5) (1977) 741–751.

[11] I. Gerace, F. Greco, The travelling salesman problem in symmetric circulant matrices with two stripes, Math. Struct. Comput. Sci. 18 (1) (2008) 165–175.
[12] I. Gerace, R. Irving, The traveling salesman problem in circulant graphs, Technical report, TR-1998-15, University of Glasgow, Department of Computing Science, 1998.
[13] F. Greco, I. Gerace, The traveling salesman problem in circulant weighted graphs with two stripes, Electron. Notes Theor. Comput. Sci. 169 (2007) 99–109.
[14] S.C. Gutekunst, B. Jin, D.P. Williamson, The two-stripe symmetric circulant TSP is in P, in: International Conference on Integer Programming and Combinatorial Optimization, Springer, 2022, pp. 319–332.
[15] S.C. Gutekunst, D.P. Williamson, Characterizing the integrality gap of the subtour LP for the circulant traveling salesman problem, SIAM J. Discrete Math. 33 (4) (2019) 2452–2478.
[16] G. Hardy, E. Wright, An Introduction to the Theory of Numbers, 6th edition, Oxford University Press, 2008.
[17] A.R. Karlin, N. Klein, S.O. Gharan, A (slightly) improved approximation algorithm for metric TSP, in: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, 2021, pp. 32–45.
[18] M. Karpinski, R. Schmied, On approximation lower bounds for TSP with bounded metrics, Electron. Colloq. Comput. Complex. 19 (8) (2012).
[19] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, John Wiley and Sons, 1985.
[20] B.D. McKay, T. Peters, Paths through equally spaced points on a circle, arXiv preprint, arXiv:2205.06004, 2022.
[21] E. Medova, Using QAP bounds for the circulant TSP to design reconfigurable networks, in: Quadratic Assignment and Related Problems, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, May 20-21, 1993, 1993, pp. 275–292.
[22] J.S.B. Mitchell, Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems, SIAM J. Comput. 28 (4) (1999) 1298–1309.
[23] T. Mömke, O. Svensson, Removing and adding edges for the traveling salesman problem, J. ACM 63 (1) (2016) 2:1–2:28.
[24] M. Mucha, $\frac{13}{9}$-approximation for graphic TSP, Theory Comput. Syst. 55 (4) (2014) 640–657.
[25] S. Oveis Gharan, A. Saberi, M. Singh, A randomized rounding approach to the traveling salesman problem, in: Proceedings of the 52nd Annual IEEE Symposium on the Foundations of Computer Science, 2011, pp. 550–559.
[26] C.H. Papadimitriou, M. Yannakakis, The traveling salesman problem with distances one and two, Math. Oper. Res. 18 (1) (1993) 1–11.
[27] A. Pasotti, M.A. Pellegrini, A new result on the problem of Buratti, Horak and Rosa, Discrete Math. 319 (2014) 1–14.
[28] A. Sebő, J. Vygen, Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs, Combinatorica 34 (5) (2014) 597–629.
[29] A. Serdyukov, On some extremal walks in graphs, Upravlyaemye Sistemy 17 (1978) 76–79.
[30] V. Shoup, A Computational Introduction to Number Theory and Algebra, Cambridge University Press, 2009.
[31] D.P. Williamson, D.B. Shmoys, The Design of Approximation Algorithms, Cambridge University Press, New York, 2011.