POLYHEDRAL TECHNIQUES IN COMBINATORIAL OPTIMIZATION: MATCHINGS AND TOURS

OLA SVENSSON

ABSTRACT

We overview recent progress on two of the most classical problems in combinatorial optimization, namely the matching problem and the traveling salesman problem. We focus on deterministic parallel algorithms for the perfect matching problem and the first constant-factor approximation algorithm for the asymmetric traveling salesman problem.
While these questions pose seemingly different challenges, recent progress has been achieved using similar polyhedral techniques. In particular, for both problems, we use linear programming formulations, even exponential-sized ones, to extract structure from problem instances to guide the design of algorithms.

MATHEMATICS SUBJECT CLASSIFICATION 2020

Primary 68W01; Secondary 68W20, 68W25, 68Q25, 68R10, 05C85

KEYWORDS

Approximation algorithms, combinatorial optimization, derandomization, linear programming, matchings, traveling salesman problem

ICM INTERNATIONAL CONGRESS OF MATHEMATICIANS 2022 JULY 6—14

Proc. Int. Cong. Math. 2022, Vol. 6, pp. 4970–4994
DOI 10.4171/ICM2022/127

1. INTRODUCTION

[1] The matching problem and the traveling salesman problem are at the heart of combinatorial optimization. In the matching problem, the goal is to pair up the vertices using the edges of the given graph. Work on matchings has contributed to the development of many core concepts in modern computer science, including linear-algebraic, probabilistic, online, streaming, and parallel algorithms. Matchings has been used to show the limitations of several models of computation such as monotone circuits and linear programs (extension complexity). Edmonds was the first to give a polynomial-time algorithm for it. His landmark paper [16] from 1965 is often credited with the idea that polynomial-time is a good abstraction of time efficiency and thus, in turn, popularizing the central complexity class P. However, half a century later, we still do not have a full understanding of the matching problem in many relevant settings such as parallel and space-bounded algorithms.

The traveling salesman problem—of finding the shortest tour of n given cities—is perhaps the most famous benchmark problem for NP-hard optimization problems, similar to what the matching problem is for efficiently solvable problems (i.e., problems in P). Indeed, the study of the traveling salesman problem has played a key role in the development and evaluation of heuristics, integer programming solvers, and approximation algorithms. Already in 1954, Dantzig et al. [14] used a linear program to solve a 49-city instance. The strength of this linear program, often referred to as the subtour elimination relaxation or Held–Karp relaxation, is one of the most fascinating open problems in combinatorial optimization.

While making progress on these questions has been notoriously difficult, there have been recent exciting advancements that we overview in this article. In the first part (Section 2), we survey progress on finding the efficient deterministic algorithms that we *know should exist* ever since the beautiful randomized algorithms for matching problems by Lovász [33] and Mulmuley, Vazirani, and Vazirani [36] were discovered several decades ago. We focus on the breakthrough work by Fenner, Gurjar, and Thierauf [18, 19] that (almost) derandomized the Isolation Lemma of [36] in the special case of perfect matchings in bipartite graphs. We then explain the challenges that we needed to overcome to extend their result to general graphs [46]. Finally, while the work of [19] has inspired much progress, it remains elusive to completely derandomize the approach of [36]. In Section 2.4, we give some fascinating open questions in this line of work.

In the second part (Section 3), we consider the traveling salesman problem. Recent years have seen exciting progress on longstanding open problems: for the asymmetric version, we achieved the first constant-factor approximation algorithm in [48]; and for the symmetric version, Karlin, Klein, and Oveis Gharan [28] made the first improvement on the classic approximation algorithm by Christofides [10] and Serdyukov [43]. We note that our focus in Section 3 is almost exclusively on approximation algorithms for the asymmetric

1 Part of the writing in this overview article is taken from a grant proposal of the author, and some descriptions (as we point out) are taken from our works [46, 48].

traveling salesman problem. We do not discuss the recent results for the symmetric version in detail. We do, however, point out similarities between one approach for the asymmetric case and the breakthrough algorithm in [28]. Finally, in Section 3.5, we give some of our favorite open problems related to the traveling salesman problem.

While the matching problem and the traveling salesman problem pose seemingly different challenges, much of the recent progress has been achieved using similar algorithmic ideas. Specifically, *polyhedral techniques*—the study of the convex hull of integer solutions and finding/exploiting properties of it—have played a key role in both our works on the matching problem [46] and the asymmetric traveling salesman problem [48]. In this overview, we highlight one polyhedral property that is central in both results, the so-called laminar structure of extreme points. We have made an effort to keep notation light and to introduce concepts when they are used. For a comprehensive reading on polyhedral techniques for combinatorial optimization, we recommend the excellent book by Schrijver [41].

2. THE ALGORITHMS THAT MUST EXIST FOR PERFECT MATCHINGS

A central problem in combinatorial optimization is the (maximum weight) perfect matching problem. Recall that a perfect matching of a graph is a subset of the edges so that every vertex is incident to exactly one edge, i.e., all vertices are paired up using edges. The *maximum weight perfect matching* problem now simply asks us to find a perfect matching of maximum weight in a given edge-weighted graph. This problem has a long and beautiful history.

For bipartite graphs, the Hungarian method is often taught in algorithm classes as a prime example of the powerful primal–dual technique. It is referred to as the Hungarian method as it relies on ideas developed by the Hungarians König and Egerváry, although amazingly we now know that the algorithm was already discovered by Jabobi more than a century ago [26]! One advantage of bipartite graphs over general graphs is that the polyhedral structure, i.e., the exact linear programming formulation, of the perfect matching problem is significantly simpler for bipartite graphs. It was in 1965 that Edmonds developed the first polynomial-time algorithm for general graphs [16]. Alongside his algorithmic discovery, he also formulated his famous description of the perfect matching polytope [15].

We have thus known efficient, i.e., polynomial-time, deterministic algorithms for the maximum weight perfect matching problem in both bipartite and general graphs for over half a century. Surprisingly, the situation is dramatically different for the following slight changes to the objective:

- *k-Extendable matching:* find a perfect matching that maximizes the sum of weights of the k heaviest edges or, equivalently, find k edges of maximum weight that can be extended to a perfect matching.

- *Exact matching:* decide (or find if it exists) whether a given edge-weighted graph has a perfect matching of weight *equal* to a target W.[2]

The status of both these problems is very intriguing: *we have yet to discover efficient deterministic algorithms that we have known should exist for decades!* But how can we be so sure that the above mentioned problems should admit efficient deterministic algorithms? The answer to this question lies in the fundamental study of the power of randomization in algorithm design. There is strong evidence (see, e.g., [25]) that any problem that admits an efficient randomized algorithm also admits an efficient deterministic algorithm. This may at first be surprising as there are several problems for which only efficient randomized algorithms are known. However, complexity theory tells us that this discrepancy is very likely due to a lack of algorithmic techniques and not due to a fundamental difference in the power of randomized and deterministic polynomial-time computation.

If we allow randomization, there is an incredibly versatile algorithmic technique developed by Mulmuley, Vazirani, and Vazirani [36], building upon earlier work by Lovász [33]. Specifically, the algorithmic technique in [36] yields efficient randomized *parallel* algorithms for all the above-mentioned problems: maximum weight perfect matching, k-extendable matching, and exact matching. While the obtained randomized algorithms are relatively simple and even parallelizable, it is a notorious problem to remove the need for randomness, i.e., to derandomize the approach. This is true even for vanilla perfect matchings and devising an efficient *deterministic parallel* algorithm for the (unweighted) perfect matching problem is a long-standing open problem.

While it remains elusive to derandomize the algorithmic technique in [36] completely, there has been significant progress in the last few years starting with the groundbreaking work of Fenner, Gurjar, and Thierauf [18, 19]. We give an overview of this progress with a focus on parallel algorithms for the perfect matching problem. In Section 2.1 we briefly explain the main techniques in the mentioned randomized parallel algorithms. We then give a more detailed description of the elegant framework of [19] which was the starting point of much of the recent progress. In Section 2.3 we then explain the challenges and our ideas for generalizing the result of [19] from biparite to general graphs. Finally, in Section 2.4, we comment on open questions and why we think progress on the exact matching and the k-extendable matching problems requires major new ideas.

2.1. Randomized parallel algorithms for the perfect matching problem

The description in this section of randomized parallel algorithms for the perfect matching problem is mainly taken from [46]. For those algorithms, linear-algebraic techniques have been a very powerful approach. They rely on the Tutte matrix of a graph

2 We remark that for the exact matching problem to be tractable, we assume that the edge-weights are polynomially bounded; otherwise, NP-completeness follows immediately from subset sum. In fact, the problem can be reduced to the case when edge-weights only take values 0 and 1, which is referred to as the exact red–blue matching problem.

$G = (V, E)$, which is the $|V| \times |V|$ matrix defined as follows:

$$T(G)_{u,v} = \begin{cases} X_{(u,v)} & \text{if } (u,v) \in E \text{ and } u < v, \\ -X_{(v,u)} & \text{if } (u,v) \in E \text{ and } u > v, \\ 0 & \text{if } (u,v) \notin E, \end{cases}$$

where we ordered the vertices arbitrarily and $X_{(u,v)}$ for $(u, v) \in E$ are variables. Tutte's theorem [52] says that $\det T(G) \neq 0$ if and only if G has a perfect matching.

A natural algorithm based on the Tutte matrix, replaces each indeterminate by a random value from a large field and then computes the determinant. If the graph has a perfect matching, the Schwartz–Zippel lemma ensures that the value of the computed determinant is nonzero with high probability. Furthermore, as computing determinants can be done efficiently in parallel [7, 13], this yields an efficient *randomized* parallel algorithm for *deciding* whether a graph has a perfect matching [33].

A second approach that will be more amenable to derandomization, adopted by Mulmuley, Vazirani, and Vazirani [36] for the search version (see also [29] for another randomized parallel algorithm for the search version, i.e., for finding a perfect matching if it exists), is to replace the indeterminates by randomly chosen powers of two. Namely, for each edge (u, v), a random weight $w(u, v) \in \{1, 2, \ldots, 2|E|\}$ is selected, and we substitute $X_{(u,v)} := 2^{w(u,v)}$. Now, let us make the crucial assumption that one perfect matching M is *isolated*, in the sense that it is the *unique minimum-weight perfect matching* (minimizing $w(M) = \sum_{e \in M} w(e)$). Then $\det T(G)$ remains nonzero after the substitution: one can show that M contributes a term $\pm 2^{2w(M)}$ to $\det T(G)$, whereas all other terms are multiples of $2^{2w(M)+1}$ and thus they cannot cancel $2^{2w(M)}$ out. The determinant can still be computed efficiently in parallel as all entries $2^{w(u,v)}$ of the matrix are of polynomial bit-length, and so we have a parallel algorithm for the decision version. An algorithm for the search version also follows: for every edge in parallel, test whether removing it causes this least-significant digit $2^{2w(M)}$ in the determinant to disappear; output those edges for which it does. The final ingredient of the randomized approach of [36] is that assigning random weights to edges does indeed isolate one matching with constant probability. This is known as the Isolation Lemma and is a powerful concept that turns out to be true in the much more general setting of arbitrary set families.

2.2. Fenner, Gurjar, and Thierauf's approach for bipartite graphs

The elegant framework introduced by Fenner, Gurjar, and Thierauf [19] has been the basis for many subsequent developments including our result on general graphs that we describe in the next section. Their starting point is the randomized algorithm of Mulmuley, Vazirani, and Vazirani [36]. It forms an attractive starting point for derandomization because the only randomized ingredient is the selection of the weight function w and there is a simple condition that guarantees its correctness: the algorithm succeeds if the weight function w is *isolating*, i.e., there is a unique minimum weight perfect matching with respect to w. Thus to find a deterministic parallel algorithm for the perfect matching it is sufficient to find (in parallel) an isolating weight function with polynomial values. In other words, we would like to derandomize the Isolation Lemma.

Fenner, Gurjar, and Thierauf's construction of isolating weight functions is actually oblivious to the considered graph. Specifically, for any $n \in \mathbb{N}$, they construct a family $\mathcal{F}_n$ of simple weight functions such that for *any* bipartite n-vertex graph there is an isolating weight function $w \in \mathcal{F}_n$. To completely derandomize [36], the weight functions in $\mathcal{F}_n$ should be simple (efficiently computable in parallel) and satisfy the following three conditions:

(1) For every n-vertex graph G, there is an isolating weight function $w \in \mathcal{F}_n$.

(2) The number of weight functions in $\mathcal{F}_n$ is at most a polynomial in n.

(3) Each weight function in $\mathcal{F}_n$ assigns integer weights that are bounded by a polynomial in n.

The last condition ensures that we can calculate the determinant of the Tutte matrix where we replaced X_e by $2^{w(e)}$ efficiently for every $w \in \mathcal{F}_n$. The first condition ensures that we are guaranteed to succeed if we try all weight functions in our family and the second condition says that we can afford to try all of them (polynomially many) in parallel.

We remark that the construction of $\mathcal{F}_n$ becomes trivial if we drop the second or last condition. Indeed, the family that contains all 0, 1-weight functions guarantees (1) and (3); and the family consisting of the single weight function w defined by $w(e_i) = 2^i$ (where pairs of vertices/edges are ordered in an arbitrary fixed order) guarantees (1) and (2). Prior to the work [19], no nontrivial bounds were known, and they almost managed to satisfy all criteria when restricted to bipartite graphs. Specifically, they construct such a family $\mathcal{F}_n$ for bipartite graphs where the polynomial bound on the size of $\mathcal{F}_n$ (the second condition) and the range of the weights (the third condition) were relaxed from polynomial to quasi-polynomial $2^{\log n^{O(1)}}$.

To construct $\mathcal{F}_n$, order the set $\{e_1, e_2, \ldots, e_{\binom{n}{2}}\}$ of potential edges in an n-vertex graph arbitrarily. It is not hard to see that the weight function w defined by $w(e_i) = 2^i$ is isolating. However, w does not have (quasi)polynomial values and it turns out that it is hard to find such a weight-function immediately. A key idea of [19] is to build the weight function in rounds, and at each step consider a much easier problem. In each round, a weight function is selected from a family $\mathcal{W}$ of $O(n^6)$ many weight functions defined by

$$\mathcal{W} = \left\{w^{(\ell)} \mid \ell = 1, \ldots, 2n^6\right\}, \quad \text{where } w^{(\ell)}(e_i) = 2^i \bmod \ell.$$

Theorem 2.1 ([19]). *For every n-vertex bipartite graph with at least one perfect matching, there is a selection of weight functions $w_1, w_2, \ldots, w_k \in \mathcal{W}$ with $k = O(\log n)$ such that there is a unique perfect matching M minimizing the lexicographic order of $(w_1(M), \ldots, w_k(M))$.*

Note that the above theorem implies the promised family of weight functions. Indeed, putting enough weight on the first weight function compared to the second and so on reduces lexicographic minimization to that of minimizing the total weight of a matching. So the family $\mathcal{F}_n = \{\sum_{i=1}^{k} n^{2(k-i)} w_i \mid w_1, \ldots, w_k \in \mathcal{W}\}$ satisfies the three criteria (1)–(3) where the polynomial bounds are replaced by the quasipolynomial bound $n^{O(\log n)}$.

Now, to prove Theorem 2.1, we first give a sufficient condition for a weight function to be isolating. We then explain how [19] ingeniously selects the weights w_i in rounds using girth (the length of the shortest cycle) as a progress measure.

Circulations: a sufficient condition for a weight function to be isolating. If a weight function w is *not* isolating, then there exist two minimum-weight perfect matchings, and their symmetric difference consists of alternating cycles. In each such cycle, the total weight of edges from the first matching must be equal to the total weight of edges from the second matching (as otherwise we could obtain another matching of lower weight). The difference between these two total weights is called the circulation of the cycle. Formally, the *circulation* of an even cycle C with respect to weight function w is defined by

$$\text{circulation}(C, w) = \left| \sum_{e \in M_1} w(e) - \sum_{e \in M_2} w(e) \right|,$$

where M_1 is the matching obtained by taking every second edge of C and M_2 is the matching $C \setminus M_1$. By the above, we can observe the following.

Observation 2.2. *If all cycles have nonzero circulation, then w is isolating.*

Weight function in $\mathcal{W}$ doubles the girth. As aforementioned, a key idea of [19] for proving Theorem 2.1 is to select the weight function in rounds, and at each step consider a much easier problem. Specifically, instead of trying to show that there is a weight function $w \in \mathcal{W}$ that assigns a nonzero circulation to *all* cycles (potentially exponentially many), let us start by finding a weight function $w \in \mathcal{W}$ that assigns a nonzero circulation to "short" cycles.

Suppose the considered bipartite graph $G = (V, E)$ has no cycle of length at most k. We will assign nonzero circulation to all cycles of length at most $2k$. To this end, we start by bounding the number of such cycles. For ease of notation, we bound the number of 8-cycles when G has no cycles of length 4. It will then be clear how to adapt the argument to the general case. The bound on the number of 8-cycles follows from a nice encoding argument. We associate a signature (a, b, c, d) with each 8-cycle, where a is the first vertex, b is the third vertex, c is the fifth vertex, and d is the seventh vertex when we traverse the cycle starting from one of the vertices. Now, if two 8-cycles have the same signature then it is easy to see that would yield a cycle of length 4. Hence, if G has no cycles of length 4, the number of 8-cycles is at most the number of signatures which is at most n^4. We can generalize this argument to get the following:

Lemma 2.3. *A graph with no cycles of length at most k has at most $2n^4$ cycles of length at most $2k$.*

We have thus bounded the number of cycles that we consider in this round by a polynomial in n. An easy argument shows that we can always find a weight function $w \in \mathcal{W}$ that assigns nonzero circulation to such a relatively small set of cycles.

Lemma 2.4. *Let $\mathcal{C}$ be a set of at most $2n^4$ even cycles, then there is a weight function in $\mathcal{W}$ that assigns nonzero circulation for every cycle in $\mathcal{C}$.*

Proof. Let w be the weight function defined by $w(e_i) = 2^i$. As already observed, w assigns nonzero circulation to all cycles. In particular, we have $\text{circulation}(C, w) \neq 0$ for every $C \in \mathcal{C}$. Furthermore, using that the w-weight of any edge is at most $2^{\#\text{edges}} \leq 2^{\binom{n}{2}}$, we also have $\text{circulation}(C, w) \leq 2^{n^2}$. We thus have

$$\prod_{C \in \mathcal{C}} \text{circulation}(C, w) \neq 0 \quad \text{and} \quad \prod_{C \in \mathcal{C}} \text{circulation}(C, w) \leq (2^{n^2})^{|\mathcal{C}|} \leq 2^{2n^6}.$$

That there is a weight function in $\mathcal{W}$ that assigns nonzero circulation to all cycles in $\mathcal{C}$, i.e., that there is an $\ell \in \{1, 2, \ldots, 2n^4\}$ such that

$$\prod_{C \in \mathcal{C}} \text{circulation}(C, w) \neq 0 \bmod \ell$$

now follows from the fact that the least common multiple of $1, 2, \ldots, 2n^6$ is greater than 2^{2n^6}, so not all these numbers can divide $\prod_{C \in \mathcal{C}} \text{circulation}(C, w)$. ∎

Given a bipartite graph $G = (V, E)$ with no cycles of length at most k, we can thus find a weight function $w \in \mathcal{W}$ such that all cycles for length at most $2k$ has nonzero circulation. The following lemma ensures that the girth is a good progress measure for bipartite graphs. It shows that the subgraph $H = (V, E')$, where $E' \subseteq E$ is the union of perfect matchings minimizing w, has no cycle of length at most $2k$.

Lemma 2.5. *Consider the subgraph $H = (V, E')$ of G where $E' \subseteq E$ is the union of perfect matchings that minimize a weight function w. Then H does not contain any cycle C with* $\text{circulation}(C, w) \neq 0$.

Before giving the proof of this lemma, let us explain how it implies Theorem 2.1. Consider *any* n-vertex bipartite graph $G = (V, E)$. There are at most $n^4 \leq 2n^4$ cycles of length at most 4. So there is a weight function $w_1 \in \mathcal{W}$ that assigns nonzero circulation to these cycles by Lemma 2.4. Let $G_1 = (V, E_1)$ be the subgraph where $E_1 \subseteq E$ is the union of perfect matchings that minimize w_1. Then the above lemma says G_1 that has no cycles of length at most 4. Lemma 2.3 then says that G_1 has at most $2n^4$ cycles of length at most 8. This allows us to repeat the same argument to show that there is a weight function $w_2 \in \mathcal{W}$ such that the graph $G_2 = (V, E_2)$, where $E_2 \subseteq E_1$ is the union of perfect matchings of G_1 that minimize w_2, has no cycles of length 8. Note that G_2 contains those perfect matchings M that minimize the lexicographic order of $(w_1(M), w_2(M))$. By repeating this $k = \lceil \log_2 n \rceil$ steps, we select weight functions $w_1, \ldots, w_k$ such that the graph G_k that contains those perfect matchings M that minimize the lexicographic order of $(w_1(M), \ldots, w_k(M))$ has no cycles of length at most n. In other words, G_k has no cycles and there is therefore a unique perfect matching in G_k.

Proof of Lemma 2.5. The argument uses that the *bipartite* perfect matching polytope has the following simple characterization:

$$\begin{aligned} x(\delta(v)) &= 1 \quad \text{for } v \in V, \\ x_e &\geq 0 \quad \text{for } e \in E. \end{aligned}$$

Here, we used $\delta(v)$ to denote the set of edges incident to vertex v and $x(F) = \sum_{e \in F} x_e$ for a subset $F \subseteq E$ of edges. Let x^* be the convex combination of all perfect matchings that minimize w. By definition, x^* is in the perfect matching polytope, and its support equals E', i.e., $x^*_e > 0$ for every $e \in E'$.

Now suppose toward contradiction that E' contains a cycle C with circulation$(C, w) \neq 0$. In other words, if we let M_1 and M_2 be the unique partitioning of C's edges into two matchings, then

$$\sum_{e \in M_1} w(e) \neq \sum_{e \in M_2} w(e).$$

Suppose $\sum_{e \in M_1} w(e) < \sum_{e \in M_2} w(e)$ (the other case is symmetric). Then

$$y_e = \begin{cases} x^*_e + \varepsilon & \text{if } e \in M_1, \\ x^*_e - \varepsilon & \text{if } e \in M_2, \\ 0 & \text{otherwise} \end{cases}$$

is a feasible solution to the bipartite perfect matching polytope for a small enough $\varepsilon > 0$. Indeed, every vertex in C is incident to exactly one edge in M_1 and one edge in M_2 and so the degree constraints are maintained; we also have $y \geq 0$ by selecting $\varepsilon > 0$ small enough since $x^*_e > 0$ for every $e \in E'$. We further have that y has lower cost than x^* because $\sum_{e \in M_1} w(e) < \sum_{e \in M_2} w(e)$. A contradiction since x^* is a convex combination of perfect matchings minimizing w. ∎

The proof of the above lemma completes the proof of Theorem 2.1 and the overview of the framework of Fenner, Gurjar, and Thierauf. A careful reader may have noted that the only place where we used that the graph was bipartite was in the proof of Lemma 2.5. In that proof, we used the simple structure of the bipartite perfect matching polytope. In the next subsection, we explain why replacing this lemma for general graphs is nontrivial and give a brief outline of the approach in [46].

2.3. Polyhedral techniques for general graphs

Parts of this section is adapted from [46]. To extend the argument to general graphs, it will be useful to look at the method explained in the previous section from a polyhedral perspective. We begin from the set of all perfect matchings, of which we take the convex hull: the perfect matching polytope. After applying the first weight function $w_1 \in \mathcal{W}$, we want to consider only those perfect matchings which minimize the weight; this is exactly the definition of a face of the polytope (e.g., face $F[1]$ in Figure 1). Recall that the goal is to show that for a small k, there exists $w_1, w_2, \ldots, w_k \in \mathcal{W}$ so that the lexicographic minimizer of $(w_1, w_2, \ldots, w_k)$ is unique. Now we need to have a smart progress measure to show that there is such a choice of $w_1, \ldots, w_k$ for a small k. It is in this part that a good polyhedral understanding plays a key role. Specifically, if we consider the set of solutions that minimize w_1 then that defines a face F_1 of the polytope (convex hull of solutions) and, then minimizing w_2 defines a subface F_2 of F_1, and so on. The goal is to show that there is a selection of

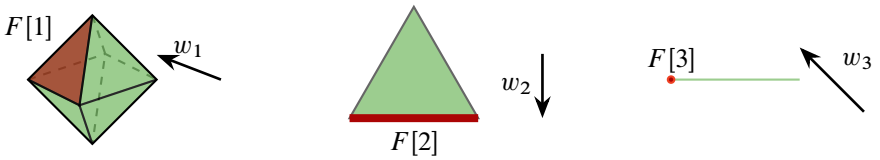

FIGURE 1
Polyhedral perspective on the construction of isolating weight function.

$(w_1, \ldots, w_k)$ so that the final face is of dimension 0, i.e., has a unique solution. In Figure 1, this happens after the choice of three weight functions.

In the bipartite case, any face is characterized by just taking a subset of edges (i.e., making certain constraints $x_e \geq 0$ tight). This simple structure of the bipartite perfect matching polytope was crucial in the proof of Lemma 2.5 and allowed [19] to have the girth as a simple progress measure as we described in the previous subsection.

In the nonbipartite case, the description of the perfect matching polytope is more involved. Namely, in addition to the degree constraints, Edmonds characterization [15] of the perfect matching polytope of a general graph $G = (V, E)$ also involves exponentially many odd-set constraints:

$$\begin{aligned} x(\delta(v)) &= 1 \quad \text{for } v \in V, \\ x(\delta(S)) &\geq 1 \quad \text{for } S \subseteq V \text{ with } |S| \text{ odd}, \\ x_e &\geq 0 \quad \text{for } e \in E. \end{aligned}$$

Recall that $\delta(v)$ denotes the set of edges incident to v and $x(F) = \sum_{e \in F} x_e$ for a subset $F \subseteq E$ of edges. Moreover, for a subset $S \subseteq V$ of vertices, we use $\delta(S)$ to denote the edges that cross the cut defined by S, i.e., the edges with exactly one endpoint in S.

Thus for general graphs, a face is not only defined by making certain constraints $x_e \geq 0$ tight but may also include tight *odd-set constraints* $x(\delta(S)) \geq 1$. This complicates our task, as depicted in Figure 2 (the same example was first given by [19] and then by [46] to demonstrate the difficulty of the general-graph case). Now a face is described by not only a subset of edges, but also a family of tight odd-set constraints. Thus we can no longer guarantee that any cycle whose circulation has been made nonzero will disappear from the support of the new face, i.e., the set of edges that appear in at least one perfect matching in this face (as, e.g., illustrated in Figure 2). Our idea of what it means to remove a cycle thus needs to be refined as well as the measure of progress we use to prove that a single matching is isolated after a few rounds.

Unfortunately, the current progress measure for general graphs is significantly more complex than for bipartite graphs and beyond the scope of this overview. Instead, we mention two crucial properties that allow us to deal with these odd-set constraints.

Decomposition into two sub-instances. The first property is easy to see: once we fix the single edge e in the matching which crosses a tight set S, the instance breaks up into two *independent* subinstances. That is, every perfect matching which contains e is the union of:

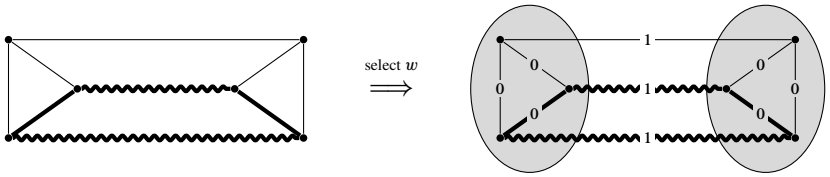

FIGURE 2

An illustration of the difficulty for general graphs. In trying to remove the bold cycle, we select a weight function w such that the circulation of the bold cycle is $|1 - 0 + 1 - 0| \neq 0$. By minimizing over w, we obtain a new, smaller subface—the convex hull of perfect matchings of weight 1—but every edge of the cycle is still present in one of these matchings. The vertex sets drawn in gray represent the new tight odd-set constraints that describe the new face (indeed, for a matching to have weight 1, it must take only one edge from the boundary of a gray set).

the edge e, a perfect matching on the vertex set S (ignoring the S-endpoint of e), and a perfect matching on the vertex set $V \setminus S$ (ignoring the other endpoint of e). Intuitively, this allows us to employ a divide-and-conquer strategy: to isolate a matching in the entire graph, we will take care of both subinstances and of the cut separating them.

One can see that the divide-and-conquer strategy would lead to a low depth recursion (and the selection of few weight functions), assuming that we could always find a balanced tight odd-set constraint. That is a tight odd-set constraint $x(\delta(S)) = 1$ with $|S| = \Omega(n)$ and $|V \setminus S| = \Omega(n)$. However, in general there is no reason to expect that we would always be able to find such a balanced cut and we combine the above strategy with a well-known structural property of the perfect matching polytope called laminarity.

Laminarity. The second crucial property that we utilize is that the family of odd-set constraints tight for a face exhibits good structural properties. Namely, it is known that at most $2n - 1$ odd-set constraints are enough to describe any face and they have a very nice structural property called laminarity. A *family of sets is laminar* if any two sets in the family are either disjoint or one is a subset of the other. While this structural property is not very hard to prove, we omit it here as we will prove and exploit a very similar polyhedral fact in Section 3.4, where we discuss the asymmetric traveling salesman problem. The structure enables a scheme where we use the laminar family to define our progress measure and make progress in a bottom-up fashion. Combining this bottom-up approach with the techniques of [19] allows us to extend Theorem 2.1 to general graphs (albeit by increasing the number of weight functions by a logarithmic factor).

Theorem 2.6 ([46]). *For every n-vertex graph with at least one perfect matching, there is a selection of weight functions $w_1, w_2, \ldots, w_k \in \mathcal{W}$ with $k = O(\log^2 n)$ such that there is a unique perfect matching M minimizing the lexicographic order of $(w_1(M), \ldots, w_k(M))$.*

2.4. Future directions

The results described *almost* (instead of completely) derandomize [36] because of the quasipolynomial ($n^{(\log n)^{O(1)}}$) instead of polynomial bounds. To further develop these techniques to show that the perfect matching problem (even for bipartite graphs) has an efficient deterministic parallel algorithm remains a prominent question (see [3, 40] for recent progress in the special case of planar graphs where the decision and counting versions were previously known). This would most likely require an alternative approach as selecting logarithmically many weight functions in rounds naturally lead to quasipolynomial bounds.

The randomized algorithm in [36] is very versatile, and it can be used to solve several natural variants of the perfect matching problem, including the aforementioned k-extendable matching problem and the exact matching problem. Any progress on efficient (even sequential) deterministic algorithms for these problems would be very interesting. Indeed, no nontrivial results are known for general graphs, and there has only been progress on the exact matching problem in very special cases [21, 30, 54].

To make further progress on these questions, we believe that it is important to develop a good polyhedral understanding of the exact matching and the k-extendable matching problems. Indeed, all the work following the approach of [19] relied on our excellent polyhedral understanding of those problems [22, 23, 46]. A concrete step is to *determine the extension complexity*[3] of the exact matching problem and the k-extendable matching problem on bipartite graphs. We remark that it is important that we restrict ourselves to bipartite graphs as already the perfect matching problem for general graphs has exponential extension complexity [39]. This is in contrast to the perfect matching polytope for bipartite graphs, which has linear (in the number of edges) extension complexity. We therefore believe that the resolution of the above question would make significant progress towards understanding the additional difficulty posed by these variants.

3. THE (ASYMMETRIC) TRAVELING SALESMAN PROBLEM

The traveling salesman problem, of finding the shortest tour that visits n given cities, is one of the best-known optimization problems. It is a cornerstone NP-hard optimization problem that has played a central role in devising and evaluating techniques for overcoming NP-hardness (see, e.g., the books [4, 11, 32]). The difference compared to problems in P is that we do not expect NP-hard optimization problems to admit efficient algorithms that are guaranteed to find optimal solutions (unless P = NP). Therefore, when confronted with such an optimization problem, we need to relax our requirements on, e.g., optimality or reliability (that the algorithm is guaranteed to work on every input). If we relax reliability, we obtain heuristics where our goal is to devise algorithms with good performance on typical instances. If we relax optimality, we obtain approximation algorithms. *Approximation algo-*

3 An extension of a polyhedron P is a polyhedron Q such that P is the image of Q under a linear map. The extension complexity of P is the minimum number of facets (inequalities) of any extension of P.

rithms are efficient (i.e., polynomial-time) algorithms that are guaranteed to find a solution that is close—within a factor called the approximation guarantee—in value to an optimal solution. The study of approximation algorithms gives a mathematically rigorous way for (i) having a more fine-grained understanding of NP-hard optimization problems (some are easier to approximate than others) and (ii) in evaluating different algorithmic techniques. It is also a very intuitive notion as, in many situations, it is sufficient to find a solution that is close to optimal but not necessarily optimal.

As for the matching problem, polyhedral techniques have been central in the development of good algorithms for the traveling salesman problem. The most studied linear programming relaxation, which is often referred to as the Held–Karp relaxation (because of the seminal paper [24])[4] or the subtour elimination relaxation (because of the structure of the formulation), has puzzled researcher for decades. Indeed, a longstanding conjecture states that the Held–Karp relaxation approximates the value of any metric traveling salesman problem instance within a factor $4/3$ when distances are symmetric (the distance $\mathrm{dist}(i, j)$ of going from city i to city j equals the distance $\mathrm{dist}(j, i)$). Experimental evidence for the conjecture is given in [6]. A similar situation also holds in the asymmetric setting (when $\mathrm{dist}(i, j)$ does not necessarily equal $\mathrm{dist}(j, i)$) albeit with a much larger gap. Closing these gaps are considered major open problems in theoretical computer science.

Recently there have been significant advances on these questions. For the asymmetric traveling salesman problem, we obtained the first constant-factor approximation algorithm in [48]; see also the work by Traub and Vygen [50] who simplified the approach and obtained a better approximation guarantee. For the symmetric version, Karlin, Klein, and Oveis Gharan [28] obtained the first improvements on the classic $3/2$-approximation algorithm by Christofides [10] and Serdyukov [43]. We focus here on algorithmic approaches for the *asymmetric* traveling salesman problem. However, we briefly comment on the breakthrough work by [28] in Section 3.3, where we discuss a similar algorithm for the asymmetric problem.

3.1. Designing approximation algorithms for ATSP

An instance of the asymmetric traveling salesman problem (ATSP) is a tuple (V, dist) where V is the set of vertices/cities and dist gives the distances between vertices. In other words, an instance is a complete directed graph with edge-weights given by dist. A tour is a cycle that visits every vertex exactly once, i.e., a Hamiltonian cycle. The goal is to find a tour $F \subseteq E$ of minimum total distance $\mathrm{dist}(F) = \sum_{e \in F} \mathrm{dist}(e)$.

Without any assumptions on the distances, a simple reduction from the problem of deciding whether a graph is Hamiltonian shows that it is NP-hard to approximate the shortest tour to within any factor. Therefore it is common to assume that the distances satisfy the triangle inequality: the distance $\mathrm{dist}(i, k)$ from i to k is no longer than the distance $\mathrm{dist}(i, j)$ from i to j plus the distance $\mathrm{dist}(j, k)$ from j to k. All results that we mention refer to this

4 The relaxation in fact dates back to the earlier paper by Dantzig et al. [14] who solved a special 49-instance of the problem.

setting and we will assume that the distances satisfy the triangle inequality from now on (without explicitly stating it). One can see that this assumption is equivalent to allowing the tour to visit cities more than once; see the remark after Theorem 3.2. This viewpoint is very convenient when designing and analyzing algorithms for ATSP.

When designing an approximation algorithm, the task is to devise a polynomial-time algorithm that, for *any* instance (V, dist), outputs a tour whose length is guaranteed to be at most a factor c worse than the length of an optimal tour. The factor $c \geq 1$ is often referred to as the approximation ratio or the approximation guarantee. In the analysis, we thus face the problem of upper bounding our cost with the cost of a complex optimal solution that is even NP-hard to compute. A common technique to overcome this difficulty is to analyze the algorithm compared to a "simpler" lower bound that we can compute in polynomial time. Such a good lower bound then often also helps in the design of the approximation algorithms. In Sections 3.2 and 3.3, we see two complementary approaches that are based on two natural lower bounds: minimum cost cycle cover and minimum spanning tree, respectively. In Section 3.4, we then give an overview of the approach for achieving a constant-factor approximation algorithm and explain how the polyhedral structure of the Held–Karp relaxation allows us to reduce the general problem to very structured distances.

3.2. The repeated cycle cover approach

In this section, we explain the elegant "repeated cycle cover" approach by Frieze, Galbiati, and Maffioli [20]. A *cycle cover* of a directed graph is a subset C of the edges so that each vertex has in-degree and out-degree equal to one. In other words, C consists of a collection of cycles so that each vertex is in exactly one. A minimum cost cycle cover C in an edge-weighted graph is a cycle cover of minimum total distance $\text{dist}(C) = \sum_{e \in C} \text{dist}(e)$. It is not hard to see that we can compute a minimum cost cycle cover in polynomial time: it reduces to that of calculating a minimum cost perfect matching in a bipartite graph. Furthermore, we have the following observation, which follows by noting that an optimal tour is a cycle cover consisting of a single cycle, so the *minimum* cost cycle cover can only have a smaller cost.

Observation 3.1. *A minimum cost cycle cover costs at most the length of an optimal tour.*

The algorithm in [20] now ensures connectivity by the following procedure that repeatedly finds cycle covers:

(1) Find a minimum cost cycle cover.

(2) Select an arbitrarily proxy node for each cycle.

(3) Recursively solve the problem on proxies.

An illustration of the algorithm is given in the left part of Figure 3. First we find a cycle cover consisting of three cycles (depicted by solid edges). In each of these cycles, we select a proxy vertex (depicted in gray); and then in the next cycle cover instance, we find a minimum cost cycle cover on those proxy vertices a, c, and f (depicted by dashed edges). In this example,

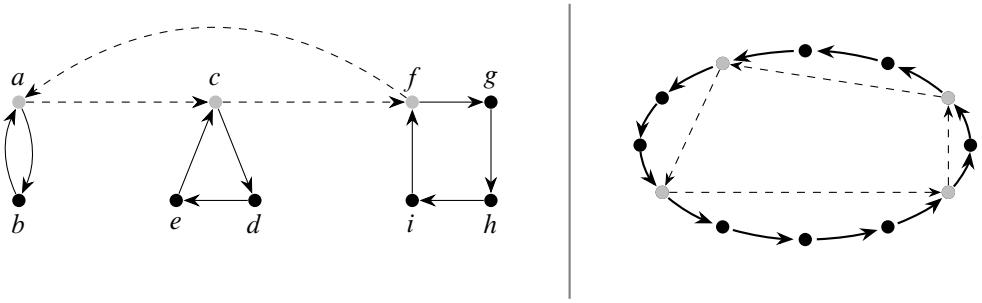

FIGURE 3
(Left) An illustration of the repeated cycle cover algorithm; (Right) Short-cutting does not increase the length of the tour by the triangle inequality.

we managed to connect the graph after only two iterations. In general, we have that each cycle cover at least halves the number of proxy vertices (they are exactly halved if each cycle in the cycle cover has length two). Thus, the algorithm selects at most $\log_2(n)$ cycle covers. We can furthermore upper bound the cost of each of these cycle covers by the length of an optimal tour. Indeed, let V_i be the set of proxy nodes when the ith cycle cover is found. Then, by Observation 3.1, the cost of the minimum cost cycle cover is at most the length of an optimal tour of V_i, which in turn is at most the length of an optimal tour that visits all vertices by the triangle inequality (see right part of Figure 3). The cost of the ith cycle cover is thus at most the length of an optimal tour. Combining the facts that we select at most $\log_2(n)$ cycle covers and each of them has cost at most the length of an optimal tour yields that the repeated cycle cover algorithm always finds a tour that is at most a factor $\log_2 n$ longer than an optimal tour.

Theorem 3.2 ([20]). *The repeated cycle cover algorithm is a* $\log_2(n)$*-approximation algorithm for ATSP.*

We remark that here we used the observation that finding a connected Eulerian (the in-degree of each vertex equals its out-degree) edge set is equivalent to finding a tour when distances satisfy the triangle inequality. Indeed, if the graph is connected and the in-degree of every vertex equals its out-degree, then we can efficiently find a so-called Eulerian tour that walks each edge exactly once. In the solution found in the left part of Figure 3, an Eulerian tour is $a-b-a-c-d-e-c-f-g-h-i-f-a$. Now any such tour can be short cut into a tour that visits each vertex exactly once by simply traversing the vertices in the same order as the Eulerian tour but not revisiting vertices (in the example this gives $a-b-c-d-e-f-g-h-i$). By the triangle inequality, this does not increase the length of the tour. Therefore, in the subsequent we will slightly abuse notation and refer to a connected Eulerian edge set as a tour.

The factor $\log_2(n)$ appears at first sight rather pessimistic. For the repeated cycle cover algorithm to return a tour with that approximation guarantee, basically all the found cycles must have length two and all cycle covers have a cost that equals the length of an opti-

mal tour. However, it turns out that such worst-case instances do exist, and it is nontrivial to obtain improved guarantees. The papers [8,17,27] refine the approach to improve the constant in front of $\log_2(n)$ but the first asymptotic improvement on the logarithmic approximation guarantee was obtained by using another natural lower bound that we describe next.

3.3. The spanning tree approach

Another lower bound on the length of an optimal tour is the cost of a minimum cost spanning tree (where we forget the orientation of the edges). This is a lower bound since if we take a tour and remove a single edge, we get a tree whose cost is upper bounded by the length of the tour (minus the length of the dropped edge).

Using the spanning tree as a lower bound naturally leads to a complementary algorithm to the repeated cycle cover approach. Instead of ensuring that the graph is Eulerian and iteratively making it connected, we first connect the graph and then add edges to make it Eulerian:

(1) Find a minimum cost spanning tree T.

(2) Find a min-weight set of edges F so that $T \cup F$ is Eulerian.

For the symmetric case, this is the famous algorithm of Christofides [10] and Serdyukov [43]. As aforementioned, the cost of the tree T is at most the length of an optimal tour. Furthermore, for the symmetric case, one can show that the cost of the second step is at most half the optimum no matter the selected tree T. This yields the classic approximation guarantee of $3/2$ for the symmetric traveling salesman problem. In contrast, there is no hope to get a good upper bound on the cost of the second step in the asymmetric case if we start with an arbitrary spanning tree. Figure 4 depicts such an example. A minimum spanning tree is depicted by solid edges on the left. Extending that tree to an Eulerian graph must add four dotted edges of large cost as depicted in the middle. However, an optimal tour (depicted on the right) only uses one of these expensive (dotted) edges. By selecting the cost of the dotted edges to be large enough and increasing the number of vertices in the middle layer, we obtain an instance in which the tour obtained by extending a minimum spanning tree is a linear (in the number of vertices) factor more expensive than the optimal tour.

To overcome this difficulty, we use the Held–Karp relaxation which provides a stronger lower bound that can be seen to generalize both the minimum cost cycle cover and minimum weight spanning tree lower bounds.

Held–Karp relaxation. The Held–Karp relaxation has a variable $x_{(u,v)} \geq 0$ for every ordered pair of vertices (u, v). The intended solution is that $x_{(u,v)}$ should indicate whether the tour goes from u to v. We let E be the set of all such ordered pairs/edges. For a subset $S \subseteq V$ of vertices, we use $\delta^+(S) = \{(u, v) \in E \mid u \in S, v \notin S\}$ and $\delta^-(S) = \{(u, v) \in E \mid u \notin S, v \in S\}$ to denote the outgoing and incoming edges to S, respectively. We also let $\delta(S) = \delta^+(S) \cup \delta^-(S)$ be the "undirected" cut and for a subset

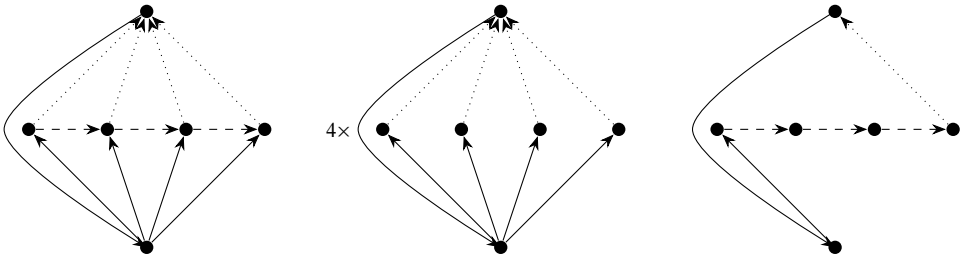

FIGURE 4

An example where correcting the degrees of a minimum spanning tree is expensive. Solid, dashed, and dotted edges have distances 1,2 and $M \gg 2$, respectively. The distances of the remaining pairs is the shortest path distance in this graph. The tour obtained by correcting a minimum spanning tree is depicted in center (using four expensive dotted edges) and an optimal tour is depicted on the right (using one expensive edge).

$F \subseteq E$ we let $x(F) = \sum_{e \in F} x_e$. With this notation, the relaxation is now defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} x_e \cdot \text{dist}(e) \\ \text{subject to} \quad & x(\delta^+(v)) = x(\delta^-(v)) = 1 \qquad v \in V, \\ & x(\delta(S)) \geq 2 \qquad \emptyset \neq S \subset V, \\ & x \geq 0. \end{aligned}$$

The first set of constraints says that each vertex should be visited once, so the in-degree and the out-degree should equal one for each vertex. The second set of constraints enforces that the solution is connected and they are sometimes referred to as subtour elimination constraints. We remark that although the Held–Karp relaxation has exponentially many constraints, it is well known that we can solve it in polynomial time either by using the ellipsoid method with a separation oracle or by formulating an equivalent compact (polynomial size) linear program.

Thin spanning trees. Consider a solution x to the Held–Karp relaxation and let $z_{\{u,v\}} = x_{(u,v)} + x_{(v,u)}$ be the solution where we dropped the orientation of edges. It is well known that $(n-1)/n \cdot z$ can be written as a convex combination of spanning trees (this can be seen by a simple calculation using Edmonds' characterization of the spanning tree polytope). In other words, there is a distribution μ over spanning trees satisfying

$$\Pr_{T \sim \mu}[e \in T] = \frac{n-1}{n} \cdot z_e \quad \text{for every } e \in E.$$

The selection of μ satisfying the above equality for every edge is not unique, and among all such distributions, [5] proposed to select the one of *maximum entropy*. In other words, instead of selecting a minimum spanning tree in the first step of the algorithm, we sample a tree T from μ. This randomized version of Christofides/Serdyukov algorithm has been very influential both for the symmetric and the asymmetric version. Although we focus on ATSP,

let us mention that the recent major breakthrough [28] that presents the first improvement in more than four decades is a deep analysis of this algorithm. Their analysis heavily relies on properties of the maximum entropy distribution of spanning trees; specifically, that this distribution has very strong negative correlation properties (called strongly Rayleigh).

To analyze the randomized algorithm for ATSP, another influential contribution of [5] is the formulation of *thinness*: a clean sufficient condition for a tree T to have a low correction cost. A *spanning tree T is α-thin* with respect to a solution x to the Held–Karp relaxation if

$$\big|\{e \in T \mid e \text{ has exactly one endpoint in } S\}\big| \leq \alpha \cdot x\big(\delta(S)\big) \quad \text{for every } \emptyset \neq S \subsetneq V.$$

In words, the number of times the tree T crosses each cut is bounded by the (fractional) crossings of the linear program solution. Now using Hoffman's circulation theorem, they bound the correction cost of a thin tree leading to the following theorem:

Theorem 3.3 ([5]). *Given a spanning tree T that is α-thin with respect to an optimal solution to the Held–Karp relaxation, we can in polynomial time find a tour whose length is at most a factor $O(\alpha)$ longer than an optimal tour.*

Asadpour, Goemans, Madry, Oveis Gharan, and Saberi [5] then obtained their $O(\log n / \log\log n)$-approximation algorithm by showing that a tree sampled from the maximum entropy distribution is with high probability $O(\log n / \log\log n)$-thin (with respect to the Held–Karp solution). This analysis is tight in the following sense: it is known that there are instances so that, if we sample a tree from the maximum entropy distribution, the obtained tree is likely to be $\Omega(\log n / \log\log n)$-thin. However, it is conjectured that a $O(1)$-thin tree always exist. This has been proven for special graph classes, such as planar graphs and more generally bounded-genus graphs [37]. Major progress was achieved by Anari and Oveis Gharan [2], who showed that there always exists a $O(\log\log(n)^{O(1)})$-thin tree. The proof is highly nontrivial and it is not known to imply a polynomial-time algorithm. One key component in the proof is, e.g., a generalization of the celebrated proof of the Kadison–Singer problem. As we further discuss in Section 3.5, it remains an intriguing open question whether there always exists a $O(1)$-thin tree.

3.4. General distances are not that general: a constant factor approximation

In this section, we give a brief overview of the recent constant-factor approximation algorithm for ATSP that was given by [48] and then simplified and improved to an approximation guarantee of 22 by [50]. The algorithm is based on a series of reductions that harness strong structural properties from the Held–Karp relaxation. These reductions reduce the task to solving ATSP on highly specialized instances. For those instances, one can then adopt the techniques in [45] that solved similar special cases. Specifically, a key component is the constant-factor approximation algorithm for so-called node-weighted ATSP instances. We say that an ATSP is *node-weighted* if the distances dist is the shortest path metric of a directed graph $G = (V, E)$ with non-negative node-weights $\{y_v\}_{v \in V}$. That is, the weight of an edge

$(u, v) \in E$ in G is $y_u + y_v$, and the distance dist between a pair (a, b) of vertices is the shortest path from a to b in this graph. See the left part of Figure 5 for an example.

Theorem 3.4 ([45]). *Given a node-weighted ATSP instance, there is a polynomial-time algorithm that returns a tour whose length is at most a constant factor longer than the optimal value of the Held–Karp relaxation.*

The algorithm in [45] is an extension of the repeated cycle cover approach: it maintains an Eulerian subset of edges and iteratively adds new Eulerian sets of edges to connect the graph. However, the algorithm in [45] is more complex because the selection of new edges is done in a very careful and nontrivial manner as to not lose more than a constant-factor in the approximation guarantee.

The approach of [48] for general distances now performs a series of reductions to apply the techniques of [45]. While [50] made excellent progress in improving the simplicity and the guarantee of the algorithm, the complete algorithm remains rather complex and we refer the reader to conference version [47] for a longer overview of the approach. Here we focus on one key insight that allows us to focus on laminarly-weighted ATSP instances which generalizes node-weighted instances but still keep a similar structure. Interestingly, the techniques we use here are similar to what was used in Section 2.3 for the matching problem. Specifically, we will use that the optimal solution to the dual linear program can be selected to be a laminar family. Recall that a laminar family $\mathcal{L}$ of subsets is such that any two sets $A, B \in \mathcal{L}$ are either subsets of each other or disjoint (see right part of Figure 5). In order to simplify the dual, we use the fact that it is equivalent to finding a tour that visits every vertex at least once and to find a tour that visits every vertex exactly once (since we assume the triangle inequality). This allows us to drop the constraint that the in-degree and out-degree of a vertex are equal to 1. That is, we obtain an equivalent formulation of the Held–Karp relaxation by replacing $x(\delta^+(v)) = x(\delta^-(v)) = 1$ by $x(\delta^+(v)) = x(\delta^-(v))$ for every vertex $v \in V$. By associating variables $(\alpha_v)_{v \in V}$ and $(y_S)_{\emptyset \neq S \subset V}$ with the degree constraints and the subtour elimination constraints, respectively, we obtain the dual linear program:

$$\begin{aligned} \max \quad & \sum_{\emptyset \neq S \subset V} 2 \cdot y_S \\ \text{subject to} \quad & \sum_{S:(u,v) \in \delta(S)} y_S + \alpha_u - \alpha_v \leq \operatorname{dist}(u, v) \quad \text{for } u \neq v \in V, \\ & y \geq 0. \end{aligned}$$

Now a key property of the dual is the laminarity of optimal solutions, which is similar to the structure that we used for the perfect matching problem in Section 2.3.

Lemma 3.5. *There exists an optimal solution (α, y) to the dual such that the support of y is a laminar family of vertex subsets.*

Proof. This proof is taken from [48]. We show the existence of a optimal laminar solution using a standard uncrossing argument (see, e.g., [12] for an early application of this technique to the Held–Karp relaxation of the symmetric traveling salesman problem). Select (α, y) to be an optimal solution to the dual minimizing $\sum_S |S| y_S$. That is, among all dual solutions that maximize the dual objective $2\sum_S y_S$, we select one that minimizes $\sum_S |S| y_S$. We claim that the support $\mathcal{L} = \{S : y_S > 0\}$ is a laminar family. Suppose not, i.e., that there are sets $A, B \in \mathcal{L}$ such that $A \cap B$, $A \setminus B$, $B \setminus A \neq \emptyset$. Then we can obtain a new dual solution $(\alpha, \hat{y})$, where $\hat{y}$ is defined, for $\varepsilon = \min(y_A, y_B) > 0$, as

$$\hat{y}_S = \begin{cases} y_S - \varepsilon & \text{if } S = A \text{ or } S = B, \\ y_S + \varepsilon & \text{if } S = A \setminus B \text{ or } S = B \setminus A, \\ y_S & \text{otherwise.} \end{cases}$$

That $(\alpha, \hat{y})$ remains a feasible solution follows since $\hat{y}$ remains nonnegative (by the selection of ε) and since for any edge e we have $\mathbb{1}_{e\in\delta(A)} + \mathbb{1}_{e\in\delta(B)} \geq \mathbb{1}_{e\in\delta(A\setminus B)} + \mathbb{1}_{e\in\delta(B\setminus A)}$. Therefore $\sum_{S:e\in\delta(S)} \hat{y}_S \leq \sum_{S:e\in\delta(S)} y_S$ and so the constraint corresponding to edge e remains satisfied. Further, we clearly have $2\sum_S \hat{y}_S = 2\sum_S y_S$. In other words, $(\alpha, \hat{y})$ is an optimal dual solution. However,

$$\sum_S |S|(y_S - \hat{y}_S) = \big(|A| + |B| - |A \setminus B| - |B \setminus A|\big)\varepsilon > 0,$$

which contradicts that (α, y) was selected to be an optimal dual solution minimizing $\sum_S |S| y_S$. Therefore, there can be no such sets A and B in $\mathcal{L}$, hence it is a laminar family. ∎

We now use the laminar structure to argue that we can assume that our distances are very structured. Let x be an optimal primal solution and let (y, α) be an optimal dual solution with laminar support $\mathcal{L} = \{S \mid y_S > 0\}$. Consider the graph $G = (V, E)$ where the edge-set is the support of x: $E = \{e \mid x_e > 0\}$. Then by complementarity slackness, we have $\text{dist}(e) = \sum_{S:(u,v)\in\delta(S)} y_S + \alpha_u - \alpha_v$ for every edge $e \in E$. Now note that for any cycle $v_0 \to v_1 \to \cdots \to v_{k-1} \to v_k = v_0$ (and thus Eulerian subset of edges), we have that its distance is given by y:

$$\sum_{i=0}^{k} \text{dist}(v_i, v_{i+1}) = \sum_{i=0}^{k-1} \left(\sum_{S:(v_i,v_{i+1})\in\delta(S)} y_S + \alpha_{v_i} - \alpha_{v_{i+1}} \right) = \sum_{i=0}^{k-1} \left(\sum_{S:(v_i,v_{i+1})\in\delta(S)} y_S \right).$$

Hence, if we let the weight of an edge $e \in E$ in G be $\text{dist}'(e) = \sum_{S:e\in\delta(S)} y_S$, then a tour has the same length in the shortest path metric of G obtained by using edge-weights $\text{dist}(e)$ as in that obtained by using edge-weights $\text{dist}'(e)$. This allows one to argue that we can replace the original distances with the shortest path metric of $G = (V, E)$ where each edge $e \in E$ has weight $\text{dist}'(e)$ without loss of generality when designing approximation algorithms with respect to the Held–Karp relaxation. In other words, the distance from a vertex u to v is given by the shortest path in $G = (V, E)$ where the weight of each edge e is given by the total weight $\sum_{S\in\mathcal{L}:e\in\delta(S)} y_S$ of the sets it crosses in a laminar family. We refer to such instances as

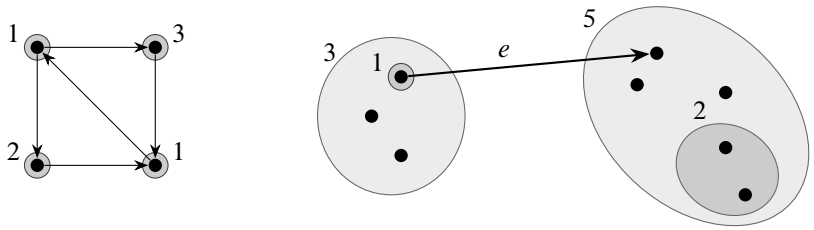

FIGURE 5

(Left) A node-weighted ATSP instance. The node-weights are depicted next to the vertices. The length of an edge is the sum of incident node weights and the length between two vertices is the shortest path distance in this graph. So the distance from the top-left vertex to the bottom-right vertex is $1 + 2 + 2 + 1 = 6$. (Right) An example of a laminarly-weighted ATSP instance. The sets of the laminar family are shown in gray, with their y-values written on their borders. We depict a single edge e that crosses three sets in the laminar family and has distance $1 + 3 + 5 = 9$.

laminarly-weighted (see right side of Figure 5). We remark that laminarly-weighted instances have some additional structure in [48] that we have simplified here.

Theorem 3.6. *Assume we have a polynomial-time algorithm that provides an α-approximation with respect to the Held–Karp relaxation for laminarly-weighted ATSP instances. Then there is a polynomial-time α-approximation algorithm with respect to the Held–Karp relaxation for the general ATSP problem.*

While laminarly-weighted instances have a similar structure to node-weighted instances, the approach in [48] performs several additional steps in order to use the techniques in [45]. In spite of the simplifications in [50], the overall algorithm remains complex and it is an interesting open problem to find a simple constant-factor approximation algorithm even for node-weighted instances. We discuss this and other open problems next.

3.5. Future directions

To give a tight analysis of the Held–Karp relaxation is a longstanding open problem (see, e.g., the two first open problems in the book [53] on approximation algorithms). We believe that the approximation guarantees given by this relaxation are $4/3$ and 2 for the symmetric and asymmetric traveling salesman problems, respectively. This would match the best known lower bounds [9].

The recent breakthrough in [28] opens up several promising directions for the symmetric version. Indeed, for the special case of unweighted shortest path metrics, the small improvement in [38] (using the same approach as in [28]) was quickly followed by more substantial improvements using different techniques [34,35,42]. It is interesting to know whether one can combine those techniques with the ones in [28]. Another exciting possibility is to exploit the laminar structure of the cost functions in the symmetric case as we did for ATSP.

The known constant-factor approximation algorithms for the asymmetric traveling salesman problem remain complex even in the case of node-weighted instances [45]. An important step for further progress is therefore to obtain a simpler constant-factor approximation algorithm. An intriguing possibility is to use the repeated-cycle cover approach. Instead of selecting a minimum cycle cover in each step, we would select one at random using the Held–Karp relaxation (similar to the randomized modification of the algorithm by Christofides/Serdyukov used in [28]). While the $\log_2(n)$ approximation guarantee is tight for the deterministic version, the approximation guarantee of the randomized version remains open.

In Section 3.3, we mentioned that the thin-tree conjecture implies a constant-factor approximation algorithm for ATSP. While we now know other methods for achieving a constant-factor approximation guarantee, the thin-tree conjecture is interesting in itself and it remains relevant for the asymmetric traveling salesman problem. In particular, it would imply a constant-factor approximation algorithm for the bottleneck version, where we wish to find a tour (Hamiltonian cycle) that minimizes the longest edge [1]. Finding a constant-factor approximation algorithm for bottleneck ATSP remains a challenging open problem and we believe that further progress is also likely to shed light on the thin-tree conjecture.

Finally, much of the work on the traveling salesman problem has been on analyzing the Held–Karp relaxation. But there are no real reasons to believe that we cannot achieve better algorithms using other lower bounds. In fact, the recent progress on the path traveling salesman problem is not with respect to a relaxation [49,51,55]. Moreover, we do not have any strong lower bounds on the relaxations obtained by using so called lift-and-project methods or hierarchies of relaxations [31,44].

ACKNOWLEDGMENTS

The author is very grateful to Jakub Tarnawski and László A. Végh. Jakub coauthored both [46] and [47] and László coauthored [47]. I also wish to thank my excellent mentors who have been central for my career, including my PhD advisor Monaldo Mastrolilli, postdoc advisor Johan Håstad, and my colleagues at EPFL.

FUNDING

This work was partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 335288–OptApprox) and the Swiss National Science Foundation project 200021-184656 "Randomness in Problem Instances and Randomized Algorithms."

REFERENCES

[1] H. An, R. D. Kleinberg, and D. B. Shmoys, Improving Christofides' algorithm for the *s-t* path TSP. *J. ACM* **62** (2015), 34:1–34:28.

[2] N. Anari and S. Oveis Gharan, Effective-resistance-reducing flows, spectrally thin trees, and asymmetric TSP. In *FOCS*, pp. 20–39, IEEE Computer Society, 2015.

[3] N. Anari and V. V. Vazirani, Planar graph perfect matching is in NC. *J. ACM* **67** (2020), 21:1–21:34.

[4] D. L. Applegate, R. E. Bixby, V. Chvatál, and W. J. Cook, *The traveling salesman problem: a computational study*. Princeton University Press, 2006.

[5] A. Asadpour, M. X. Goemans, A. Madry, S. Oveis Gharan, and A. Saberi, An $O(\log n / \log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. *Oper. Res.* **65** (2017), no. 4, 1043–1061.

[6] G. Benoit and S. Boyd, Finding the exact integrality gap for small traveling salesman problems. *Math. Oper. Res.* **33** (2008), no. 4, 921–931.

[7] S. J. Berkowitz, On computing the determinant in small parallel time using a small number of processors. *Inform. Process. Lett.* **18** (1984), no. 3, 147–150.

[8] M. Bläser, A new approximation algorithm for the asymmetric TSP with triangle inequality. *ACM Trans. Algorithms* **4** (2008), no. 4.

[9] M. Charikar, M. X. Goemans, and H. J. Karloff, On the integrality ratio for the asymmetric traveling salesman problem. *Math. Oper. Res.* **31** (2006), no. 2, 245–252.

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

[11] W. J. Cook, *In pursuit of the traveling salesman: mathematics at the limits of computation*, Princeton University Press, 2014.

[12] G. Cornuéjols, J. Fonlupt, and D. Naddef, The traveling salesman problem on a graph and some related integer polyhedra. *Math. Program.* **33** (1985), no. 1, 1–27.

[13] L. Csanky, Fast parallel inversion algorithm. *SIAM J. Comput.* **5** (1976), 618–623.

[14] G. Dantzig, R. Fulkerson, and S. Johnson, Solution of a large-scale traveling-salesman problem. *Oper. Res.* **2** (1954), 393–410.

[15] J. Edmonds, Maximum matching and a polyhedron with 0, 1 vertices. *J. Res. Natl. Bur. Stand.* **69** (1965), 125–130.

[16] J. Edmonds, Paths, trees, and flowers. *Canad. J. Math.* **17** (1965), 449–467.

[17] U. Feige and M. Singh, Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In *APPROX*, pp. 104–118, Springer, 2007.

[18] S. A. Fenner, R. Gurjar, and T. Thierauf, Bipartite perfect matching is in quasi-NC. In *STOC*, pp. 754–763, ACM, 2016.

[19] S. A. Fenner, R. Gurjar, and T. Thierauf, Bipartite perfect matching is in quasi-NC. *SIAM J. Comput.* **50** (2021), no. 3.

[20] A. M. Frieze, G. Galbiati, and F. Maffioli, On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks* **12** (1982), no. 1, 23–39.

[21] R. Gurjar, A. Korwar, J. Messner, and T. Thierauf, Exact perfect matching in complete graphs. *ACM Trans. Comput. Theory* **9** (2017), 8:1–8:20.

[22] R. Gurjar and T. Thierauf, Linear matroid intersection is in quasi-NC. *Comput. Complexity* **29** (2020), no. 2, 9.

[23] R. Gurjar, T. Thierauf, and N. K. Vishnoi, Isolating a vertex via lattices: Polytopes with totally unimodular faces. *SIAM J. Comput.* **50** (2021), no. 2, 636–661.

[24] M. Held and R. M. Karp, The traveling-salesman problem and minimum spanning trees. *Oper. Res.* **18** (1970), 1138–1162.

[25] R. Impagliazzo and A. Wigderson, $P = BPP$ if E requires exponential circuits: derandomizing the XOR lemma. In *STOC*, pp. 220–229, ACM, 1997.

[26] Jacobi's bound. https://www.lix.polytechnique.fr/~ollivier/JACOBI/jacobiEngl.htm, accessed: 2021-10-25.

[27] H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko, Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM* **52** (2005), no. 4, 602–626.

[28] A. R. Karlin, N. Klein, and S. Oveis Gharan, A (slightly) improved approximation algorithm for metric TSP. In *STOC*, pp. 32–45, ACM, 2021.

[29] R. M. Karp, E. Upfal, and A. Wigderson, Constructing a perfect matching is in random NC. *Combinatorica* **6** (1986), no. 1, 35–48.

[30] A. V. Karzanov, Maximum matching of given weight in complete and complete bipartite graphs. *Cybernetics* **23** (1987), no. 1, 8–13.

[31] J. B. Lasserre, An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM J. Control Optim.* **12** (2002), no. 3, 756–769.

[32] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The traveling salesman problem: a guided tour of combinatorial optimization*, Wiley, 1991.

[33] L. Lovász, On determinants, matchings, and random algorithms. In *FCT*, pp. 565–574, Akademie-Verlag, Berlin, 1979.

[34] T. Mömke and O. Svensson, Removing and adding edges for the traveling salesman problem. *J. ACM* **63** (2016), 2:1–2:28.

[35] M. Mucha, 13/9-approximation for graphic TSP. *Theory Comput. Syst.* **55** (2014), no. 4, 640–657.

[36] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, Matching is as easy as matrix inversion. *Combinatorica* **7** (1987), no. 1, 105–113.

[37] S. Oveis Gharan and A. Saberi, The asymmetric traveling salesman problem on graphs with bounded genus. In *SODA*, pp. 967–975, 2011.

[38] S. Oveis Gharan, A. Saberi, and M. Singh, A randomized rounding approach to the traveling salesman problem. In *FOCS*, pp. 550–559, IEEE Computer Society, 2011.

[39] T. Rothvoss, The matching polytope has exponential extension complexity. *J. ACM* **64** (2017), 41:1–41:19.

[40] P. Sankowski, NC algorithms for weighted planar perfect matching and related problems. In *ICALP*, pp. 97:1–97:16, LIPIcs. Leibniz Int. Proc. Inform. 107, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[41] A. Schrijver, *Combinatorial optimization – polyhedra and efficiency*. Springer, 2003.

[42] A. Sebö and 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** (2014), no. 5, 597–629.

[43] A. I. Serdyukov, O nekotorykh ekstremal'nykh obkhodakh v grafakh. *Upr. Sist.* **19** (1978), 76–79.

[44] H. D. Sherali and W. P. Adams, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.* **3** (1990), no. 3, 411–430.

[45] O. Svensson, Approximating ATSP by relaxing connectivity. In *FOCS*, pp. 1–19, IEEE Computer Society, 2015.

[46] O. Svensson and J. Tarnawski, The matching problem in general graphs is in quasi-NC. In *FOCS*, pp. 696–707, IEEE Computer Society, 2017.

[47] O. Svensson, J. Tarnawski, and L. A. Végh, A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In *STOC*, pp. 204–213, ACM, 2018.

[48] O. Svensson, J. Tarnawski, and L. A. Végh, A constant-factor approximation algorithm for the asymmetric traveling salesman problem. *J. ACM* **67** (2020), 37:1–37:53.

[49] V. Traub and J. Vygen, Approaching 3/2 for the s-t-path TSP. *J. ACM* **66** (2019), 14:1–14:17.

[50] V. Traub and J. Vygen, An improved approximation algorithm for ATSP. In *STOC*, pp. 1–13, ACM, 2020.

[51] V. Traub, J. Vygen, and R. Zenklusen, Reducing path TSP to TSP. In *STOC*, pp. 14–27, ACM, 2020.

[52] W. T. Tutte, The factorization of linear graphs. *J. Lond. Math. Soc.* **22** (1947), 107–111.

[53] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge University Press, 2011.

[54] R. Yuster, Almost exact matchings. *Algorithmica* **63** (2012), no. 1–2, 39–50.

[55] R. Zenklusen, A 1.5-approximation for path TSP. In *SODA*, pp. 1539–1549, SIAM, 2019.

OLA SVENSSON

Ecole Polytechnique Fédérale de Lausanne (EPFL), School of Computer and Communication Sciences, CH-1015 Lausanne, Switzerland, ola.svensson@epfl.ch