An Optimal Algorithm for the Stacker Crane Problem on Fixed Topologies

Yike Chen* Ke Shi† Chao Xu‡

Abstract

The Stacker Crane Problem (SCP) is a variant of the Traveling Salesman Problem. In SCP, pairs of pickup and delivery points are designated on a graph, and a crane must visit these points to move objects from each pickup location to its respective delivery point. The goal is to minimize the total distance traveled. SCP is known to be NP-hard, even on tree structures. The only positive results, in terms of polynomial-time solvability, apply to graphs that are topologically equivalent to a path or a cycle.

We propose an algorithm that is optimal for each fixed topology, running in near-linear time. This is achieved by demonstrating that the problem is fixed-parameter tractable (FPT) when parameterized by both the cycle rank and the number of branch vertices.

1 Introduction

In the stacker crane problem (SCP), a stacker crane must retrieve and deliver a set of m items from and to specified locations within a warehouse. The goal is to determine an optimal order for these operations, constructing a tour that completes the task while minimizing the total distance traveled. The problem is typically modeled as a problem on a weighted graph, where the stacker crane moves between vertices, traversing the edges.

SCP was first studied by Frederickson et al. [10], who provided an equivalent formulation involving a tour on a mixed graph and designed a 9/5-approximation algorithm. Subsequently, there have been many studies focused on developing approximation algorithms for various SCP variants [14, 18, 20].

Regarding exact algorithms, Frederickson and Guan demonstrated that SCP is NP-hard even on trees [8], ruling out some of the simplest graph classes as candidates for polynomial-time solvability. However, Atallah and Kosaraju showed that when the input graph is a path, SCP can be reduced to a minimum spanning tree problem. Moreover, if the input graph is a cycle with n vertices, there exists an $O(m + n\log n)$-time algorithm [2]. Frederickson later improved the running time to match that of finding a minimum spanning tree, and also proving the algorithm to be optimal [9].

Since paths and cycles are topologically equivalent to a single edge or a self-loop, respectively, one might conjecture that SCP is solvable in polynomial time for any fixed topology. Fixed topologies have practical significance: the layout of warehouses is usually fixed, often resembling a grid with a limited number of aisles [4, 17]. For further details on various industrial applications of SCP, we refer readers to [6].

Our Contributions We present an optimal algorithm for solving the SCP for fixed topologies. Our algorithm generalizes Frederickson's approach, reducing to his algorithm when the topology is a path or a cycle [9]. Because our algorithm is more general, it also serves as an alternative, and we believe more intuitive, proof of the correctness of Frederickson's algorithm.

*cyike9982@gmail.com, University of Electronic Science and Technology of China.
†self.ke.shi@gmail.com, University of Science and Technology of China.
‡the.chao.xu@gmail.com, University of Electronic Science and Technology of China.

2 Preliminaries

We consider mixed graphs, which can contain both undirected edges and directed edges, the latter being referred to as *arcs*. A graph that contains only arcs is called a directed graph, while one that contains only undirected edges is called an undirected graph. A vertex is termed a *branch* vertex if its degree is at least 3.

For an arc (s, t), s is the tail and t is the head. For an undirected edge, both vertices are tails and heads. A *walk* is a sequence of edges such that the tail of the subsequent edge matches the head of the preceding edge. A *tour* is a walk that starts and ends at the same vertex. Let $\mathrm{MST}(m)$ denote the worst-case running time for finding a minimum spanning tree in a graph with m edges.

We now provide a graph-theoretical definition of the SCP. In SCP, we are given a simple graph $B = (V, E)$, called the *base graph*, together with a list of *requests* R. A request is an ordered pair of vertices, represented as an arc. For simplicity, we assume that the requests are distinct, although our results can handle cases with duplicate requests.

We use the following formulation of SCP, as introduced by Frederickson et al. [10].

Problem 1 (Stacker Crane Problem (SCP)) *Given a simple graph $B = (V, E)$ on n vertices and a set of p arcs R, with a cost function $c : E \cup R \to \mathbb{R}^+$, find a min-cost tour in the mixed graph $G = (V, E \cup R)$ that traverses each arc in R exactly once.*

Intuitively, by assigning the cost of an arc $(s, t) \in R$ to be the length of the shortest st-path in B, traversing an arc represents moving an object from s to t.

An *edge subdivision* operation involves inserting a new vertex in the middle of an edge. Specifically, given an edge uv, we delete the edge, add a new vertex w, and connect u and v via w by adding edges uw and wv. A graph H is called a *subdivision* of a graph G if H can be obtained through a sequence of edge subdivision operations starting from G. Two graphs, G_1 and G_2, are *topologically equivalent* if there exists a graph G that is a subdivision of both G_1 and G_2. Our goal is to show that SCP for a fixed topology can be solved in optimal running time.

Problem 2 (Stacker Crane Problem with Fixed Topology (SCP(H))) *Given a simple graph $B = (V, E)$ on n vertices that is topologically equivalent to a graph H, and a set of p arcs R, with a cost function $c : E \cup R \to \mathbb{R}^+$, find a min-cost tour in the mixed graph $G = (V, E \cup R)$ that traverses each arc in R exactly once.*

2.1 Circulations

Consider a mixed graph $G = (V, E \cup A)$ with a set of undirected edges E and directed arcs A. Assume an arbitrary orientation on the edges, providing a forward direction. We define $\delta^-(v)$ and $\delta^+(v)$ as the sets of in-edges/in-arcs and out-edges/out-arcs of a vertex v, respectively. Each edge and arc has an associated lower bound, upper bound, and cost function, denoted as $\ell, u : E \cup A \to \mathbb{Z}$ and $c : E \cup A \to \mathbb{R}$.

A *circulation* is a function $f : E \cup A \to \mathbb{Z}$ such that $\sum_{e \in \delta^-(v)} f(e) = \sum_{e \in \delta^+(v)} f(e)$ for every vertex $v \in V$. The circulation f is *feasible* with respect to the lower bounds ℓ and upper bounds u if $\ell(e) \leq f(e) \leq u(e)$ for every edge $e \in E \cup A$. Typically, the values of ℓ and u are clear from the context.

The cost of a circulation f under the cost function c is given by $\sum_{a \in A} f(a)c(a) + \sum_{e \in E} |f(e)|c(e)$. A feasible circulation with the minimum cost is called a *min-cost circulation*. An edge or arc e is termed *fixed* if $\ell(e) = u(e)$. An edge e is considered *free* if $\ell(e) = -\infty$ and $u(e) = \infty$. An arc is *free* if $\ell(e) = 0$ and $u(e) = \infty$. The *support* of f, denoted by $\mathrm{supp}(f)$, is defined as $\{e \mid f(e) \neq 0, e \in E \cup A\}$.

For a circulation f, we use $[f]$ to denote the set of all tours corresponding to f in the natural way: a tour where $f(e)$ represents the difference between the number of times the tour traverses the edge in the forward and backward directions. The set $[f]$ is referred to as a *homology class*, and two tours in $[f]$ are said to be *homologically equivalent*.

For the SCP problem with input graph $G = (V, E \cup R)$, the *corresponding circulation problem* is a circulation problem on G where each edge in E is free, and each arc in R is fixed with a flow value of 1. The costs of the edges and arcs remain the same as in the original SCP formulation.

2.2 Cycle space

For an undirected graph $G = (V, E)$, the functions $f : E \to \mathbb{Z}$ can be represented as vectors in $\mathbb{Z}^E$. The space of circulations is a submodule of $\mathbb{Z}^E$, known as the *integral cycle space*, or simply the cycle space. If the graph is connected, the dimension of the cycle space is $|E| - |V| + 1$ [3]. This value, $|E| - |V| + 1$, is referred to as the *cycle rank*.

A circulation f is called a *unit cycle flow* if $\mathrm{supp}(f)$ forms an undirected cycle (after considering all edges as undirected) and $|f(e)| = 1$ for all $e \in \mathrm{supp}(f)$. Let T be a spanning tree in G. For each edge $uv \notin T$, the *fundamental cycle* C_{uv} with respect to T is defined as the unit cycle flow that traverses uv in the forward direction and then traces back from v to u along the unique path in T.

The fundamental cycles with respect to T form a basis for the cycle space. That is, let $C_1, \ldots, C_r$ be the fundamental cycles with respect to T. Then, for any circulation f, we have $f = \sum_{i=1}^{r} \lambda_i C_i$ for some coefficients $\lambda \in \mathbb{Z}^r$ [15].

2.3 Techniques

Atallah and Kosaraju solved SCP(H) for the cases where H is either a path or a cycle [2]. Using a more modern and general perspective, their approach can be summarized as a three-step algorithm:

1. Find a min-cost circulation f in the corresponding circulation problem.
2. Find all circulations g such that $\|f - g\|_\infty \leq b$ for some bound b.
3. Find a min-cost tour in each homology class $[g]$, and return the minimum.

It is crucial that H is a path or a cycle, as their technique in each of these steps relies heavily on this property. We describe how to eliminate such a restriction.

In the first step, finding a min-cost circulation when H is either a path or a cycle was solved through an ad-hoc process in *linear time*. On a path, such a circulation is unique. For a cycle, they considered enumerate all possible flow sent along a single edge and search for the minimum. Alternatively, use a min-cost flow algorithm on a one-tree [13]. Both can obtain linear time algorithm. We show that min-cost circulation can be solved in linear time for arbitrary fixed H by reducing it to a problem similar to linear programming in a fixed dimension. See section 4.1.

For the second step, Atallah and Kosaraju showed that $b = 0$ suffices for paths, leading to an optimal algorithm. In the same paper, they established a bound of $b = p$ for cycles, where p is the number of requests. Fortunately, their technique does not explicitly generate all circulations, but instead searches through them using a binary search, implemented alongside step 3. Later, Frederickson improved the bound on b to 1 by squeezing the value of the min-cost tour between two one-dimensional functions, thereby achieving the desired proximity result [9]. This improvement not only made the algorithm faster but also allowed steps 2 and 3 to be separated.

In general, it is not difficult to argue that for any H, a bound of $b = p$ suffices. The number of feasible circulations in step 2 would then be $O(p^r)$, where r is the cycle rank of B. This is too large to yield a fast algorithm. Unfortunately, Frederickson's technique does not generalize to graphs with higher cycle ranks. Therefore, we need a proximity theorem for arbitrary graphs. This is the most technical part of our paper. As we will show in section 3, a bound of $b = r$ suffices. The algorithm to explicitly enumerate $O((2r + 1)^r)$ such circulations will be described in section 4.2.

For the third step, Atallah and Kosaraju found the min-cost tour in a homology class by computing a minimum spanning tree in an auxiliary graph. The auxiliary graph is constructed by contracting edges in the circulation that defines the homology class. Once again, computing a minimum spanning tree suffices because H is a path or a cycle. In section 4.3, we show that the correct problem to solve is the *minimum Steiner tree* in the auxiliary graph, which is an NP-hard problem in general. However, since the number of Steiner branch vertices depends only on H, the running time remains efficient.

3 Proximity result

Let $B = (V, E)$ be the base graph. Consider the mixed graph $G = (V, E \cup R)$ in the SCP problem and the corresponding circulation problem on G. Assume f is the min-cost circulation in G. We will show that there exists a circulation g in G such that the homology class $[g]$ contains the min-cost tour, and $\|f - g\|_\infty \leq r$, where r is the cycle rank of B.

For a unit cycle flow C, we call kC a cycle flow of value k. A circulation f is said to contain a circulation g if $|g(e)| \leq |f(e)|$ for each edge e, and $g(e)$ and $f(e)$ have the same sign. A circulation is called *elementary* if it does not contain a cycle flow of value 2. Intuitively, this means that no part of the flow circulates around a cycle more than once. In fact, we prove the proximity result by showing that $f - g$ is an elementary circulation in B, the undirected base graph.

3.1 Elementary Circulations

For a graph or mixed graph G, let $G' = \text{sym}(G)$ be the directed graph obtained by replacing each edge with two opposing arcs. Interestingly, we prove the proximity result by considering circulations in the residual graph of G', as it is easier to reason about.

A circulation f is called *connected* if $\text{supp}(f)$ is connected. The *residual graph* for a given directed graph G with lower and upper bounds ℓ and u, and a feasible circulation f, is the directed graph G_f along with new lower and upper bounds ℓ_f and u_f. For each arc $a \in A$, define $u_f(a) = u(a) - f(a)$ and $\ell_f(a) = \ell(a) - f(a)$. The residual graph G_f consists of all arcs for which either $u_f(a) \neq 0$ or $\ell_f(a) \neq 0$. Note that the lower bounds in the residual graph do not have to be non-negative. However, it is necessary that $\ell_f(a) \leq 0 \leq u_f(a)$, meaning that the zero circulation is always feasible. The fixed arcs in the original graph do not appear in the residual graph, so the complexity of the residual graph can be much smaller.

One can observe that the SCP can be reduced to a min-cost *connected* circulation problem on $G' = (V, A \cup R) = \text{sym}(G)$. Here, A are the arcs obtained from symmetrizing E. For arcs in A, the lower bound is 0 and the upper bound is infinity. For arcs in R, both the lower bound and upper bound are 1. If f' is a min-cost connected circulation in G', then it corresponds to a min-cost tour of the same cost in G.

Theorem 3.1 *Let f be a min-cost circulation in $G = (V, E \cup R)$, where arcs in R are fixed to* 1*, and edges in E are free. There exists a circulation g such that the min-cost tour is in the homology class $[g]$ and $g - f$ is an elementary circulation in the base graph $B = (V, E)$.*

Proof: Let $\Delta = g - f$. We can choose g such that $\|\Delta\|_1$ is minimized. Note that Δ is a circulation in B because f and g agree on the fixed arcs in R, hence Δ is zero outside of E.

Let $G' = \text{sym}(G)$, and let f' be the min-cost circulation in G', which corresponds to the min-cost circulation f in G in a natural way. Let g' be the min-cost connected circulation in G', such that $g(uv) = g'(u, v) - g'(v, u)$ if (u, v) is the positive orientation of the edge uv. Let $\Delta' = g' - f'$, which is a circulation in $B' = \text{sym}(B)$.

Consider the residual graph $G'_{f'}$. We label the arcs in $G'_{f'}$ as follows: for each edge $e \in E$, if one of the arcs derived from e has negative flow in Δ', we label it as e^-, and the other arc as e^+. If an arc has positive flow, we label it as e^+ and the other arc as e^-. If both flows are zero, we label them arbitrarily. Certainly, Δ' is a feasible circulation in $G'_{f'}$ because $f' + \Delta' = g'$ is a feasible circulation in G'.

Assume Δ is not elementary. Then, there exists an unit cycle flow C such that $2C$ is contained in Δ. In terms of $G'_{f'}$, this would imply that $\Delta'(e^+) - \Delta'(e^-) \geq 2$ for each $e \in C$. We consider a cycle flow C' in $G'_{f'}$ whose undirected version is C, and which uses the maximum number of edges with negative labels. That is, for each edge e, we set $C'(e^+) = 1$ if $\Delta'(e^+) > 0$ and $\Delta'(e^-) = 0$.

Certainly, $\Delta' - C'$ is feasible in $G'_{f'}$. The cost of C' cannot be negative, since the residual graph of a min-cost circulation cannot contain a negative cost cycle [1]. This implies that the cost of $g' - C'$ is no greater than the cost of g'.

Additionally, we need to show that $\text{supp}(g' - C')$ is connected. For each $e \in \text{supp}(C)$, there are two cases:

1. If $C'(e^+) = 1$, then $\Delta'(e^-) = 0$ and $\Delta'(e^+) \geq 2$. Therefore, $g'(e^+) - C'(e^+) = f'(e^+) + \Delta'(e^+) - 1 \geq 1$. Hence, $e^+ \in \text{supp}(g' - C')$.
2. If $C'(e^-) = -1$, then $g'(e^-) - C'(e^-) = g'(e^-) + 1 \geq 1$. Hence, $e^- \in \text{supp}(g' - C')$.

Since $\text{supp}(g' - C')$ would include at least one of e^+ or e^- for each $e \in C$, if $\text{supp}(g')$ is connected, then $\text{supp}(g' - C')$ is also connected. Because $g' - C'$ is a connected circulation of no greater cost than g', we have that $[g - C]$ also contains a min-cost tour. However, $\|\Delta - C\|_1 < \|\Delta\|_1$ because Δ contains C. This contradicts our choice of Δ having minimal L_1 norm. □

3.2 Elementary Circulations and L_∞ Norm

In this section, we bound the number of elementary circulations in a *graph* using its cycle rank. First, we strengthen the flow decomposition theorem for circulations, which typically states that a circulation can be decomposed into m cycle flows, where m is the number of edges [19]. We improve this bound to r, the cycle rank of the graph.

Theorem 3.2 *Every non-negative circulation on a graph with cycle rank r can be decomposed into at most r non-negative cycle flows. That is, $f = \sum_{i=1}^{r} \lambda_i C_i$, where $C_1, \dots, C_r$ are cycle flows and $\lambda_1, \dots, \lambda_r \geq 0$.*

Proof: We prove this by induction on the cycle rank r.

If the cycle rank is 0, then the graph contains no cycles, and the zero circulation is the only possible circulation.

Now, consider the case where the cycle rank is r. We can assume the graph is connected; otherwise, we can apply the proof to each connected component separately. Suppose there exists a cycle C with positive flow. Let $\lambda = \min_{e \in C} f(e)$. We include λC as a term in the decomposition.

Let D be the set of edges with zero flow after reducing the flow on C by λ. Assume that $G - D$ has k components. Observe that $k \leq |D| + 1$. If $k = |D| + 1$, this would imply that removing a single edge in D disconnects the graph, contradicting the fact that G contains a cycle C. Hence, $k \leq |D|$.

Let the components of $G - D$ be $V_1, \dots, V_k$. By the inductive hypothesis, the remaining circulation can be decomposed into $\sum_{i=1}^{k} (|E_i| - |V_i| + 1) = (m - |D|) - n + k \leq m - n = r - 1$ cycle flows. Therefore, we can decompose f into r cycle flows. □

Next, we show that elementary circulations have a small L_∞ norm.

Theorem 3.3 *Let f be an elementary circulation in a graph with cycle rank r. Then, $\|f\|_\infty \leq r$.*

Proof: Let f be an elementary circulation. Assume f is non-negative; if not, we reverse the orientation of the edges with negative flow so that f becomes non-negative. By Theorem 3.2, f can be decomposed into r non-negative cycle flows. Since f is an elementary circulation, each of the r cycle flows is a unit cycle flow. Hence, for each edge e, we have $f(e) = \sum_{i=1}^{r} C_i(e) \leq \sum_{i=1}^{r} 1 = r$. Therefore, $\|f\|_\infty \leq r$. □

The bound in Theorem 3.3 is tight. Consider a cycle rank r graph with $2(r+1)$ vertices consists of a path $v_1,\dots,v_{2r}$. There are also edges e_{r-i} between v_{2r-i} and v_{1+i} for $i \le r$. See Figure 3.1. It has cycle rank r. Let C_i be the fundamental cycle using edge e_i in the smaller to larger vertex direction, where $i \ge 1$. Consider the circulation $f = \sum_{i=1}^{r} C_i$, it is elementary, and $f(e_0) = r$.

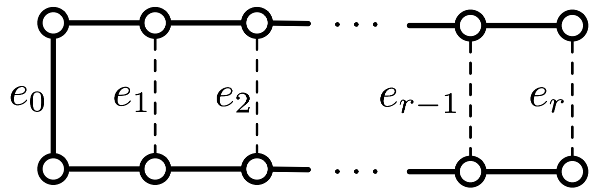

Figure 3.1. Example graph where there is an elementary circulation where $\|f\|_\infty = r$.

4 The new algorithm

This section describes the optimal algorithm for SCP(H).

4.1 Min-Cost Circulation

Recall that we are interested in finding the min-cost circulation f of $G = (V, E \cup R)$, where $f(a) = 1$ for all $a \in R$. The min-cost circulation problem was shown to be solvable in $m^{1+o(1)}$ time, where m is the number of edges in the graph [5]. In our case, the graph has $p + m$ edges. We can show that there is a linear time algorithm for base graphs with fixed cycle rank.

Theorem 4.1 *Let r be a fixed number. Consider a min-cost circulation problem where the edges are either fixed or free, and the free edges form a graph with cycle rank r. The min-cost circulation can be found in $O(m)$ time.*

Proof: Let $B = (V, E)$ be the undirected graph containing all the free edges, and let R be the set of fixed edges. We have $G = (V, E \cup R)$. Consider a spanning tree T on B, and let the edges in $E \setminus T$ be $e_1,\dots,e_r$. We define $g(\lambda_1,\dots,\lambda_r)$ to be the cost of the min-cost circulation f in G such that $f(e) = 1$ for $e \in R$ and $f(e_i) = \lambda_i$ for $1 \le i \le r$. To find the min-cost circulation, we have to solve $\min_\lambda g(\lambda)$. Note that there is an integral minimizer, as λ_i represents the flow value on edge e_i, which can always be taken to be integral.

Finding the minimum is equivalent to solving the following linear program. For each $a \in R$, compute the fundamental cycle C_a, and let $b_e = \sum_{e \in C_a} C_a(e)$. For each $e \in E$ and $1 \le i \le r$, we define $A_{e,i} = C_{e_i}(e)$.

Here, A is the edge-fundamental cycle incidence matrix, which is a network matrix [15].

$$
\begin{aligned}
\min_{\lambda \in \mathbb{R}^{E \setminus T}, x \in \mathbb{R}^T} \quad & \sum_e x_e \\
\text{subject to} \quad & x_e \ge c(e)\,|A_e \cdot \lambda + b_e| \quad \forall e \in E
\end{aligned}
$$

Zemel showed that the minimizer of mathematical programs of the above form can be seen as a generalization of r-dimensional L_1 linear regression, which can be solved using techniques similar to Megiddo's constant-dimension linear programming algorithm [16]. Consequently, the problem can be solved in $2^{O(2^r)} m = O(m)$ time [21]. □

4.2 Generating Circulations Near the Min-Cost Circulation

In this section, we show how to generate all circulations that are close to the min-cost circulation in terms of the L_∞ norm.

Lemma 4.2 *For an undirected graph with cycle rank r, the number of circulations f such that $\|f\|_\infty \le k$ is at most $(2k+1)^r$ and can be found in $O((2k+1)^r m)$ time.*

Proof: Consider any fundamental cycle basis of G, a graph with cycle rank r, with respect to some spanning tree T. The values on the edges outside T uniquely determine the circulation. For a circulation f such that $\|f\|_\infty \le k$, the absolute value of the flow on each edge outside T is at most k. Hence, there can be at most $2k+1$ choices for each of the r edges outside T. This shows that there can be at most $(2k+1)^r$ circulations with L_∞ norm at most k.

To construct all such circulations, we find a fundamental cycle basis $C_1, \ldots, C_r$. This takes $O(rm)$ time. Next, we generate circulations of the form $\sum_{i=1}^r \lambda_i C_i$ one by one, where $-k \le \lambda_i \le k$. The sequence of circulations is generated using a generalized Gray code, so two consecutive circulations differ in only a single cycle [11]. Therefore, it takes $O(m)$ time to generate a circulation from the previous circulation by augmenting a cycle. The total running time for generating all circulations with L_∞ norm no larger than k is $O((2k+1)^r m)$. □

Corollary 4.3 *Let f be the min-cost circulation in G. The number of circulations g in G such that $\|f - g\|_\infty \le r$ is $O((2r+1)^r)$ and they can be found in $O((2r+1)^r m)$ time, where m is the number of edges in G.*

Proof: By Lemma 4.2, there can be at most $(2r+1)^r$ circulations with L_∞ norm no larger than r in B. These circulations can be computed in $O((2r+1)^r m)$ time. For each such circulation h, we compute $g = f + h$. This takes an additional $O(m)$ time per circulation. Thus, the total time complexity is $O((2r+1)^r m)$. □

4.3 Min-Cost Tour in a Given Homology Class

Recall that we are interested in finding the min-cost tour in the homology class $[f]$ for some given circulation f in $G = (V, E \cup R)$, where $f(a) = 1$ for $a \in R$.

If f is already connected, then we are done, as we can find a tour that uses each edge e exactly $|f(e)|$ times. Otherwise, we need to find a tour in $[f]$. Note that in addition to the edges in $\text{supp}(f)$, some extra edges might need to be used by the tour. Specifically, there are some edges outside $\text{supp}(f)$ that must be traversed both forward and backward exactly once.

To address this, we take G and contract each connected component of $\text{supp}(f)$ into a single vertex. Let the resulting graph be B'. Note that only edges in E might remain, as all arcs in R have been contracted. Therefore, B' is a graph instead of a mixed graph. Moreover, we change the weight of the edges to twice their original cost since each edge would be traversed once forward and once backward. We then reduce the problem to the minimum Steiner tree problem.

For a graph $G = (V, E)$ with a set of terminal vertices $T \subseteq V$, $V \setminus T$ are called *Steiner vertices*. A Steiner tree is a tree that contains all vertices in T. The minimum Steiner tree with respect to a weight function $w : E \to \mathbb{R}^+$ is a Steiner tree of minimum weight.

We are interested in solving the minimum Steiner tree problem on B', where the terminals are the contracted vertices. This would give us the minimum set of extra edges the tour has to traverse to be connected. The minimum Steiner tree problem is NP-hard in general, but note that B' has only a bounded number of Steiner branch vertices. The number of Steiner branch vertices in B' is at most the number of branch vertices in B. Indeed, the contracted vertices are not Steiner vertices, so all Steiner branch vertices must be branch vertices in B. The Steiner vertices of degree 1 and 2 can be preprocessed away, so only Steiner branch vertices remain [7]. Hence, the minimum Steiner tree can be solved in $O(2^k \operatorname{MST}(m))$ time for a graph containing k Steiner branch vertices and m edges [12].

Since B' can have at most m edges, the minimum Steiner tree in B' can be found in $O(2^k \operatorname{MST}(m))$ time. This leads us to the following theorem.

Theorem 4.4 *For an input mixed graph $G = (V, E \cup R)$, let f be a circulation where $f(a) = 1$ for each $a \in R$. Finding a min-cost tour in $[f]$ takes $O(p + 2^k \operatorname{MST}(m))$ time, where $m = |E|$, $p = |R|$, and k is the number of branch vertices in $B = (V, E)$.*

4.4 Putting Everything Together

Combining all the components, we obtain the desired algorithm as described in Figure 4.1. We are now ready to prove our main theorem.

Theorem 4.5 *SCP(H) with an input graph of n vertices and p requests can be solved in $O(\operatorname{MST}(n) + p)$ time.*

Proof: Let B be the base graph and R be the set of requests. For a fixed topology, the cycle rank r and the number of branch vertices k are constants. If m is the number of edges in B, then $m = n + r - 1 = O(n)$.

Consider the algorithm in Figure 4.1. By Theorem 4.1, computing the min-cost circulation takes $O(m + p)$ time. By Corollary 4.3, enumerating all $O(1)$ possible circulations close to the min-cost circulation takes $O(m)$ time. By Theorem 4.4, finding the minimum-cost tour in the homology class of each circulation takes $O(\operatorname{MST}(m) + p)$ time. Using the fact that $m = O(n)$, we obtain a final running time of $O(\operatorname{MST}(n) + p)$. □

The running time in Theorem 4.5 is optimal. The term $\operatorname{MST}(n)$ is unavoidable, as finding the minimum spanning tree on n edges can be reduced to SCP(P) in linear time, where P is a path. Moreover, since reading the input requires at least $O(p)$ time, the term p is also tight.

```
SCP(B = (V,E), R, c):
  G ← (V, E ∪ R)
  f ← the min-cost circulation
  𝒞 ← all circulations g such that ‖f − g‖∞ ≤ r
  for g ∈ 𝒞
      record MinCostTourInHomologyClass(G, g, c) as a candidate
  return the min-cost candidate
```

Figure 4.1. Pseudocode for solving SCP given base graph B, requests R and cost function c.

5 Discussion

We simplified the presentation of our results, but our approach can be extended in multiple ways without increasing the time complexity.

As mentioned earlier in the preliminaries, we can handle duplicate edges in R without altering the complexity of the algorithm. Furthermore, we can allow demands on edges in R. Specifically, we can introduce a function $d : R \to \mathbb{N}$ such that, for each $a \in R$, the tour is required to traverse a exactly $d(a)$ times.

Our current cost function is symmetric; for an edge $e = uv$, traversing from u to v or from v to u incurs the same cost. However, our algorithm can be adapted to work with asymmetric cost functions as well.

```
MinCostTourInHomologyClass(G, f, c):
   ⟨⟨Find the min-cost tour in [f]⟩⟩
   H ← contract all components of supp(f) in G
   for each edge e ∈ V(H)
         w(e) ← 2c(e)
   T ← the vertices from contracting a component of supp(f)
   S ← Minimum Steiner tree on H with weights w and terminals T
   W ← construct a tour from S and f
   return W
```

Figure 4.2. Finding the min-cost tour for a given graph G and a circulation f on G

Finally, the algorithm can be extended to handle multiple stacker cranes, each starting at a different vertex, while optimizing the total distance traversed.

Although we provide an optimal algorithm, the constant factor hidden in the time complexity is extremely large, making it impractical for real-world applications. If the fixed topology has cycle rank r and k branch vertices, then the hidden factor is $2^{O(2^r)} + (2r+1)^r \cdot 2^k$. The bottleneck of $2^{O(2^r)}$ arises from solving the min-cost flow problem on a graph that is a tree with r additional edges. In practice, one can use slower, off-the-shelf min-cost flow algorithms to avoid the exponential dependency on r. However, there are two potential avenues for improvement:

1. The problem is very close to linear programming in a constant dimension, so techniques other than Megiddo's algorithm might be beneficial for improving the running time.

2. In the proof of Theorem 4.1, the matrix A is a network matrix, but Zemel's algorithm is designed for arbitrary matrices. This presents another opportunity for optimization.

The factor $(2r+1)^r$ is unlikely to be significantly improved if we need to enumerate all elementary circulations, as their number can be exponential with respect to r. However, there may be strategies to only consider a much smaller subset of elementary circulations. For example, our enumeration of elementary circulations is independent of cost information, which could be leveraged to reduce the search space.

Acknowledgements We would like to thank Jianbo Wang and Siyue Liu for the initial discussions related to this problem. Chao would like to thank Zexuan Liu and Luze Xu for their discussions on the proximity results of a related integer program.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows*. Pearson, Upper Saddle River, NJ, February 1993.

[2] Mikhail J. Atallah and S. Rao Kosaraju. Efficient Solutions to Some Transportation Problems with Applications to Minimizing Robot Arm Travel. *SIAM Journal on Computing*, 17(5):849–869, October 1988.

[3] Bela Bollobas. *Modern graph theory*. Graduate Texts in Mathematics. Springer, New York, NY, 1998 edition, December 2013.

[4] Hadrien Cambazard and Nicolas Catusse. Fixed-parameter algorithms for rectilinear steiner tree and rectilinear traveling salesman problem in the plane. *European Journal of Operational Research*, 270(2):419–429, 2018.

[5] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623, 2022.

[6] Ángel Corberán and Gilbert Laporte. *Arc Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015.

[7] Cees Duin. *Preprocessing the Steiner Problem in Graphs*, pages 175–233. Springer US, Boston, MA, 2000.

[8] G.N. Frederickson and D.J. Guan. Nonpreemptive Ensemble Motion Planning on a Tree. *Journal of Algorithms*, 15(1):29–60, July 1993.

[9] Greg N. Frederickson. A Note on the Complexity of a Simple Transportation Problem. *SIAM Journal on Computing*, 22(1):57–61, February 1993.

[10] Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[11] D. J. Guan. Generalized gray codes with applications. 1998.

[12] S. L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1(2):113–133, 1971.

[13] Bahman Kalantari and Iraj Kalantari. *A Linear-Time Algorithm for Minimum Cost Flow on Undirected One-Trees*, pages 217–223. Springer US, Boston, MA, 1995.

[14] Jianping Li, Xiaofei Liu, Weidong Li, Li Guan, and Junran Lichen. Approximation algorithms for the generalized stacker crane problem. In Xiaofeng Gao, Hongwei Du, and Meng Han, editors, *Combinatorial Optimization and Applications*, pages 95–102, Cham, 2017. Springer International Publishing.

[15] Christian Liebchen and Romeo Rizzi. Classes of cycle bases. *Discrete Applied Mathematics*, 155(3):337–355, 2007.

[16] Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, jan 1984.

[17] Kees Jan Roodbergen and René de Koster. Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43, 2001.

[18] Yuhui Sun, Wei Yu, and Zhaohui Liu. Approximation algorithms for some min–max and minimum stacker crane cover problems. *Journal of Combinatorial Optimization*, 45(1):18, 2022.

[19] David P. Williamson. *Network Flow Algorithms*. Cambridge University Press, 2019.

[20] Wei Yu, Rui-Yong Dai, and Zhao-Hui Liu. Approximation Algorithms for Multi-vehicle Stacker Crane Problems. *Journal of the Operations Research Society of China*, 11(1):109–132, March 2023.

[21] Eitan Zemel. An O(n) algorithm for the linear multiple choice knapsack problem and related problems. *Information Processing Letters*, 18(3):123–128, March 1984.