Séminaire Lotharingien de Combinatoire **80B** (2018)
Article #47, 12 pp.

Proceedings of the 30th Conference on Formal Power Series and Algebraic Combinatorics (Hanover)

Parking Functions on Oriented Trees

Westin King*[1] and Catherine Yan†[1]

[1]*Department of Mathematics, Texas A&M University. College Station, TX 77843. USA.*

Abstract. Classical parking functions arise from an analogy of drivers attempting to park in spots along a one-way street, which we consider a directed path, via a fixed parking process. We give a new generalization of the parking process, as well as prime parking functions, to all directed graphs. We then present some enumerative results for trees with edges oriented either towards or away from a root.

Keywords: parking function, digraph, prime parking functions

1 Introduction

Classical parking functions first appeared in a 1966 paper by Konheim and Weiss [3], where they examined the probability that n drivers could find a parking spot on a one-way street with n spaces when their starting location was randomly chosen. Suppose instead of random starting positions, the i^{th} driver prefers spot s_i. One-by-one, the drivers enter and drive to their preferred spots. If it is available, the driver parks there, otherwise parking in the first open spot afterwards.

Definition 1.1. If the parking procedure allows all the drivers to park, the sequence of preferences, $s = (s_1, s_2, \ldots, s_n)$, is called a *parking function of length n*.

For a parking lot with n spaces, there are $(n+1)^{(n-1)}$ distinct parking functions [3, 6]. Figure 1 gives an example.

Figure 1: A path with classical parking function $s = (1, 3, 2, 3, 1)$.

Alternatively, let $s \in [n]^n$. Then s is a parking function if and only if we have

$$|\{j : s_j \geq i\}| \leq n + 1 - i \text{ for all } i \in [n]. \tag{1.1}$$

An immediate observation from this characterization is that the order in which the cars enter does not matter. If we concern ourselves with only the number of drivers preferring

*wking@math.tamu.edu
†cyan@math.tamu.edu

each spot, we can write terms in non-decreasing order. Such parking functions are called *increasing* and are counted by the Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$ [6, Exercise 6.19s].

Another interesting subset of classical parking functions arises by making the inequalities in Equation (1.1) strict, whenever possible.

Definition 1.2. A parking function s is called *prime* if and only if for all $2 \leq i \leq n$, we have

$$|\{j : s_j \geq i\}| < n + 1 - i.$$

Prime parking functions are also those which, after removing any 1 from the sequence, are parking functions on the first $n-1$ spaces (see [6, Exercise 5.49f]). Named by Gessel, there are $(n-1)^{(n-1)}$ prime parking functions of length n. The parking function in Figure 1 is a prime parking function since both $(3,2,3,1)$ and $(1,3,2,3)$ fill the first four spots. The name "prime" arises because the parking function can not be decomposed into two smaller ones by deleting an edge from the graph. We discuss this property in Section 3.

The parking process has seen a limited extension to digraphs, in particular those in which each vertex has out-degree at most 1. In this case, the path a driver takes after failing to park at her preferred spot is unique. In [4], Lackner and Panholzer focus on trees on vertex set $[n]$ with edges oriented towards a root and mapping digraphs, those with vertex set $[n]$ and edge set $E = \{(i, f(i))\}_{i=1}^{n}$ for some function $f : [n] \to [n]$. Let F_n denote the total number of parking functions on trees with $[n]$ vertices and M_n denote likewise the total number of parking functions on mapping digraphs. They prove $n \cdot F_n = M_n$ and give precise formulas for each.

In Section 2, we give a generalization of the parking process and extend the notion of prime parking functions to arbitrary digraphs. In Section 3, we count the total number of prime parking functions on trees with edges oriented towards a root. We then reverse the edge orientation in Section 4 and give a relationship between trees with edges oriented away from the root and inverse mapping digraphs. Finally, in Section 5, we give some directions for future research.

2 The Parking Process on Digraphs

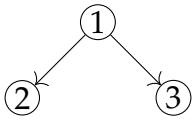
Postnikov and Shapiro have already defined a notion of parking functions on a (di)graph G, called G-parking functions [5]. Their extension is concerned with the number of (directed) spanning trees of a graph, and thus the K_{n+1}-parking functions (where K_{n+1} is the complete graph) are in bijection with the classical parking functions of length n. Our generalization instead extends the parking process and is distinct.

Consider a parking lot more complex than a line, where the parking spots are represented by vertices of a digraph D and the paths between parking spots are the edges. We

generalize the process whereby the drivers find a parking spot: beginning their search at their preferred spot, moving through D following edge orientations, and parking in the first unoccupied space they encounter. This parking process is deterministic when the vertices in D have outdegree at most one, as drivers have a unique path along which to search. However, if a vertex has outdegree at least two, a driver may have to choose along which branch to travel. In the case of a general digraph D with vertex set $[n]$ and $s \in [n]^n$ we give the following definition [7].

Definition 2.1. For a digraph D, a sequence $s \in [n]^n$ is a *parking function on D* if and only if drivers with preference sequence s can choose the paths they take during parking in such a way that all drivers park.

If s is a parking function on D, we will say the pair (D, s) is a parking function and for simplicity, we will assume $V(D) = [n]$. On digraphs with at least one vertex of outdegree ≥ 2, the order in which spots are filled by drivers is not necessarily well-defined. For example, in the following digraph, the sequence (1,1,1) is a parking function, but the order in which spots 2 and 3 are filled is not unique. The sequence (1,1,2) is also a parking function because the second driver can choose to travel to 3 during her search.



We have the following additional characterization of a parking function on digraphs. For $i, j \in [n]$, we say $i \preceq_D j$ if and only if there exists a directed path from i to j in D. By convention, we will say that $i \preceq_D i$, making $\preceq_D$ a quasiorder on the vertices of D. For vertex i, let

$$V_D(i) = \{j \in V(D) : i \preceq_D j\}.$$

Then for any $A \subseteq [n]$ define

$$R_D(A) = \bigcup_{i \in A} V_D(i).$$

Theorem 2.2. *Let D be a digraph with vertex set $[n]$ and $s \in [n]^n$. Then s is a parking function on D if and only if for all $A \subseteq [n]$ we have*

$$|\{i : s_i \in R_D(A)\}| \leq |R_D(A)|.$$

We omit the proof due to length, but we mention a few words on the main ideas.

Idea of proof for Theorem 2.2. As in the classical case, if (D, s) is a parking function, then so is (D, s^*) for any rearrangement s^* of the letters of s. Thus, we need only show that all cars can park for some ordering of s.

For a pair (D, s) satisfying the subset hypothesis of Theorem 2.2, we choose some $\emptyset \neq A \subsetneq \{s_i : i \in [n]\}$, if possible, such that $|\{i : s_i \in R_D(A)\}| = |R_D(A)|$ holds. We use induction to show that s can be broken into a parking function on the subgraph induced by $R_D(A)$ and its complement in D. If no such A exists, we find a maximal strongly connected subset B of vertices such that no edge (i, j) has $i \in B$ and $j \notin B$. Since B is strongly connected and no edge leaves B, all cars preferring B must and do park in B. We then construct a digraph D' and sequence s' that represents the remaining vertices and cars after those preferring B have parked and show that (D', s') is a parking function. □

Theorem 2.2 is a "Hall-like" condition that matches cars to vertices they could potentially park at. However, it is not a direct consequence of the Marriage Theorem since the parking process requires that cars must park at the first empty spot they arrive at and not necessarily the spots a matching would pair them with. We also extend the notion of prime parking functions to digraphs.

Definition 2.3. A parking function (D, s) is *prime* if and only if for all $A \subseteq [n]$ such that $R_D(A) \neq [n]$ we have

$$|\{i : s_i \in R_D(A)\}| < |R_D(A)|.$$

If we choose D to be the path in the case of classical parking functions, we see that for any $A \subseteq [n]$, $|R_D(A)| = |R_D(\{a_1\})| = n + 1 - a_1$ where a_1 is the smallest element of A. Thus, the prime parking functions are those who, for $2 \leq j \leq n$, have

$$|\{i : s_i \geq j\}| < n + 1 - j,$$

and so these are indeed the classical prime parking functions.

3 Prime Parking Functions On Sink Trees

In this section, we consider *sink* tree digraphs, those rooted trees with vertex set $[n]$ and edges oriented towards the root. For a sink tree T and $v \in [n]$, we let $T_v = \{u : u \preceq_T v\}$, the vertices in the maximal subtree rooted at v. Restricting Theorem 2.2 to sink trees is equivalent to the definition given in [4] and gives the following characterization for parking functions on sink trees:

Corollary 3.1. *The pair (T, s) is a* parking function *if and only if for all $v \in [n]$, we have $|\{i \in [n] : s_i \in T_v)\}| \geq |T_v|$. Further, (T, s) is prime if and only if the inequality is strict for all non-root vertices v.*

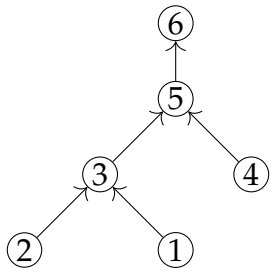


Figure 2: A sink tree.

Figure 2 shows a sink tree on which $(2,2,5,1,4,2)$ is a parking function and $(1,4,1,2,2,4)$ is a prime parking function.

Because paths between points are unique, prime parking functions have the following alternative description. Let (T,s) be a parking function. For an edge e, we say e is *used* by s if, after failing to park at her preferred spot, some driver crosses e while searching for a parking spot. If every edge in T is used by s, then this implies (T,s) is prime. On the other hand, if (T,s) is prime, we see that s uses any edge (u,v) by considering the maximal subtree rooted at u.

Theorem 3.2. *For $n \geq 1$, let P_n be the total number of prime tree parking functions (T,p) with $|T| = n$. Then*

$$P_n = (2n-2)!$$

To prove this, we use the following result of Lackner and Panholzer [4]:

Theorem 3.3. *Let F_n be the total number of parking functions on sink trees with vertex set $[n]$ and set $F(x) = \sum_{n\geq 1} F_n \frac{x^n}{(n!)^2}$. Then $F(x)$ satisfies the equation*

$$F(x) = T(2x) + \ln\left(1 - \frac{T(2x)}{2}\right), \tag{3.1}$$

where $T(x)$ is the tree generating function satisfying $T(x) = xe^{T(x)}$. Further, F_n is given by

$$F_n = ((n-1)!)^2 \cdot \left(\sum_{i=0}^{n-1} \frac{(n-i)\cdot(2n)^i}{i!}\right).$$

Sketch of proof of Theorem 3.2. We decompose any parking function (T,s) into a "core" component supporting a prime parking function and some collection of other components with normal parking functions, attached to the core component. The core component is given by the maximal subtree T_0 containing the root of T such that the subsequence s_0 of s consisting of cars preferring T_0 is a prime parking function on T_0. The

other components, $\{(T_i, s_i)\}_{i=1}^{r}$, are identified by deleting the edges in T which are incident to exactly one vertex in T_0 and the respective subsequences of s. Figure 3 gives a general depiction of this decomposition. Dashed edges are those unused edges connecting the T_i to the component T_0, which has a prime parking function.

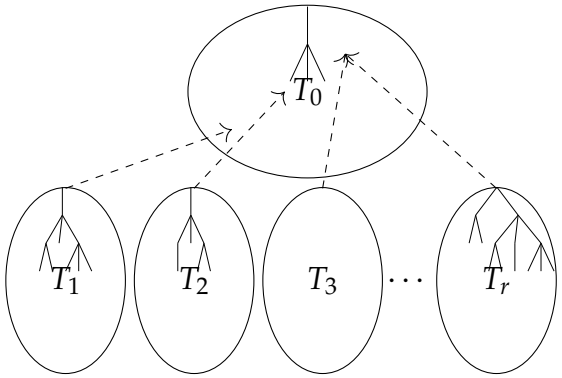


Figure 3: Decomposing into a "core" component containing the root.

This decomposition is unique for every parking function, so in order to construct a parking function with n drivers, we may select parking functions for the prime and additional components, their labellings, how their preference sequences merge to form one of length n, and how the others attach to the prime component. This gives

$$F_n = \sum_{r\geq 0} \frac{1}{r!} \sum_{\sum_{i=0}^{r} k_i = n} P_{k_0} F_{k_1} \cdots F_{k_r} \binom{n}{k_0, k_1, \ldots, k_r}^2 (k_0)^r.$$

We set

$$P(x) = \sum_{n\geq 1} P_n \frac{x^n}{(n!)^2},$$

and sum over n to obtain the relation

$$F(x) = P\left(xe^{F(x)}\right). \tag{3.2}$$

The $(n!)^2$ is chosen to account for both the label permutations on the trees and the choice of indices in which each s_i appears.

If we set $z = z(x) = xe^{F(x)}$ and $y = y(x) = \frac{T(2x)}{2}$ and combine Equation (3.2) with Equation (3.1) we see that

$$P(z) = 2y + \ln(1-y).$$

Solving for y in terms of z by using the relation $T(x) = xe^{T(x)}$, and applying the compositional inverse of z, we conclude

$$P(x) = 2xC(x) + \ln(1 - xC(x)),$$

where $C(x)$ is the ordinary generating function for the Catalan numbers and has analytic expression

$$C(x) = \frac{1 - \sqrt{1 - 4x}}{2x}.$$

After taking the derivative and some algebra, we notice

$$P'(x) = C(x).$$

Hence,

$$P(x) = \sum_{n \geq 1} \frac{C_{n-1}}{n} x^n,$$

and the conclusion follows. □

The simple formula of Theorem 3.2 demands a bijective proof, so we will construct one. Given a prime parking function (T, p), we first standardize the labels of T by considering T as a plane tree and ordering siblings by when a driver first crosses the edge to their parent vertex, earliest on the right. We then choose the standard labeling to be given by post-order: starting from the left, travel around the border of the tree labeling as one reaches the right side of a vertex. The first tree in Figure 4 shows a tree labeled via post-order. There are $n!$-many relabelings of a tree with a standard labeling, so what remains is to count the number of prime parking functions with standardized labelings.

To do this, we construct a bijection α, which sends a prime parking function (T, p) with a standard labeling to a plane tree O whose non-root vertices are labeled by $[n-1]$. There are C_{n-1} plane trees on n vertices and $(n-1)!$ labelings of the non-root vertices, combined with the $n!$-many parking functions relabeled to the same standard labelings, there are $n!C_{n-1}(n-1)! = (2n-2)!$ prime parking functions.

Idea of the bijection α. We inductively define α. For the base case, if T is a singleton, then $\alpha((T, p))$ is an unlabeled vertex. Otherwise, α is defined through the following steps:

1. Park all except the final driver. Deleting edges not yet used defines collection of prime parking functions linearly ordered by the order in which the final driver would visit the vertices of the trees on her search for a parking spot.

2. For each component in this decomposition, consider the set containing the indices of the cars preferring the component and if the final driver would first enter the component at the k^{th} smallest vertex, mark the k^{th} smallest element of the set.

3. Apply α to the standardized versions of each component.

4. Relabel the results of Step 3 with the unmarked elements from Step 2, preserving relative order, and label the roots by the marked elements. Attach the roots to a new unlabeled root in order.

To reverse, one follows the steps backwards, being sure to attach a vertex to the left of any siblings to preserve the ordering of the tree. □

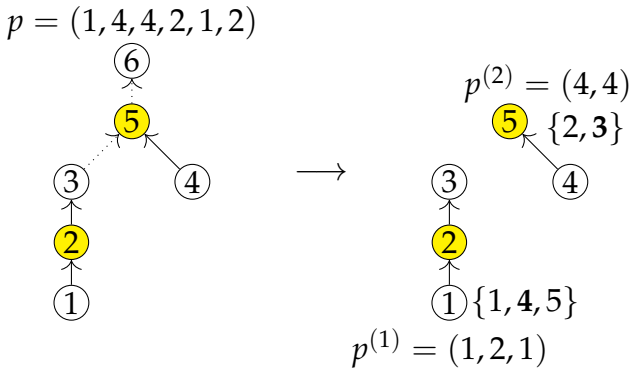


Figure 4: Steps 1 and 2 of α.

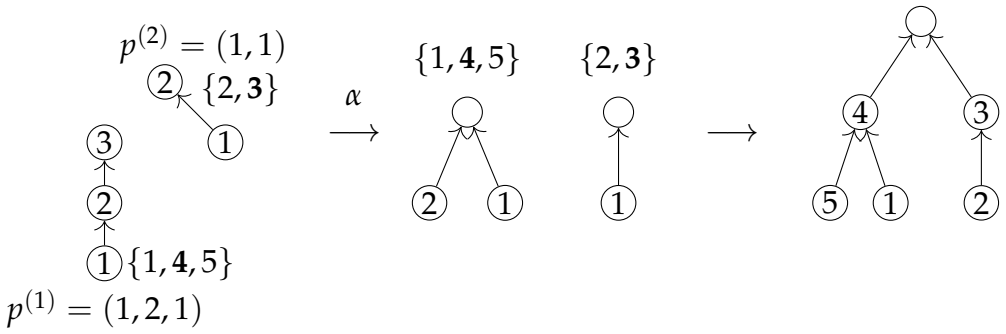


Figure 5: Steps 3 and 4 of α. Note the relabeling from Figure 4.

Figure 4 gives an example of steps 1 and 2, while Figure 5 shows steps 3 and 4, after labeling the components via post-order. The dotted edges in Figure 4 are those

unused before the final driver and the shaded vertices are the ones in each component that the final driver would first encounter during her search and correspond to the bold elements in the sets. Notice that the elements in the set near $p^{(i)}$ are the indices in p of the subsequence $p^{(i)}$ for $i \in \{1, 2\}$.

Details of this bijection can be found in [2].

4 Source Trees and Mappings

We now turn our attention to *source* trees, those with edges oriented away from the root. For a sink tree T, we denote by $\widetilde{T}$ the source tree obtained by reversing edge orientations. Recall that F_n is the total number of parking functions on sink trees with n vertices, and we denote $\widetilde{F}_n$ similarly for source trees. An exact formula for F_n is given in Theorem 3.3, but we have yet to find a formula for $\widetilde{F}_n$. Although related, the exact relationship between the two quantities F_n and $\widetilde{F}_n$ is not immediately clear. The initial values of F_n and $\widetilde{F}_n$ are given by $1, 6, 132, 6384$ and $1, 6, 135, 6760$, respectively. Restricting to individual trees, we have some comparison of the number of parking functions.

Theorem 4.1. *The number of parking functions on a sink tree T is not greater than the number of parking functions on the corresponding source tree $\widetilde{T}$. Equality occurs if and only if T is a path.*

Idea of proof. Given a parking function (T, s), we reversibly construct a parking function $(\widetilde{T}, \widetilde{s})$ by reassigning driver preference along judiciously chosen paths, dependent on s. In fact, $\widetilde{s} = s_\tau$ for some $\tau \in \mathfrak{S}_n$. The parking function (T, s) can be considered as a collection of prime parking functions by parking drivers and highlighting edges as they are crossed. Components connected by highlighted edges are prime by construction. Our driver reassignment works on each of these prime components individually, so it is sufficient to assume that (T, s) is in fact prime.

Beginning with the smallest leaf, travel along the path to the root until the number of drivers preferring a vertex in the path is equal to the number of vertices in the path. Let there be k_1 vertices in the path. Reassign drivers preferring the i^{th} vertex in the path to instead prefer the $(k_1 + 1 - i)^{\text{th}}$. If the i^{th} vertex has label α and the $(k_1 + 1 - i)^{\text{th}}$ vertex has label β, add the transposition $(\alpha\beta)$ to τ. Proceed to the next smallest leaf. At any point, if one would add a vertex to a path that is already a member of a previous path, skip over it. Once one has accounted for all the leaves, reverse edge orientations to form a source tree. By construction, all drivers are able to park if they follow the path their preferred vertex was in.

The parking function on $\widetilde{T}$ which has all cars preferring the root is only obtainable in this way when T happens to be a path, from which the result follows. □

Summing the parking functions over all trees with $|T| = n$,

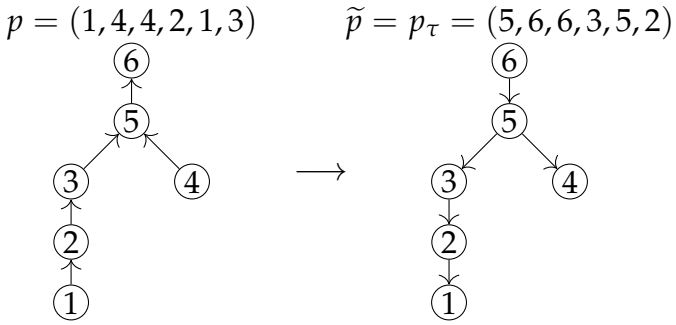


Figure 6: Constructing a parking function on $\widetilde{T}$ from one on T. $\tau = (15)(23)(46)$

Corollary 4.2. *For $n \geq 1$, we have*

$$F_n \leq \widetilde{F}_n$$

with equality if and only if $n \in \{1, 2\}$.

We next consider parking functions on mapping digraphs and study their relationship to those on the oriented trees. For $f : [n] \to [n]$, we consider the digraphs G_f and $\widetilde{G}_f$ with vertex set $[n]$ and edge sets $E(G_f) = \{(i, f(i)) : i \in [n]\}$ and $E(\widetilde{G}_f) = \{(f(i), i) : i \in [n]\}$. We will call G_f the *sink* mapping digraph and $\widetilde{G}_f$ the *source* mapping digraph. These graphs are formed by attaching the roots of sink or source trees, respectively, to cycles. We allow multiedges when the cycle is of length 2 and loops when the cycle is only a single root vertex. Let M_n and $\widetilde{M}_n$ denote the total number of parking functions (G_f, s) and $(\widetilde{G}_f, t)$, summed over all $f : [n] \to [n]$. Then we have

Theorem 4.3. *For $n \geq 1$,*

$$n \cdot \widetilde{F}_n = \widetilde{M}_n$$

Idea of proof. Consider a parking function $(\widetilde{T}, \widetilde{s})$ and some vertex $v \in \widetilde{T}$. Identify the edges in the path between the root and v that could be removed without affecting parking. If an edge is not necessary, a driver must prefer the terminal vertex of the edge, otherwise the vertex remains unfilled. Denote these terminal vertices by $\{v_i\}_{i=1}^r$ with indices increasing away from the root. At least one driver must prefer each v_i, so let ℓ_i be the index in $\widetilde{s}$ at which v_i first appears. Finally, add the vertex v_i to a set J if and only if $\ell_i > \ell_m$ for $1 \leq m \leq i-1$. Notice that the root of the tree, denoted v_1, is always in J. For each $v_i \in J$, detach the path edge with terminus v_i to create the components of the mapping digraph, and assign the first path edge in each component as the new terminus of each edge just detached. Add a new edge from v to the first path edge in the component of v. This creates a collection of components, each with one cycle, for which

the vertices in the cycles are precisely those who were on the path between the root and v in $\widetilde{T}$. □

Figure 7 gives and example of this transformation and has $J = \{1, 3, 5\}$, creating three components. The chosen vertex is shaded, while the dotted edges are those unused during parking. Since the edges are unused, they can be manipulated without affecting the drivers' ability to park.

$$\widetilde{s} = (2, 3, 4, 1, 3, 5, 1)$$

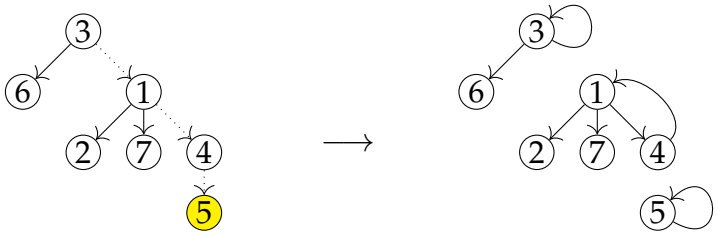


Figure 7: Turning a source tree into an inverse mapping digraph.

The proof is structurally similar to one by Lackner and Panholzer on sink trees and digraphs (Theorem 3.6, [4]), which states that for $n \geq 1$, $n \cdot F_n = M_n$. However, our result is distinct because, unlike on sink trees and mapping digraphs, the order in which spots are filled during parking on source trees and mapping digraphs is not well-defined. We must determine which edges to manipulate in order to form the mapping digraph without considering which spots along the path between the root and v are filled at a given step during parking. Additionally, it is not immediately clear that at least one edge in the cycle of each component of the source mapping digraph is not necessary for parking, which is crucial to reversing the bijection.

So far, formulas for $\widetilde{F}_n$ and the number of prime parking functions on source trees, $\widetilde{P}_n$, have proven elusive. In the case of both F_n and P_n, one can decompose the parking function based on the final vertex filled. However, since the order in which vertices are occupied is not necessarily well-defined when the maximum outdegree of a vertex is more than 1, this approach does not work for general source trees.

5 Final Words and Future Work

In this paper, we generalized both the parking process and the notion of *prime* parking functions to general digraphs. We then gave an enumerative result on trees with edges orients towards a root and discussed some relationships between several families

of digraphs. We note here that we can also generalize other special kinds of parking functions, such as *increasing parking functions*, which Butler, Graham, and Yan [1] have done to trees. This opens up many more enumeration problems, several of which we have answers to. Below we present several other avenues for research.

1. Find a formula for $\widetilde{F}_n$, the total number of parking functions on source trees with vertex set $[n]$.
2. Similarly, find formulas for the number of increasing parking functions (called parking distributions, see [1]) and the number of prime parking functions on source trees.
3. Find the exact relationship between F_n and $\widetilde{F}_n$.
4. Study the number of parking functions on various families of graphs.

Acknowledgments

The authors would like to thank the referees for their numerous helpful remarks and suggestions.

References

[1] S. Butler and C. Yan. "Parking distributions on trees". *European J. Combin.* **65** (2017), pp. 168–185. DOI: 10.1016/j.ejc.2017.06.003.

[2] W. King and C. Yan. "Prime Parking Functions on Rooted Trees". 2018. arXiv: 1804.01616.

[3] A. Konheim and B. Weiss. "An Occupancy Discipline and Applications". *SIAM J. Appl. Math.* **14**.6 (1966), pp. 1266–1274. DOI: 10.1137/0114101.

[4] M.-L. Lackner and A. Panholzer. "Parking functions for mappings". *J. Combin. Theory Ser. A* **142** (2016), pp. 1–28. DOI: 10.1016/j.jcta.2016.03.001.

[5] A. Postnikov and B. Shapiro. "Trees, parking functions, syzygies, and deformations of monomial ideals". *Trans. Amer. Math Soc.* **142**.8 (2004), pp. 3109–3142. DOI: 10.1090/S0002-9947-04-03547-0.

[6] R. Stanley. *Enumerative Combinatorics*. Vol. 2. Cambridge University Press, 1999.

[7] C. Yan. "Parking Functions on Digraphs". Preprint. 2014.