Parking Functions of Fixed Displacement

Lucas Chaves Meyles, Richter Jordaan,
Gordon Kirby, Sam Sehayek, Ethan Spingarn

ICERM 2022

Abstract

Displacement in the context of classical parking functions measures the total number of spots passed over when all the cars have parked. Utilizing Tutte polynomials, the number of parking functions with a given total displacement is known. We obtain enumerative results for the number of classical and prime parking functions with a fixed displacement partition when up to three cars are displaced, and suggest a direction to generalize these formulae for any number of cars. Moreover, an algorithm is obtained that computes the number of parking functions exhibiting specific displacements that is significantly more efficient than direct sieving techniques.

1 Introduction

Parking functions were introduced independently in the contexts of [13, 9, 14] and are an established area of research that have connections to many combinatorial objects like labeled trees, non-crossing partitions, the Shi arrangement, symmetric functions, and others. See [16] for a survey. In this paper we will consider their relation to hashing problems in which parking functions are in terms of numbered cars parking on a linear lot.

The goal of this paper is to provide tools to enumerate the number of parking functions exhibiting specific displacement partitions. The paper is structured as follows: We begin by introducing the relevant preliminaries. Chief among them is the observation that all displacement for a parking function is completely contained within the nontrivial primes in its prime decomposition. In Section 3, this connection will be fully developed and results will be proven about the general shape, length, and structure of primes exhibiting displacement. These ideas become crucial in understanding our technique towards obtaining direct enumerative results for cars that displace up to 3 cars, the subject of Section 4. In the case that a single car is displaced, Second-order Eulerian numbers make an appearance. In general, the procedures followed in this paper can be applied to larger numbers of displaced cars, which will be discussed, however the formulae become slightly unwieldy and are not believed to add any particular insight so they are omitted from this paper. For clarity, many of the longer case-work combinatorial proofs will be presented in the Appendix A.

In Section 5 an algorithm will be introduced that offers significant efficiency advantages to direct sieving techniques for identifying and enumerating parking functions with fixed displacement but varying displacement partitions. One of the important tools this algorithm relies on is the subject of Section 2, the Parking Rearrangment for a parking function, which is a specific reordering that essentially captures all the information about a parking function's displacement. We will do some analysis on the specific computational advantages to the proposed algorithm and conclude with some data drawn from this program for specific displacement partitions in parking functions of various lengths. Appendix B will comprise of the algorithm's Sage implementation with full documentation.

1.1 Preliminaries

Consider the scenario where n cars are trying to park on a one-way street with n parking spots and each car has a preferred parking slot (between 1 and n). Starting with the first car, each car drives up to its preferred slot. If the slot is available, the car parks. If the slot is occupied, the car parks in the next available slot after its preference. However, since the cars drive on a one-way street, if all of the slots at and beyond the car's preferred slot are taken, the car fails to park.

If we store the preferred spots of the cars in an n-tuple $\alpha = (a_1, a_2, \ldots, a_n)$, where $a_i \in [n]$ is the preferred slot of the i-th car, then α is a classical **parking function** of length n if and only if all cars are able to park under these conditions. An equivalent definition is that when arranged in nondecreasing order, α satisfies $a_i \leqslant i$. Denote by PF_n the set of all parking functions of length n.

As an example, $\beta = (4, 1, 2, 2, 1)$ is a parking function, with cars parking in slots 4, 1, 2, 3, and 5 respectively. However, the sequence $(4, 3, 2, 2, 3)$ is not a parking function, since when the last car tries to park with preference 3, all of the slots from 3 to 5 are already occupied, so the last car fails to park.

One subset of classical parking functions of particular interest in this paper are prime parking functions. A parking function of length n is a **prime parking function** if removing any instance of a 1 yields a parking function of length $n-1$. For instance, the parking function $(1, 3, 2, 1, 2)$ is prime, since removing the first instance of a 1 produces the sequence $(3, 2, 1, 2)$ and removing the second instance of a 1 produces the sequence $(1, 3, 2, 2)$, both of which are parking functions (in general, it turns out that the choice of 1 removed doesn't matter). However, the parking function $(1, 1, 2, 4)$ is not prime, since after removing a 1, the sequence $(1, 2, 4)$ is not a classical parking function of length 3. The set of all prime parking functions of length n is denoted by PPF_n. Prime parking functions are also defined in [15] and are examples of rational parking functions which are discussed at length in [3]. For length n they are enumerated by the formula $(n-1)^{n-1}$ (c.f. [15, 3]).

With the previous examples in mind, we notice that since different cars are allowed to have the same preferred slot, a parking function may have cars that are forced to park beyond their preferences. If car in a parking function has preferred slot a, it parks in slot $s \geqslant a$ and we say that the car experiences **displacement** $s - a$. For a parking function $\alpha = (a_1, a_2, \ldots, a_2)$, we can write the permutation $s = (s_1, s_2, \ldots, s_n) \in Sym_n$ encoding that car i parks in slot s_i. This permutation is referred to in the literature (see [7]) as the **outcome** of the parking function. Then the **displacement vector** $d = (d_1, d_2, \ldots, d_n) = (s_1 - a_1, s_2 - a_2, \ldots, s_n - a_n)$ is the n-tuple measuring how many slots beyond its preference each car parks. The previous example $\beta = (4, 1, 2, 2, 1)$ has displacement vector $(0, 0, 0, 1, 4)$. In general, the first entry of the displacement vector is 0, and the others are nonnegative integers.

For any parking function $\alpha \in PF_n$, we call the sum of the entries of the displacement vector the **total displacement** $D(\alpha)$ of the parking function. The total displacement measures the total number of slots that are passed over by all cars in a parking function in the entire parking procedure. Since the entries of the displacement vector are nonnegative integers, the nonzero terms of the displacement vector arranged in nonincreasing order gives the **displacement partition** $\lambda \vdash D(\alpha)$ of a parking function, the integer partition of the total displacement. The parking function $\beta = (4, 1, 2, 2, 1)$ has total displacement $D(\beta) = 5$ and has displacement partition $\lambda \vdash 5 = 4 + 1$.

2 Parking Order and the Parking Rearrangement

Definition 2.1. A parking function $\alpha = (a_1, \ldots, a_n) \in PF_n$ is **parking-ordered** if $a_i \leqslant i$ for all $i \in [n]$. Denote the set of all parking-ordered parking functions by PF'_n.

Any parking function $\beta = (b_1, \ldots, b_n)$ can be rearranged into an parking-ordered parking function β' (referred to as the **parking rearrangement** of β) as follows. Let $s = (s_1, \ldots, s_n)$ be the outcome permutation of $[n]$ which records where each car in β parks. Then,

$$\begin{pmatrix} s_1 & s_2 & \cdots & s_n \\ b_1 & b_2 & \cdots & b_n \end{pmatrix} \xrightarrow{\sigma} \begin{pmatrix} 1 & 2 & \cdots & n \\ b_{\sigma(1)} & b_{\sigma(2)} & \cdots & b_{\sigma(n)} \end{pmatrix} = \beta'.$$

In other words, if the car with preference b_i parks in spot s_i, then index s_i of β' will have number b_i at that index. The σ above is the unique permutation s^{-1}.

Intuitively, parking order records where the car with a certain preference parks. The advantage in arranging a parking function in parking-order is that the new parking function has the same displacement partition as the original one. Furthermore, the new parking-ordered parking function is as close as possible to ascending order without sacrificing the displacement partition. We formalize these statements with the following lemma.

Lemma 2.2. *Let $\alpha = (a_1, \ldots, a_n) \in PF_n$ have displacement partition $\lambda = \lambda_1 + \cdots + \lambda_k$. Then, let α' be the parking rearrangement of α. Then, α' is a parking function that has the same displacement partition λ as α. Furthermore, α' is in parking order.*

Proof. By Definition 2.1, α' is a permutation of the elements of α so α' is a parking function of length n.

Displacements occur when $a_i \neq s_i$. Thus, each $i \in [n]$ either contributes 0 displacement or λ_i displacement, as recorded by λ. Because σ in Definition 2.1 keeps s_i and a_i in the same column for all $i \in [n]$, $s_i - a_i = s_{\sigma(i)} - a_{\sigma(i)}$ for all $i \in [n]$. This shows that α' has the same displacement partition as α.

We now show that α' is in parking order. Notice that it is impossible for a car with preference a_i to park in a spot which is less than a_i. Hence, $a_i \leqslant s_i$ for all $i \in [n]$. Because the columns stay together in the rearrangement of α under σ, we have that $a_{\sigma(i)} \leqslant i$ or that $a'_i \leqslant i$, proving that α' is in parking order. ☐

Lemma 2.3. *Let α and β be two parking functions. Then the relation $\alpha \sim \beta$ defined by "α and β have the same parking rearrangement" is an equivalence relation on parking functions.*

Proof. Since each car in a parking function parks in only one spot, the process of rearranging a parking function into parking order is well-defined, so $\alpha \sim \alpha$ for all parking functions α. Symmetry and transitivity are immediate.

☐

Lemma 2.2 and Lemma 2.3 can be summarized in the following proposition.

Proposition 2.4. *Parking rearrangements partition the set of parking functions into classes that all have the same displacement partition.* ☐

As stated above, parking-ordered parking functions are as close as possible to weakly increasing order without sacrificing the displacement partition. Other reorderings often fail to preserve the displacement partition. However, the following theorem, the proof of which can be found in [16], shows us that any reordering will preserve the total displacement.

Theorem 2.5 (c.f. [16])**.** *Total displacement is preserved under the symmetric group action, i.e. if $a \in PF_n$ has $D(a) = k$, then $D(\sigma(a)) = k$ for all $\sigma \in S_n$.*

Example 2.6. The parking function $(1, 1, 2)$ and the permuted version $(1, 2, 1)$ both have total displacements of 2 but in the first parking function two cars are displaced by 1 spot each, whereas the second parking function only has a single car being displaced by 2.

Remark 2.7. From Lemma 2.3, we can see that a sufficient condition on whether a parking function's displacement partition is preserved under σ is when α and $\sigma(\alpha)$ have the same parking rearrangement. However this condition is not necessary since $(1, 1, 3, 2, 2)$ and $(1, 2, 2, 1, 3)$ both share the displacement partition $(3, 2, 1)$ whilst having different parking orders (both are already in parking order).

Directly from the definition for parking-ordered, we can count the number of parking-ordered parking functions. The condition $a_i \leqslant i$ implies that at every index i we have i values to choose, so we have a total of $n!$ ways to form a parking-ordered parking function of length n, which is codified in the following proposition.

Proposition 2.8. *The number of parking-ordered parking functions of length n is $|PF'_n| = n!$.* □

3 Prime Decomposition of Parking Functions

Recall that a prime parking function is a parking function which remains a parking function after any instance of 1 is removed. One can verify that an equivalent definition of prime parking functions is that if α is arranged into increasing order β, then $b_1 = 1$ and $b_i < i$ for all $2 \leqslant i \leqslant n$ in β. Denote by PPF_n the set of all prime parking functions of length n. The principal result in this section is that classical parking functions can be partitioned into disjoint prime parking functions, which is captured in Theorem 3.1. The advantage of this point of view is that the problem of studying displacement in parking functions can be reduced to the same study within the setting of prime parking functions.

This following result shares some similarities with a remark from Gessel in [10] that every increasing parking function can be uniquely decomposed into prime parking functions. However, we utilize the parking rearrangement when decomposing a parking function into its prime parking functions because the structure of the displacement partition of a parking function from this point of view is completely characterized by its non-trivial component primes.

Theorem 3.1. *Let $\alpha \in PF'_n$ be a parking-ordered parking function of length n. Then α can be decomposed uniquely into prime parking functions whose displacement determines the displacement partition of α.*

Proof of Theorem 3.1. Let $\alpha = (a_1, \ldots, a_n) \in PF'_n$ be a parking-ordered parking function of length n. First we partition α into blocks that correspond to prime parking functions which we will call the **component primes**. Each block is a maximal subsequence $(a_i, \ldots a_r)$ such that $a_i = i$, $a_j \geqslant a_i$ for all $j \geqslant i$, and $a_{r+1} = r + 1$ (except when $r = n$).

In parking order the ith car c_i parks in spot i. Thus, if α has blocks $A_1, A_2, \ldots, A_k$ then cars with preferences in A_j only park in $|A_j|$ consecutive spots following the spots occupied by cars with preference in previous blocks.

Since α is in parking-order, cars indexed within A_j will only park in some spot within the indices of A_j. Thus, displacement occurs only within these blocks.

We will now prove that each block in α is a shifted prime parking function. Let $A_j = (a_{j_1}, \ldots, a_{j_r})$ be a block in α with r elements. Then $a_{j_1} = j_1$ and for any $i \in \{j_1 + 1, \ldots, j_r\}$ either $a_i < i$ or if $a_i = i$ then there is some $\ell > i$ such that $a_\ell < a_i$

Arrange A_j into weakly increasing order and denote this by $B_j = (b_{j_1}, \ldots, b_{j_r})$. Then, the above implies that $b_i < i$ for all $i \in \{j_1 + 1, \ldots, j_r\}$ and $b_{j_1} = a_{j_1} = j_1$. Altogether this implies A_j is a prime parking function P_j when each entry is shifted by subtracting by $j_1 - 1$. $\square$

Definition 3.2. For finite tuples of integers $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$ we will write $B + k$ to mean $(b_1 + k, \ldots, b_n + k)$ and we denote the **pipe** $A|B$ as the append of A with $B + m$ where $|A| = m$. E.g. for $A = (1, 2, 1)$ and $B = (1, 1, 4, 3, 2)$, $A|B = (1, 2, 1, 4, 4, 7, 6, 5)$.

Remark 3.3. Although the hypothesis of Theorem 3.1 requires that α be a parking-ordered parking function, any parking function can be split into its component prime parking functions. Take any parking function $\alpha \in PF_n$ and change it into its parking rearrangement α' by the canonical permutation σ_α (the inverse of the outcome permutation). Then, perform the process outlined in Theorem 3.1 to identify the component prime parking functions $P_1, \ldots, P_k$ (these are the appropriately shifted blocks from the proof above, now thought of as prime parking functions). Once the prime parking functions have been stratified, apply σ_α^{-1} to $P_1|P_2|\ldots|P_k$ (The pipe of these prime parking functions) to recover the original ordering of the parking function, but now with the component prime parking functions.

Notice that the pipe $P_1|\ldots|P_k$ is the same integer tuple as $(A_1, \ldots, A_k)$ where A_i is a block defined in the proof of Theorem 3.1. We will often abuse our notation to refer to these A_i's as component primes even though the actually prime parking function is the shifted version P_i.

Moreover, the proof of Theorem 3.1 proves that cars with preferences in one block will not park in another block. In tandem with the previous paragraph, this implies that displacement for any parking function only occurs *within* its component primes.

Example 3.4. Let $\alpha = (4, 2, 6, 1, 1, 5, 4)$. Following the process outlined in Theorem 3.1, $\alpha' = (1, 2, 1, 4, 5, 6, 4)$. Decomposing α' into shifted primes:

$$\alpha' = (1, 2, 1 , 4, 5, 6, 4),$$

where the values highlighted in red correspond to the first prime parking function $P_1 = (1, 2, 1)$ while the values highlighted in teal correspond to the second prime parking function, $P_2 = (1, 2, 3, 1)$. Then, rearranging α into its original order, we obtain

$$\alpha = (4 , 2 , 6 , 1, 1 , 5, 4).$$

We can conveniently visualize how and where α splits into its component prime parking functions using labeled Dyck paths. Recall that there is a bijection between parking functions of length n and labeled Dyck paths from $(0, 0)$ to (n, n) which remain above the $x = y$ line (c.f. [6]). Figure 1 as seen below is the Dyck path associated with α.

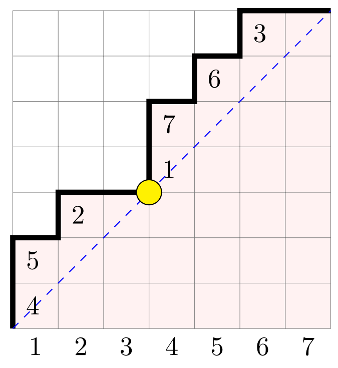

Figure 1: The Dyck Path associated with α.

We claim that exactly where the Dyck Path hits the $x = y$ is where α splits. According to the bijection, vertical steps at index i occur for each instance of preference i in the parking function. A horizontal step from column i to $i + 1$ occurs when there are no more possible vertical steps at column i or, in other words, when there are no more preferences of i to count. Therefore, the shape of the Dyck path only depends on the value of the preferences in the parking function, not their position. Then, arranging α into its parking rearrangement α' should not change the shape of its associated Dyck path. Then, the Dyck path hits the $x = y$ line at $(1, 1)$ when there is only one 1, at $(2, 2)$ when there are only 2 elements from the set $\{1, 2\}$, and so on. In general, the Dyck path hits the $x = y$ line at (k, k) when there are only k elements from the set $\{1, \ldots, k\}$ in α. This is equivalent to conditions 1 and 2 from Theorem 3.1, so α splits where its Dyck path hits the $x = y$ line.

Notice that α has 3 elements from the set $\{1, 2, 3\}$, so it hits the $x = y$ line at $(3, 3)$. This also implies that a new prime parking function begins at index 4, which agrees with the conditions in Theorem 3.1.

The "returns" to the line $x = y$ correspond exactly to the idea of a breakpoint as defined in [12].

Definition 3.5. For a parking function $\alpha \in \mathrm{PF}_n$, b is a **breakpoint** for α if $|\{i \mid \alpha_i \leqslant b\}| = b$.

Remark 3.6. With this definition in mind, it is clear that the number of component primes for a parking function coincides with the number of breakpoints. Also we observe that the shift that turns a block A_i from the proof of Theorem 3.1 to a prime parking function P_i is exactly the value of the breakpoint immediately preceeding A_i.

The ability to split parking functions into its component prime parking functions is crucial in proving many other intriguing facts. First, we use Theorem 3.1 to enumerate the number of parking-ordered prime parking functions, denoted by PPF_n'. We require the following definition to enumerate PPF_n'.

Definition 3.7. A permutation $\sigma = (\sigma_1, \ldots, \sigma_n) \in S_n$ is **decomposable** if there exists an $i \in [n]$ such that every element in $(\sigma_1, \ldots, \sigma_i)$ is less than every element in $(\sigma_{i+1}, \ldots, \sigma_n)$. A permutation σ is **indecomposable** if σ is not decomposable. Denote the set of indecomposable permutations of length n by SI_n. Connected, irreducible, and indecomposable are used interchangeably.

Example 3.8. Let $\sigma = (3,1,2,5,6,4) \in S_6$ be a permutation of size 6. Then σ is decomposable because $(3,1,2 \mid 5,6,4)$ splits σ where everything to the left of the bar is less than everything to the right. Let $\tau = (3,5,6,1,4,2) \in S_6$. Then τ is indecomposable as there is no $i \in [6]$ where everything to the left of and including index i is less than everything to the right.

Theorem 3.9. *The number of parking-ordered prime parking functions of length n is equal to the number of indecomposable permutations of length n. In other words, $|PPF_n'| = |SI_n|$.*

Proof. We prove this using a combinatorial argument. Let p_n count the number of parking-ordered prime parking functions of length n. We will count the number of non-prime parking-ordered parking functions, so let $\alpha = (a_1, \ldots, a_n)$ be a parking function of this kind. By Theorem 3.1, there exists an index $k+1$ which is the last breakpoint of α. Then, $(a_1, \ldots, a_k)$ is some parking-ordered parking function, and $(a_{k+1}, \ldots, a_n)$ is a parking-ordered prime parking function. By Proposition 2.8, there are $k!$ possibilities for $(a_1, \ldots, a_k)$ and there are p_{n-k} possibilities for $(a_{k+1}, \ldots, a_n)$. Furthermore, $k+1$ records where α splits so k can range anywhere from 1 to $n-1$. Thus, the number of non-prime parking-ordered parking functions is

$$\sum_{k=1}^{n-1} k! p_{n-k}.$$

Because we want to count prime parking-ordered parking functions, we obtain the recurrence relation

$$p_n = n! - \sum_{k=1}^{n-1} k! p_{n-k}.$$

There is only one prime parking-ordered parking function of length 1, which is (1). Hence, the initial value for this recurrence is $p_1 = 1$. According to [5] , this recurrence relation is the exact same as that for indecomposable permutations, proving that $|PPF_n'| = |SI_n|$. □

Corollary 3.9.1. *The enumeration for the number of parking-ordered prime parking functions of length n may be obtained from the generating function*

$$1 - \frac{1}{\sum_{k=0}^{\infty}(k!x^k)}$$

Proof. The recurrence in Theorem 3.9 is the subject of the sequence OEIS A003319, and the desired generating function is given in [8]. □

Prime parking functions are the building blocks of all displacement in parking functions, which suggests that it is useful to consider how displacement arises in prime parking functions. We begin with bounds on the length of a prime parking function for a fixed displacement partition. The following lemma provides an upper bound on the length of a prime parking function based only on its total displacement.

Proposition 3.10. *The longest possible length for a prime parking function α with total displacement $D(\alpha) = k$ is $k+1$. Furthermore, the number of prime parking functions α with total displacement $D(\alpha) = k$ and length $k+1$ is*

$$\frac{(k+1)!}{2}.$$

Proof. First, we show that a prime parking function α with a total displacement of k and length $k+1$ exists. Let

$$\alpha = (1, 2, 3, \ldots, k-1, k, 1).$$

The vector α has length $k+1$ and has a total displacement of k by the final 1 in the sequence. No other car contributes to the displacement as the first k cars form a permutation on $[k]$. Moreover, α is a prime parking function: arranging α in nondecreasing order yields

$$\beta = (1, 1, 2, 3, \ldots, k-1, k),$$

which satisfies the property that $\beta_1 = 1$ and $\beta_i < i$.

We now need to show that $k+1$ is in fact the maximal length for a prime parking function with total displacement k. For the sake of contradiction, suppose there exists a prime parking function $\alpha = (a_1, \ldots, a_\ell)$ with total displacement k and length $\ell > k+1$. Arranging α in nondecreasing order yields

$$\beta = (a_{\sigma(1)}, \ldots, a_{\sigma(\ell)}).$$

Because α is a prime parking function, $a_{\sigma(i)} < i$ for $2 \leqslant i \leqslant \ell$ and $a_{\sigma(1)} = 1$. In addition, total displacement is invariant under permutation by Theorem 2.5, so β has total displacement k. However, the condition that $a_{\sigma(i)} < i$ for $2 \leqslant i \leqslant \ell$ in conjunction with $a_{\sigma(1)} = 1$ implies that each of those $a_{\sigma(i)}$ must experience a displacement of at least 1. Because there are $\ell - 1$ preferences from spots 2 through ℓ, each displaced by at least 1, the total displacement is at least $\ell - 1 > k$, a contradiction. Therefore, the longest prime parking functions with displacement k are of length $k+1$.

We now show that the number of those maximal length prime parking functions with total displacement k is $\frac{(k+1)!}{2}$. We will do so by showing that a prime parking function has length $k+1$ and total displacement k if and only if it is a permutation of $[k]$ with a 1 inserted at any index.

Let α be a permutation of $[k]$ over $k+1$ spots with an extra 1 inserted at index i for $1 \leqslant i \leqslant n+1$. If we order α in nondecreasing order, then

$$\beta = (1, 1, 2, 3, \ldots, k-1, k)$$

which clearly satisfies the conditions for being a prime parking function as specified in the beginning of the proof. Furthermore, α has length $k+1$ and has a total displacement of k.

Let $\alpha = (a_1, a_2, \ldots, a_{k+1})$ be a prime parking function with length $k+1$ and total displacement k. Arranging α into nondecreasing order, we get that

$$\beta = (a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(k+1)}).$$

By the prime parking function condition, $a_{\sigma(1)} = 1$ and $a_{\sigma(i)} < i$ for $2 \leqslant i \leqslant k+1$. The restriction that the total displacement is k then prompts the claim that $a_{\sigma(1)} = 1$ and $a_{\sigma(i)} = i - 1$ for all $2 \leqslant i \leqslant k+1$. Suppose otherwise, so there exists an i from 2 through $k+1$ such that $a_{\sigma(i)} = j$ for some $1 \leqslant j < i - 1$. Then the preference $a_{\sigma(i)}$ will contribute $i - j > 2$ to the total displacement. Each other element of β must contribute at least 1 to the total displacement by the prime parking function condition. Therefore, the total displacement must be at least $2 + (k-1) \cdot 1 = k+1$, a contradiction to the fact that the total displacement must be k.

Hence, a prime parking function has length $k+1$ and total displacement k if and only if it is a permutation of $[k]$ with a 1 inserted at any index. The number of such prime parking functions is

$$\frac{(k+1)!}{2}$$

as there are $k+1$ elements in $[k]$ with the extra 1, but there are two 1s, so the division by 2 accounts for the repetition. $\square$

To provide a lower bound on the length of a prime parking function with a fixed displacement partition, the following lemma will be necessary.

Lemma 3.11. *For a fixed displacement partition $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_k$, a minimal length parking function with displacement partition λ can be attained with the displacement terms arising in nondecreasing order in the parking process. In other words, there's a parking function of minimal length where the first car to be displaced is displaced by λ_k, the second displaced car is displaced by λ_{k-1}, and so on, until the last displaced car is displaced by λ_1.*

Proof. First, suppose two cars are being displaced, one by $\ell > 0$ and one by $k > \ell$. In order to achieve a displacement of k, at least k slots must already be occupied. Similarly, the displacement of ℓ requires at least ℓ slots to be already occupied. Hence it's optimal to have the $\ell+1$ cars (contributing to and including the car displaced by ℓ) comprise the slots that build up the displacement for k. Indeed, this approach produces a parking function of total length $k+1$, whereas if the displacement of k came first, an additional entry to produce displacement ℓ would have to be appended, requiring $k+1+1 > k+1$ entries. In general, this reasoning shows that it's optimal to have smaller displacements absorbed in the necessary build up of larger displacements, which is guaranteed with displacements occur in nondecreasing order. Hence a parking function following this process, at each step only adding terms necessary to achieve the next largest displacement, achieves minimal length. $\square$

Now we can provide a lower bound on the length of a prime parking function with a fixed displacement partition.

Theorem 3.12. *Let α be a prime parking function of length n with displacement partition $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_k$ with $k \geqslant 1$. Then the follow algorithm provides a lower bound on n.*

- *In step 1, initialize a counter variable with initial value λ_k+1*
- *At step $i > 1$, let c denote the value of the counter variable. If $c \geqslant \lambda_i$, increment c by 1. Otherwise, set c equal to $\lambda_i + 1$. Repeat for a total of k steps*
- *Return the value of the counter variable*

The lower bound is attained.

Proof. We claim that after after step i, the value of the counter records the length of a minimum length of a parking function with displacement partition equal to the i rightmost values of λ, which are $\lambda_{k-i+1}, \lambda_{k-i+2}, \ldots, \lambda_k$. Let's prove this claim by induction on i.

For the base case $i = 1$, suppose $\lambda_k = v$. The minimum length parking function with displacement partition λ_k is a permutation of $[v]$ followed by a 1, which has length $v+1$. Since the value of the counter variable is initially set to $\lambda_k + 1 = v + 1$ in the first step, the base case holds.

For the inductive step, we assume the value of the counter c after i steps records the minimum length parking function with displacement partition $\lambda_{k-i+1}, \lambda_{k-i+2}, \ldots, \lambda_k$. There are the two cases: $c \geqslant \lambda_{n-i}$, or not. For the first case, $c \geqslant \lambda_{n-i}$. By Lemma 3.11, we know that the minimal length parking function on the rightmost $i+1$ values can be attained with displacement λ_i occurring last, so its total length is at least one more than the length of the minimal parking function on the i rightmost terms. The minimal parking function β_i on the i rightmost values of λ requires $c \geqslant \lambda_{n-i}$ spots, so another added car will park in slot $c+1$. Now, notice that appending the value $c+1-\lambda_i > 0$ to β_i preserves the previous displacements and adds a displacement term of λ_{n-i}. Since this parking function is obtained by adding exactly one entry to β_i, the resulting parking function must be of minimal length to displace the rightmost $i+1$ terms.

Now suppose $c < \lambda_{n-i}$. Again, by Lemma 3.11, we know that there's minimal length parking function on the rightmost $i+1$ values can with displacement λ_i occurring last. Since there are less than λ_{n-i} slots occupied in the minimal length parking function β_i that displaces the rightmost i terms, and producing a displacement λ_{n-i} requires at least λ_{n-i} occupied slots, we can't produce a displacement of λ_{n-i} by adding only one term. Therefore adding the required $\lambda_{n-i}-c$ terms $c+1$ to λ_{n-i}, and then one more car with preference 1 and hence displacement λ_{n-i}, maintains the previous i displacements and minimally adds displacement λ_{n-i}. In this case, there are $\lambda_{n-i}+1$ occupied slots, which gives the claimed value of the counter variable after $i+1$ steps. This parking function must be minimal on the rightmost $i+1$ terms, since any parking function with a displacement term λ_{n-i} has at least $\lambda_{n-i}+1$ terms, and the constructed parking function above has length exactly $\lambda_{n-i}+1$. This completes the induction.

The lower bound in the lemma follows, since after the algorithm terminates, the value of the counter records the minimum length of a parking function with displacement partition λ, which is at least the minimum length of a prime parking function with displacement partition λ.

Also observe that the previous inductive proof provides a construction of the minimal-length parking function, which is also prime. Let's show this by induction on the number of steps of the algorithm. For the base case, a permutation followed by a 1 is prime. Now consider the inductive case. When $c < \lambda_{n-i}$, the algorithm appends a 1 to a parking function that, by the inductive hypothesis, is prime, which produces another prime parking function. One can verify that given a prime parking function of length v, appending any value other than $v+1$ produces another prime parking function of length $v+1$. Hence when $c \geqslant \lambda_{n-i}$, the parking function produced by the algorithm remains prime after appending $c+1-\lambda_i < c+1$, completing the inductive step. Hence the lower bound is attained. □

4 Direct Enumerations

4.1 A Single Displaced Car

Theorem 4.1. *The number of parking functions of size n with a displacement partition of $\lambda = k > 0$ is*

$$\frac{n!}{k+1}(n-k).$$

Proof. If the displacement partition of $\alpha \in PF_n$ is $\lambda = k$, then only one car, say car i with preference $a_i \in [n]$, is displaced, and it's displaced by k. Since k is the only part of the partition—and all displacement in a parking function occurs within a parking function's prime components by Remark 3.3—α must have a single nontrivial prime component which contributes the displacement term k. Every car outside of this prime component parks without displacement.

If the displaced car has preference a_i, then in order to have a displacement of k, the k spots $a_i, a_i+1, \ldots, a_i+k-1$ must already be occupied before car i parks. Since only car i is displaced, each of these k cars must park without displacement, so there are no repeated preferences in this nontrivial prime component before car i parks. Therefore, α's nontrivial prime component is of the form $a_i, a_i+1, \ldots, a_i+(k-1), a_i$.

We will now count the number of parking functions subject to the constraint in the previous paragraph. The number of ways to choose the starting value a_i is $n-k$ as a_i can be any value in $[n-k]$. The number of ways to place the values $a_i, a_i+1, \ldots, a_i+(k-1)$ in the n spots is $\binom{n}{k+1}$. The number of ways to permute these values is $k!$. Finally, the number of ways to permute the remaining values of the parking function that must park at their preference is $(n-(k+1))!$. This results in

$$(n-k)\binom{n}{k+1}k!\,(n-(k+1))! = \frac{n!}{k+1}(n-k)$$

as the total number of ways to count the number of parking functions with displacement partition k. □

The fact that a prime parking function with displacement partition $\lambda = k > 0$ has length $n = k+1$ gives the following corollary.

Corollary 4.1.1. *The number of prime parking functions of length n with displacement partition $\lambda = k > 0$ is*

$$\left\{ \begin{array}{ll} k! & \text{for } n = k+1, \\ 0 & \text{else} \end{array} \right\}$$

□

Remark 4.2. Notice that when $k = 1$ in Theorem 4.1, the formula we recover gives the *Lah numbers*[1]

$$\frac{n!}{2}(n-1),$$

matching the result obtained in [2].

We present further connections with parking functions with one car displaced to other enumerative formulae, such as the second-order Eulerian numbers defined below.

Definition 4.3. The **Second-order Eulerian numbers** $A(n,m)$ count the permutations of the multiset $\{1,1,2,2,\ldots,n,n\}$ with m ascents which have the property that for each k, all the numbers appearing between the two occurrences of k in the permutation are greater than k. As shown in [11], they satisfy the recurrence relation

$$A(n,m) = (2n-m-1)A(n-1,m-1) + (m+1)A(n-1,m).$$

Remark 4.4. Notice that $A(n,n) = 0$ for any integer n. This follows from the observation that any permutation of the multiset $\{1,1,2,2,...,n,n\}$ that satisfies the second condition in Definition 4.3, when restricted to consecutive integers $i, i+1$, looks like one of $\{i,i,i+1,i+1\}$, $\{i,i+1,i+1,i\}$, and $\{i+1,i+1,i,i\}$, which have at most one ascent. Since there are $n-1$ pairs of consecutive integers, the permutation can have at most $n-1$ ascents, so $A(n,n) = 0$.

Notice also that $A(n,n-1) = n!$ follows from the given recurrence relation since $A(1,0) = 1$ and $A(n,n-1) = nA(n-1,n-2) + nA(n-1,n-1) = nA(n-1,n-2) + 0$.

[1] *Lah numbers* are the focus of OEIS: A001286

Corollary 4.4.1. *The number of parking functions of length n with one car displaced,*

$$\sum_{k=1}^{n-1} \frac{(n-k)n!}{k+1} \tag{1}$$

is equal to the Second-order Eulerian number $A(n, n-2)$.[2]

Proof. The formula is obtained by summing the result of Theorem 4.1 from $k = 1$, the minimum a car can be displaced, to $k = n-1$, the maximum a car can be displaced. We prove the corollary by showing the Second-order Eulerian number $A(n, n-2)$ and the formula have the same initial value and satisfy the same recurrence relation. For the Second-order Eulerian numbers, $A(2, 0) = 1$. The initial value for (1) is when $n = 2$, and indeed $\sum_{k=1}^{1} \frac{(2-k)2!}{k+1} = 1$.

Applying the more general recurrence from [11] to the specific tuple $(n, n-2)$, and following Remark 4.4, we have

$$\begin{aligned} A(n, n-2) &= (n+1)A(n-1, n-3) + (n-1)A(n-1, n-2) \\ &= (n+1)A(n-1, n-3) + (n-1)(n-1)! \end{aligned}$$

Through algebraic manipulations, one can see that

$$\sum_{k=1}^{n-1} \frac{(n-k)n!}{k+1} = (n+1)\sum_{k=1}^{n-2} \frac{((n-1)-k)(n-1)!}{k+1} + (n-1)(n-1)!$$

so the same recurrence is satisfied and the result follows. ☐

4.2 Two Displaced Cars

Now we enumerate classical and prime parking functions of length n where two cars are displaced. When two cars are displaced it is helpful to separate into cases corresponding to the order in which the cars experience displacement.

Definition 4.5. For a parking function α with displacement partition $\lambda = d_1 + d_2 + \cdots + d_m \vdash D(\alpha)$ (where the d_i's are nonincreasing) where m cars are displaced, the **displacement order** is the m-tuple whose i-th entry is equal to the displacement of the i-th displaced car in the parking function. After all cars have parked, reading the displacements of parked cars from left to right gives the displacement order. For example, the parking function $(1, 2, 2, 1, 4, 2)$ has displacement order $(2, 3)$, which is shown visually in Figure 2.

[2]Second-order Eulerian number $A(n, n-2)$ is the focus of OEIS A002538

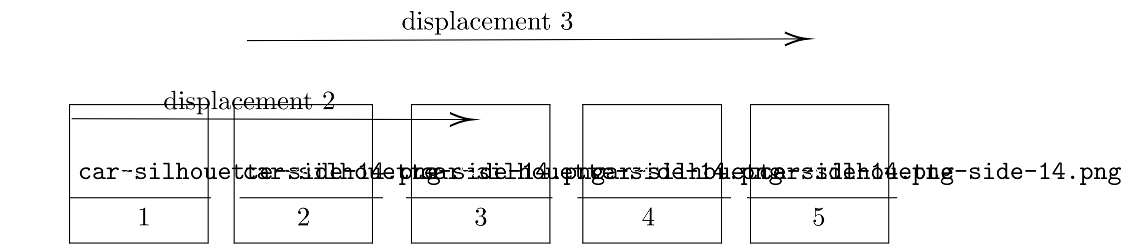

Figure 2: The parking configuration for the parking function $(1, 2, 1, 4, 2)$. Reading from left to right, the first displaced car, c_3, is displaced by 2 and the second displaced car, c_5, is displaced by 3, so the displacement order is $(2, 3)$.

The parking function $(3, 4, 3, 1, 1)$ has displacement order $(1, 2)$ since after all cars have parked, the earliest parking spot taken by displaced car corresponds to a car displaced by 1 and the second smallest slot taken by a displaced car corresponds to displacement 2. Figure 3 visually shows this parking process.

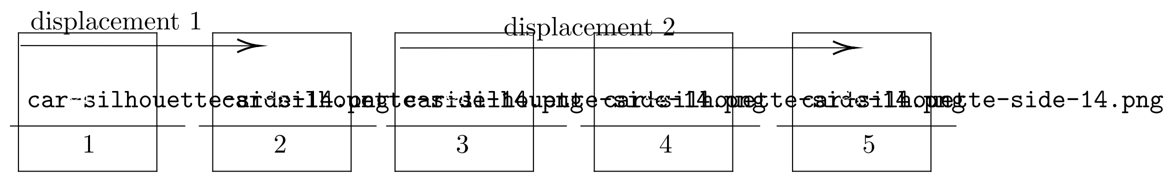

Figure 3: The parking configuration for the parking function $(3, 4, 3, 1, 1)$. The left-right order of the displacement arrows give the displacement order $(1, 2)$.

It is important to observe that the displacement order comes from the left-right order of displaced cars after all cars have parked, rather than the order of displacement terms in time. For instance, in the previous example, car c_3 parks and is displaced before c_5 in the parking process, yet since c_5's occupied slot is before c_3's, the displacement term corresponding to c_5 comes first in the displacement order.

For a displacement partition with m distinct terms, the $m!$ possible displacement orders are mutually exclusive and their union gives all possible ways a parking function can produce the desired displacement partition.

When 2 cars are displaced in a parking function, the displacements either happen in a single prime or in disjoint primes. Since we've already characterized a single displaced car in a prime parking function in Theorem 4.1, we now characterize the possible parking orders that give rise to 2 displaced cars where both displacements happen in a single prime.

For brevity's sake, we introduce a **discrete interval notation** for any sequence of consecutive terms. This should not be confused with the standard continuous interval notation. For any sequence $(a, a + 1, \ldots, a + k)$, we rewrite this as

$$[a, a + k] := (a, a + 1, \ldots, a + k).$$

Lemma 4.6. *Let α be a prime parking function with displacement partition $\lambda = k + \ell$ with $1 \leqslant \ell < k \leqslant n - 1$. Then exactly one of the following is true:*

1. *The displacement order is (ℓ, k), and the parking order of the prime is given by*

$$(\underbrace{[a, a+j-1]}_{j\ terms}, \underbrace{[a+j, a+j+\ell-1]}_{\ell\ terms}, a+j, \underbrace{[a+\ell+j+1, a+k+i-2]}_{k-\ell+i-j-2\ terms}, a+i-1)$$

for integers $0 \leqslant j \leqslant k - \ell - 1$,$1 \leqslant i \leqslant \ell + j + 1$. The prime has length $k + i$.

2. *The displacement order is (k, ℓ), and the parking order of the prime is given by*

$$(\underbrace{[a, a+k-1]}_{k\ terms}, a, \underbrace{[a+k+1, a+k+i-1]}_{i-1\ terms}, a+k-\ell+i)$$

for an integer $1 \leqslant i \leqslant \ell$. The prime has length $k + i + 1$.

Remark 4.7. The variable a in the lemma is just some value in $[n]$ and represents the smallest value in this prime component of the parking function. We use a rather than the number 1 to make clear that in a classical parking function, the smallest value of a prime component may not be 1 until a shift is performed on the component.

Proof of Lemma 4.6. There are two cars displaced, one by k and one by ℓ. Let c_k be the car displaced by k and c_ℓ be the car displaced by ℓ.

In order for c_ℓ to be displaced by ℓ, we must have the subpattern $(a + j, a + j + 1, \ldots, a + j + \ell - 1, a + j)$, where the j accounts for the fact that the subpattern needn't start at the first position a in the prime (where a is the smallest repeated value in the prime, or the number 1 in numbers, up to a shift). Indeed, since the smaller displacement ℓ comes before the larger displacement k, the subpattern producing the displacement ℓ has $k - \ell$ possible "starting values", namely $a, a + 1, \ldots, a + j$, for $0 \leqslant j \leqslant k - \ell - 1$. The j terms a through $a + j - 1$ precede the subpattern producing the displacement of ℓ.

For example, if the displacement order is $(2, 5)$, we need a buildup of 5 cars before any car can be displaced by 5, so we can attain the initial displacement of 2 via 12145, 12325, or 12343 before any car is displaced by 5, giving $5 - 2 = 3$ options.

After the displacement of ℓ, we must produce a displacement of k. Let p denote the preference of c_k, and let $i = p - a + 1$. Since by construction, a is the smallest repeated value in the prime, we have $p \geqslant a$ and so $i \geqslant 1$. Also, to remain a single prime, c_k must pass over some value in the prime subpattern producing displacement ℓ, so we have $p \leqslant a + i + \ell$ and so $i \leqslant \ell + j + 1$. In order for c_k to have displacement k, it must park in slot $p + k = a + i - 1 + k$. Hence c_k passes over slots the $\ell - i + j + 2$ slots p through $a + \ell + j$, as well as the $k + i - \ell - j - 2$ slots $a + \ell + j + 1$ through $a + i + k - 2$, for a combined total displacement of $(\ell - i + j + 2) + (k + i - \ell - j - 2) = k$. So the subpattern $(a + \ell + j +, \ldots, a + i + k - 2)$, which has $k + i - \ell - j - 2$ entries, must precede the final entry. Finally, c_k parks with preference $p = a + i - 1$. Since j can range from 0 to $k - \ell - 1$ and i can range from 1 to $\ell + j + 1$, case 1 follows.

Now suppose that c_k parks before c_ℓ. In order for c_k to be displaced by k, there must be the subpattern $(a, a + 1, \ldots, a + k - 1, a)$. Since both displacements happen in a single prime parking function, c_ℓ parks after c_k and c_ℓ must pass over at least one of the $k + 1$ values in the subpattern $(a, a + 1, \ldots, a + k - 1, a)$. Since $k + 1$ slots are already occupied and $\ell < k$, c_ℓ cannot have a preference before c_k's prime starts. Indeed, since slot $a+k+1$ is the next available slot, c_ℓ's preference p is at least $a+k+1-\ell$, since otherwise c_ℓ would have displacement more than ℓ. Let $i = p-(a+k+1-\ell)+1$, so $i \geqslant 1$. Also, $p \leqslant a+k$ to remain a single prime, so $i \leqslant \ell$. Since c_ℓ has displacement ℓ, it must park in slot $p+\ell = a+k+i$, so the $i-1$ slots $a+k+1$ through $a+k+i-1$ need to be taken before c_k parks, adding the subpattern right before the last entry. Then c_k parks with preference $p = a+k-\ell+i$. Hence case 2 follows.

□

Proposition 4.8. *For a prime parking function α with displacement partition $\lambda = k+k$ where $1 \leqslant k \leqslant n-2$, the parking order of the prime is given by*

$$(\underbrace{[a, a+k-1]}_{k\ terms}, a, \underbrace{[a+k+1, a+k+i-1]}_{i\text{-}1\ terms}, a+i)$$

for an integer $1 \leqslant i \leqslant k$. The length of the parking order of the prime is $k+1+i$.

Proof. There are two cars displaced, both by k. Let c_1 denote the first car displaced, and let c_2 denote the second. By assumption, both displacements happen in a single prime.

In order for c_1 to be displaced by k, there must be the length $k+1$ subpattern $(a, a+1, \ldots, a+k-1, a)$, which results in the first $k+1$ slots being occupied. Suppose that c_2 has preference $a+i$. Since both displacements happen in a single prime, c_2 parks after c_1 and c_2 must pass over at least one of the $k+1$ values in the previous subpattern, so $i \leqslant k$. However, we note that c_2 cannot have preference a, since in this case it would park in slot $a+k+1$ and would have displacement $k+1 > k$, so the earliest preference that c_2 can has is $a+1$. Hence $1 \leqslant i \leqslant k$. In order for c_2 to have displacement k, it must park in slot $a+i+k$, which means that the $i-1$ slots $a+k+1$ through $a+i+k-1$ must be occupied before c_2 parks. Hence for $i>1$, the subpattern $(a+k+1, \ldots, a+i+k-1)$ must precede the final entry $a+i$. After c_2 parks, slots a through $a+i+k$ are all occupied, so the pattern has length $k+1+i$. Since i can range from 1 to k, the lemma follows. □

Remark 4.9. When two cars are displaced, the pattern of the displacement partition $\lambda = k+k$ is identical to the pattern of the displacement partition $\lambda = k+\ell, \ell < k$ with displacement order (k, ℓ), after replacing ℓ with k. The final preference for the latter is is $a+k-\ell+i$, which after substituting k for ℓ, becomes $a+i$ which is the final preference of the former.

Given the patterns that produce a displacement partition, then counting the rearrangements of each pattern that preserve the displacement partition gives the total number of parking functions in question.

Theorem 4.10. *The number of parking functions of length n with displacement partition $\lambda = k+\ell$, where $1 \leqslant \ell < k \leqslant n-1$ is given by*

$$\frac{n!(n-k)(k-\ell)}{(k+1)(\ell+1)} + \sum_{i=2}^{\ell+1} \frac{n!(n-k-i+1)}{(\ell+1)(k+i)} + \sum_{i=1}^{\ell} \frac{n!(n-k-i)}{(k+i+1)(k+1)}$$
$$+2k!\ell!\binom{n-(k+\ell)}{2}\binom{n}{k+1}\binom{n-(k+1)}{\ell+1}(n-(k+\ell+2))!$$

for $n \geqslant k+\ell+2$.

Proof. We can isolate the disjoint cases by Lemma 4.6 and enumerate them separately. For each case, we determine how parking functions of length n contain the pattern, accounting for the ways that valid rearrangements of certain letters in the pattern can happen.

Case 1. Both displacements happen in a single prime and the displacement order is (ℓ, k).
Following the notation of the lemma, assume $1 \leqslant i \leqslant \ell + 1$.

Subcase (i): $i = 1$
First, consider $i = 1$. We note that the length of the pattern is $k + 1$, producing a factor of $(n - k)$ to account for the ways to choose $a \in [n - k]$. Also, there are

$$\binom{n}{k+1}(n-(k+1))! = \frac{n!}{(k+1)!}$$

ways to select the $k + 1$ slots in the parking function of length n that the pattern will occupy and arrange the remaining $n - (k + 1)$ entries outside of the pattern (which must be a permutation of $[n - k - 1]$ in order to avoid contributing any more displacement).

For the case $i = 1$, there may be different ways to choose the "starting value" of the subpattern that generates the ℓ displacement, corresponding to the choice of j in Lemma 4.6.

There must be some subpattern of length $\ell + 1$ that contributes the ℓ displacement. There are k letters before the last a (the very last entry in the pattern), so there are $k - \ell$ "starting values" for the subpattern that produces the ℓ displacement (in the previous example, the starting values were a and $a + 1$). This introduces a factor of $(k - \ell)$. There are

$$\binom{k}{\ell+1}(k-(\ell+1))! = \frac{k!}{(\ell+1)!}$$

ways to select the slots in the pattern occupied by the subpattern producing ℓ, and arrange the remaining entries (which can be placed in any order, since none of them experience any displacement) before the final a. Finally, since all of the first ℓ entries in the subpattern producing displacement ℓ park in their preferred slots, we can arrange these first ℓ entries in any order, giving a factor of $\ell!$.

Combining gives that for $i = 1$, the number of parking functions from case 1 is given by

$$\frac{n!k!\ell!(n-k)(k-\ell)}{(k+1)!(\ell+1)!} = \frac{n!(n-k)(k-\ell)}{(k+1)(\ell+1)}$$

Subcase (ii): $i \geqslant 2$
First, notice that for $i > 1$, the "starting value" of the subpattern producing displacement ℓ must be a, since otherwise the entries before the "starting value" correspond to cars that are neither displaced nor under the influence of a displaced car, and hence would themselves be trivial prime parking functions and would not be part of the single prime that contributes the displacements. Hence we can assume $j = 0$.

We note that the length of the pattern is $k + i$, producing a factor of $(n - k - i + 1)$ to account for the ways to choose $a \in [n - k - i + 1]$. Also, there are

$$\frac{n}{k+i}(n-(k+i))! = \frac{n!}{(k+i)!}$$

ways to select the $k + 1 + i$ slots in the parking function of length n that the subpattern will occupy and arrange the remaining $n - (k + 1 + i)$ entries outside of the pattern (which must be a permutation of $[n-k-1-i]$ in order to avoid contributing any more displacement). Also, we notice that the first ℓ terms before the first displaced car can be arranged in any order, which can happen in $\ell!$ ways. In addition, the $k-\ell+i-2$ terms between the first displaced car and the last entry are all parking without being displaced, so they can be placed anywhere in that region of $(k-\ell+i-2)$ entries, or before. Hence there are $\ell+1+k-\ell+i-2 = k+i-1$ choices where the first entry in the subpattern can go, $k+i$ choices where the second can go, and so on, ending with $\ell+2$ choices where the last entry in the region can go, which introduces the product $(k+i+1)(k+i)(k+i-1)\dots(\ell+2) = \frac{(k+i+1)!}{(\ell+1)!}$.

Combining and summing over the values of i (which represent disjoint patterns) gives that for $i > 1$, the number of parking functions from subcase 2 is given by

$$\sum_{i=2}^{\ell+1} \frac{n!\ell!(k+i-1)!(n-k-i+1)}{(\ell+1)!(k+i)!} = \sum_{i=2}^{\ell+1} \frac{n!(n-k-i+1)}{(\ell+1)(k+i)}$$

Case 2. The displacement order is (k, l), so the first car is displaced by k. Following the notation of the lemma, assume $1 \leqslant i \leqslant \ell$. Observe that we don't have to split up into subcases based on the "starting values" from case 1. This choice of "starting value" from the previous case arose because the displacement ℓ came first, and since $\ell < k$, we needed to add extra cars before the final a in order to build up the displacement of k, which gave extra room to produce the ℓ displacement using the extra cars. But in this case, since $k > l$ and the displacement of k is coming first, there is no extra room to shift the starting value. So consider arbitrary $1 \leqslant i \leqslant \ell$.

The length of the pattern is $k+1+i$, so there are $n-k-i$ choices for $a \in [n-k-i]$. Similarly to the reasoning in case 1, there are $\frac{n!}{(k+1+i)!}$ ways to select the indices in the parking function that the pattern will occupy, and arrange the entries outside of the parking function. Also, we notice that the first k terms before the second a can be arranged in any order, which can happen in $k!$ ways. Finally, the $i-1$ terms between the second a and the last entry are all parking without being displaced, so they can be placed anywhere in that region of $(i-2)$ entries, or before. Hence there are $k+1+i-1 = k+i$ choices where the first entry in the subpattern can go, $k+i-1$ choices where the second can go, and so on, ending with $k+2$ choices where the $k-\ell+i-2$-th entry can go, which introduces the product $(k+i)(k+i-1)(k+i-2)\dots(k+2) = \frac{(k+i)!}{(k+1)!}$.

Combining and summing over the values of i (which represent disjoint patterns) gives that the number of parking functions from case 2 is given by

$$\sum_{i=1}^{\ell} \frac{n!k!(k+i)!(n-k-i)}{(k+i+1)!(k+1)!} = \sum_{i=1}^{\ell} \frac{n!(n-k-i)}{(k+i+1)(k+1)}$$

Case 3. Disjoint Primes

In this case, the primes are disjoint. Within each nontrivial prime, the structure exactly coincides with the case in which one car was displaced. The parking order of that structure is completely determined by the starting value of the prime component and the displacement within that prime. Thus, if one prime displaces a single car by k and the other prime displaces a single car by ℓ, the choice for starting values of both components (a and b, respectively) is given by $2\binom{n-(k+l)}{2}$, or the number of ways to pick $\ell+1$ and $k+1$ sized blocks of consecutive integers from $[n]$. This indicates the slots that cars with preferences within these primes will park.

Then, we can choose the spots for $(a, a+1, \ldots, a+k-1, a)$ in $\binom{n}{k+1}$ ways, and the spots for $(b, b+1, \ldots, b+\ell-1, b)$ in $\binom{n-(k+1)}{\ell+1}$ ways. Moreover, there are $k!$ ways to order $(a, a+1, \ldots, a+k-1)$ before the repeated a, and there are $\ell!$ ways to order $(b, b+1, \ldots, b+\ell-1)$ before the repeated b. Finally, the number of ways to permute the entries in the parking function that are not in either prime is $(n-(k+\ell+2))!$. Combining gives that the number of parking functions from case 3 is given by

$$\binom{n-(k+\ell)}{2}\binom{n}{k+1}\binom{n-(k+1)}{\ell+1}2k!\ell!(n-(k+\ell+2))!$$

The longest case is the disjoint one, which requires that $n \geqslant k+\ell+2$, hence this enumeration is well-defined for $n \geqslant k+\ell+2$. $\square$

The following corollary is useful for our later enumerations.

Corollary 4.10.1. *For either of the possible displacement orders, (ℓ, k) or (k, ℓ), the number of parking functions of length n with displacement partition $\lambda = k+\ell$ where $1 \leqslant \ell < k \leqslant n-1$ and $n \geqslant k+\ell+2$, where the two displaced cars belong to disjoint primes and with the chosen displacement order, is*

$$k!\ell!\binom{n-(k+\ell)}{2}\binom{n}{k+1}\binom{n-(k+1)}{\ell+1}(n-(k+\ell+2))!$$

Proof. We provide a finer count of the disjoint case of Theorem 4.10. All steps of the enumeration from the previous proof are the same, except fixing a displacement order amounts to fixing the relative order between a and b (with a and b defined in Theorem 4.10). Indeed, swapping a and b in the previous count, and the resulting factor of 2, accounts for the fact that after having chosen the slots occupied by the two subpatterns, we can have either one have a smaller initial value, which is equivalent to swapping the displacement order. So we remove the factor of 2 and the corollary follows.

$\square$

We can easily convert an enumeration of parking functions of length n with a fixed displacement partition to an enumeration of prime parking functions of the displacement partition, by substituting the length of the prime for n.

Corollary 4.10.2. *The number of prime parking functions with displacement partition $\lambda = k+\ell$ where $1 \leqslant \ell < k \leqslant n-1$ is given by*

$$\left\{\begin{array}{lr} \frac{k!(k-\ell)}{\ell+1}, & \textit{for } n = k+1, \\ \frac{(k+i-1)!}{k+1} + \frac{(k+i-1)!}{\ell+1}, & \textit{for } n = k+i, 2 \leqslant i \leqslant \ell+1 \\ 0 & \textit{else} \end{array}\right\}$$

Proof. We know that prime parking functions are parking functions whose prime decomposition is itself. Hence we can use Lemma 4.6 to characterize the cases that the prime parking function can take and Theorem 4.10 to count them. We take the cases from the theorem corresponding to a single prime, and include only terms from the count that account for ways to permute the pattern within the prime. Equivalently, we can set n to be equal to the length of the prime.

A single prime of length $k+1$ can only happen in a single way, corresponding to the $i=1$ subcase of case 1 in Lemma 4.6. Setting $n=k+1$, the first row of the corollary follows.

For $2 \leqslant i \leqslant \ell+1$, we note that the prime of length $k+i$ can occur in two ways: with displacement ℓ first, substituting i into the theorem, or with displacement k first, substituting $i-1$ into the theorem (since the length of the pattern in this case is $k+j+1$ for $1 \leqslant j \leqslant \ell$). Substituting $n=k+i$ for this range of i, the rest of the corollary follows. □

Remark 4.11. Corollary 4.2 can easily be converted into an enumeration of the number of parking functions with displacement partition $\lambda = k+\ell$ for $k+1 \leqslant n \leqslant k+\ell+1$, which completes the enumeration of Theorem 4.10 for all values of n. The disjoint case is impossible for $k+1 \leqslant n \leqslant k+\ell+1$, so the number of parking functions with displacement partition $k+\ell$ with length in this range are given by the number of prime parking functions in Corollary 4.2. The rearrangement term is found from the same reasoning as in the proof of Theorem 4.10, corresponding to the number of ways to choose a, the ways to choose the slots taken by the prime subpattern, and the ways to rearrange outside terms, all of which only depend on the length of the nontrivial prime. This process converts the above enumeration of prime parking functions to the enumeration of parking functions for $k+1 \leqslant n \leqslant k+\ell+1$.

Theorem 4.12. *The number of parking functions with displacement partition $\lambda = k+k$, where $1 \leqslant k \leqslant n-2$, is given by*

$$\sum_{i=1}^{k} \frac{n!(n-k-i)}{(k+i+1)(k+1)} + (k!)^2(n-(2k+2))!\binom{n-2k}{2}\binom{n}{k+1}\binom{n-(k+1)}{k+1}$$

for $n \geqslant 2k+2$.

Proof. By Remark 4.9, the pattern of the displacement $\lambda = k+k$ is the same as displacement $\lambda = k+\ell$, where $1 \leqslant \ell < k$ with displacement order (k,ℓ), after replacing ℓ with k. Hence after substituting k for ℓ in the term from Theorem 4.10 and term in Corollary 4.10.1 corresponding to displacement order (k,ℓ), we have the desired enumeration.

□

Corollary 4.12.1. *The number of prime parking functions with displacement partition $\lambda = k+k$ for $1 \leqslant k \leqslant n-2$ is given by*

$$\begin{Bmatrix} \frac{(k+i)!}{k+1}, & \textit{for } n = k+1+i, 1 \leqslant i \leqslant k \\ 0 & \textit{else} \end{Bmatrix}$$

Proof. As in Corollary 4.10.2, we can evaluate the enumeration in Theorem 4.12 with n equal to the length of the prime that contributes the displacements.

□

Remark 4.13. Analogously to Remark 4.11, the above enumeration can easily be used to determine the number of parking functions with displacement partition $\lambda = k+k$ of length n, where $k+2 \leqslant n \leqslant 2k+1$. This completes the enumeration of Theorem 4.12 for all n.

4.3 Three Displaced Cars

Now we characterize the prime parking functions with 3 displaced cars and enumerate prime parking functions of length n with a fixed displacement partition as a sum over each possible displacement order. First, we prove a lemma to help with the enumeration.

Lemma 4.14. *In any prime parking function of length n, the first $n-1$ entries form a parking function (not necessarily prime) of length $n-1$. The last car must be displaced from its preference α_n, say by k spots, and must park in the n-th spot. Thus, $n = \alpha_n + k$.*

Proof. Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ be a prime parking function of length n and $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ be a rearrangement into weakly increasing order. Since α is prime, we know that $\beta_1 = 1$ and $\beta_i < i$ for all $1 < i \leqslant n$. Suppose α_n is sent to β_k for some $k \in [n]$. Let $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_{n-1})$ be the same rearrangement β but with the β_k removed, i.e. $\gamma = (\beta_1, \ldots, \widehat{\beta_k}, \ldots, \beta_n)$. In this new arrangement, it is evident that from index k and beyond, $\gamma_i < i+1$ so that $\gamma_i \leqslant i$. Indeed γ satisfies the criterion of being a classical parking function, as desired.

If the entire parking function is prime, then $\alpha_i < n$ for all i, so the preference of every car, including the last car one, is less than n. Since the first $n-1$ entries form a parking function, the first $n-1$ slots are occupied before the last car parks, so the last car must park in slot n and is displaced. Hence $n = \alpha_n + k$.

□

In the following observation we make the upper bound on the largest displacement term explicit, so that it doesn't have to be explicitly written in the rest of the subsection.

Remark 4.15. For displacement partition $\lambda = k + \ell + m$ where $1 \leqslant m < l < k$, we have $k < n$ since $n-1$ is the maximum displacement of the third displaced car. For displacement partition $\lambda = k + \ell + \ell$, where $1 \leqslant \ell < k$ we have $k < n$ also. For displacement partition $\lambda = k + k + \ell$ where $1 \leqslant \ell < k$, we have $k < n-1$, and for displacement partition $\lambda = k + k + k$ where $k \geqslant 1$, we have $k < n-2$. These upper bounds on the largest displacement term are assumed throughout the rest of the subsection.

We now introduce the idea of strongly and weakly grouped prime parking functions in order to exploit separate properties of each in our enumerations.

Definition 4.16. Suppose a prime parking function of length n has $m > 1$ displaced cars. Let $(i_1, i_2, \ldots, i_m)$ denote the indices in the parking function of the m displaced cars. If for any $1 \leqslant j \leqslant m$, the first i_j entries of the parking function form a prime parking function (of length i_j), then the original prime parking function is **strongly grouped**. In other words, a strongly grouped prime parking function has that every time a car is displaced, the displacement overlaps the prime of the previous displaced car. One can observe that for strongly grouped prime parking functions, the displacement order is equal to the order that displacements happen in time (the i-th car displaced in the parking function is also the displaced car that parks in the i-th smallest spot among all displaced cars). If a prime parking function is not strongly grouped, call it **weakly grouped**. In particular, when there are 3 displaced cars, a prime parking function is strongly grouped if and only if after removing the final entry, the remaining parking function is prime.

For example, the prime parking function $(1,2,3,1,2,6,5)$ is strongly grouped. The displacements occur at indices 4, 5, and 7, and the subsequences $(1,2,3,1)$,$(1,2,3,1,2)$, and $(1,2,3,1,2,6,5)$ are all prime. After removing the final entry, the resulting parking function $(1,2,3,1,2,6)$ is prime.

Figure 4 shows where each car c_i parks and represents displacements as line segments from preferred slot to final parking spot.

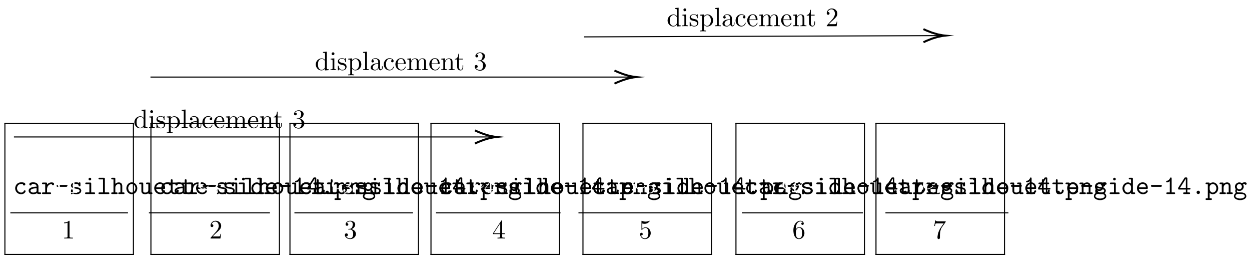

Figure 4: The parking configuration for the parking function $(1, 2, 3, 1, 2, 6, 5)$. This parking function is strongly grouped, as every displacement (the displacement line segments above the cars, which represent nontrivial prime factors) overlaps the previous one.

An example of a weakly grouped prime parking function is $(1, 2, 3, 1, 4, 5, 5, 2)$. There is a displacement at index 7, but the length 7 subsequence $(1, 2, 3, 1, 4, 5, 5)$ is not prime. Indeed, after removing the final entry, the resulting parking function $(1, 2, 1, 4, 5, 4)$ is not prime. Figure 5 visually represents the parking process.

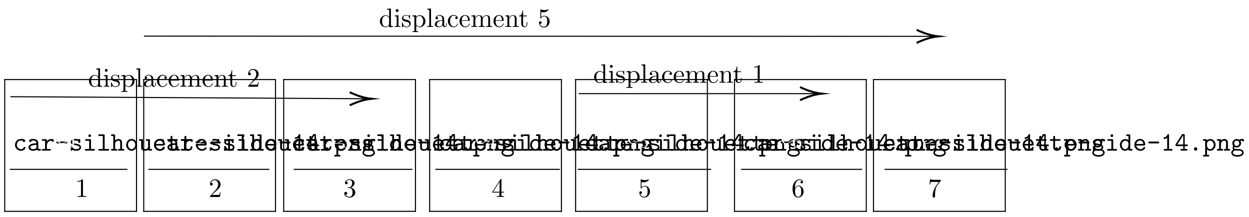

Figure 5: The parking configuration for the parking function $(1, 2, 3, 1, 4, 5, 5, 2)$. This parking function is weakly grouped: before the last car parks, the first two displacement segments (nontrivial prime factors) are disjoint. The second displacement does not overlap the first.

With the definition of strongly/weakly grouped prime parking functions in mind, the following lemma is useful to characterize the possible displacement orders of a weakly grouped prime parking function.

Lemma 4.17. *A weakly grouped prime parking function with displacement order (a, b, c) has $c \geqslant b + 2$.*

Proof. Let c_f denote the car displaced by c, c_b the car displaced by b, and c_a the car displaced by a. Since c_a and c_b belong to disjoint primes before c_f parks, there are $v \geqslant 0$ additional spaces from the end of c_a's prime and before the start of c_b's. By Lemma 4.14, the entries before c_f parks form a parking function, and since the entire parking function becomes prime after c_f parks, c_f must intersect c_a's prime and pass over all values in c_b's prime. Hence even if c_f's preference is the last entry in c_a's prime, it must pass over $1 + v + b + 1 = b + v + 2 \geqslant b + 2$ slots. Hence since c_f has displacement c, we must have $c \geqslant b + 2$. □

Corollary 4.17.1. *For weakly grouped prime parking functions with* 3 *displaced cars:*

1. *If the displacement partition is* $\lambda = k + \ell + m$ *where* $1 \leqslant m < \ell < k$*, the displacement order must be* $(m, \ell, k), (\ell, m, k)$*, or* (k, m, ℓ)*.*
2. *If the displacement partition is* $\lambda = k + \ell + \ell$*, where* $\ell < k$*, the displacement order must be* (ℓ, ℓ, k)*.*
3. *If the displacement partition is* $\lambda = k + k + \ell$*, where* $\ell < k$*, the displacement order must be* (k, ℓ, k)*.*
4. *No weakly grouped prime parking functions with displacement partition* $\lambda = k + k + k$ *exist.*

Proof. A weakly grouped prime parking function must have that the penultimate entry in the displacement order is strictly less than the last entry in order to satisfy Lemma 4.17. □

Using Lemma 4.14, which shows that the length of a prime parking function is completely determined by the final preference in the prime parking function and the value of the final displacement, we can find the possible lengths of weakly grouped prime parking functions.

Lemma 4.18. *Suppose a prime parking function has a car* c_m *displaced by* m*, a car* c_ℓ *displaced by* ℓ*, and after both, a car* c_k *displaced by* k*.*

If the displacement order is (m, ℓ, k)*, then there exists a weakly grouped prime parking function of length* L *if and only if* $\max(\ell + m + 2, k) \leqslant L - 1 \leqslant k + m$*.*

Proof. First, before c_k parks, by Lemma 4.14, the entries form a parking function. Since the prime parking function is weakly grouped, c_m and c_ℓ belong to disjoint primes before c_k parks. Hence there are at least $(m + 1) + (\ell + 1)$ slots in the parking function before c_k parks.

Note that with a parking function of length $m+\ell+2$ slots, a car that then parks with preference 1 can only be displaced $m + \ell + 2$ slots. Hence if $k > m + \ell + 2$, more slots must be added, namely those up to and including the k-th slot, which would produce a parking function of length k before c_k parks. Hence the length L of the parking function before c_k parks is at least $\max(\ell + m + 2, k)$. This lower bound is attained when there are no extra spaces between the first two cars' primes.

Now let's consider the largest possible length of the prime parking function after c_k parks. Since the first two displaced cars c_m and c_ℓ belong to disjoint primes before c_k parks, c_k must intersect the first displaced car's prime and pass over the second displaced car's prime. Then, if k is sufficiently large, extra slots will need to be added after the second displaced car's prime in order to produce displacement k. To maximize the length of the total prime parking function, the preference of c_k should be at the last slot occupied by the first disjoint prime. Since c_m's prime is to the left of c_ℓ's, the length of the total prime parking function is maximized (and attained) when c_k has preferred spot $m + 1$, in which case the length of the total prime parking function is $k + 1$ (the k slots c_k passes over and the one slot it parks in) plus the first m slots in c_m's prime. Hence the length L of the parking function before c_k parks is $(k + m + 1) - 1 = k + m$. Hence the reverse direction follows. All other values of L satisfying the inequality can be attained by inserting additional spaces between the first and second cars' primes and c_m having the same preference as in the minimal case, completing the forward direction. □

Removing the last entry of a weakly grouped prime parking function produces a parking function with 2 cars that are displaced disjointly. Based on the possible lengths for this disjointly-grouped parking function, computed above, and the previous enumeration of disjointly-grouped parking functions with 2 displaced cars, we count the total number of weakly grouped prime parking functions in the following lemma.

Lemma 4.19. *The number of weakly grouped prime parking functions with displacement partition $\lambda = k + \ell + m$, where $1 \leqslant m < \ell < k$, is given below.*

Case 1. The displacement order is (m, ℓ, k). If $k \geqslant \ell + 2$, the count is given by

$$\sum_{n=\max(\ell+m+2,k)}^{k+m} \ell!m!\binom{n-(\ell+m)}{2}\binom{n}{m+1}\binom{n-(m+1)}{\ell+1}(n-(\ell+m+2))!$$

Case 2. The displacement order is (ℓ, m, k). If $k \geqslant m + 2$, the count is given by

$$\sum_{n=\max(\ell+m+2,k)}^{k+\ell} \ell!m!\binom{n-(\ell+m)}{2}\binom{n}{m+1}\binom{n-(m+1)}{\ell+1}(n-(\ell+m+2))!$$

Case 3. The displacement order is (k, m, ℓ). If $\ell \geqslant m + 2$, the count is given by

$$\sum_{n=k+m+2}^{k+\ell} k!m!\binom{n-(k+m)}{2}\binom{n}{m+1}\binom{n-(m+1)}{k+1}(n-(k+m+2))!$$

There are no weakly grouped primed parking functions with any other displacement order.

Proof. First, the three displacement orders above give the only possible displacement orders to consider, by Corollary 4.17.1.

Consider the first two cases, where the last car that is displaced is displaced by k. By Lemma 4.14, the last entry in the parking function must be the entry that contributes the k displacement. Also, by the same lemma, the value of k and the final preference completely determine the length of the the prime parking function. Hence if the prime parking function has length $n + 1$, then the first n elements represent a parking function with 2 cars being displaced disjointly. Since Lemma 4.18 gives all possible values of the length of the parking function before the final preference, and by Lemma 4.14, we know that for each n in the sum, the final preference overlaps the first prime (hence giving a larger prime parking function), ranging over all values of n and summing via Corollary 4.10.1 gives exactly the desired count.

Notice that the bounds and summand are also well-defined. The last term of the summand requires that $n \geqslant \ell+m+2$, which is satisfied by the lower index range in the sum. For displacement order (m, ℓ, k), using Lemma 4.17, the upper range of the index $k + m$ has that $k + m > k$ and $k + m \geqslant \ell + m + 2$. Analogous reasoning holds for displacement order (ℓ, m, k).

We use the same reasoning as above for the case of displacement order (k, m, ℓ). To show that the summand is well-defined, we must have that $n \geqslant k + m + 2$, which is satisfied by the lower index of the counter variable. To show that the bounds are well-defined, note that $\ell \geqslant m + 2$, so $\ell + k \geqslant k + m + 2$, as needed. ☐

Theorem 4.20. *The number of weakly grouped prime parking functions with* 3 *cars displaced, where at least two displacements are equal, is given below.*

1. *The displacement partition is $\lambda = k + k + \ell$, where $1 \leqslant \ell < k$, the displacement order is (k, ℓ, k), and $k \geqslant \ell + 2$. The count is*

$$\sum_{n=k+\ell+2}^{2k} k!\ell! \binom{n-(\ell+k)}{2} \binom{n}{\ell+1} \binom{n-(\ell+1)}{k+1} (n-(\ell+k+2))!$$

2. *The displacement partition is $\lambda = k+\ell+\ell$, where $1 \leqslant \ell < k$, the displacement order is (ℓ, ℓ, k), and $k \geqslant \ell + 2$. The count is*

$$\sum_{n=\max(2\ell+2,k)}^{k+\ell} (\ell!)^2 \binom{n-2\ell}{2} \binom{n}{\ell+1} \binom{n-(\ell+1)}{\ell+1} (n-(2\ell+2))!$$

For any other displacement orders for displacement partition $\lambda = k + \ell + \ell$ or $\lambda = k + k + \ell$, and for displacement partition $\lambda = k + k + k$, there are no weakly grouped prime parking functions.

Proof. By Corollary 4.17.1, the above two cases are the only ones that can have a weakly grouped prime parking function with 3 displaced cars and at least two equal displacement terms. The counts follow from the same reasoning as Lemma 4.19. □

Having enumerated weakly grouped prime parking functions, we now consider strongly grouped prime parking functions with 3 displaced cars. First we characterize the patterns that can give rise to a strongly grouped prime parking functions with a fixed displacement partition.

Lemma 4.21. *For a strongly grouped prime parking function, α, with displacement partition $\lambda = k + \ell + m$, where unless otherwise stated, $1 \leqslant m < \ell < k$, exactly one of the following is true:*

Case 1. The displacement order is (m, ℓ, k). Fix i where $1 \leqslant i \leqslant m + 1$.

Subcase (i): We have $k \geqslant \ell + i$.
Up to a shift in the "starting value" as in Remark 4.22, the parking order of the prime is given by

$$(\underbrace{[a, a+m-1]}_{m\ terms}, a, \underbrace{[a+m+1, a+i+\ell-2]}_{\ell-m+i-2\ terms}, a+i-1, \underbrace{[a+\ell+i, a+k+j-2]}_{k-\ell-i+j-1\ terms}, a+j-1)$$

for some $1 \leqslant j \leqslant \ell + i$. The total length of the prime is $k + j$.

Subcase (ii): We have $k < \ell + i$.
Up to a shift in the "starting value" as in Remark 4.22, the parking order of the prime is given by

$$(\underbrace{[a, a+m-1]}_{m\ terms}, a, \underbrace{[a+m+1, a+i+\ell-2]}_{\ell-m+i-2\ terms}, a+i-1, \underbrace{[a+\ell+i, a+\ell+i+j-2]}_{j-1\ terms}, a+\ell+j+i-k-1)$$

for some $1 \leqslant j \leqslant k$. The total length of the prime is $\ell + i + j$.

Case 2. The displacement order is (m, k, ℓ) with $\ell \leqslant k$. Fix i where $1 \leqslant i \leqslant m + 1$. The parking order of the prime is given by

$$(\underbrace{[a, a+m-1]}_{m\ terms}, a, \underbrace{[a+m+1, a+i+k-2]}_{k-m+i-2\ terms}, a+i-1, \underbrace{[a+k+i, a+k+i+j-2]}_{j-1\ terms}, a+k+i+j-\ell-1)$$

for some $1 \leqslant j \leqslant \ell$. The total length of the prime is $k+i+j$.

Case 3. The displacement order is (ℓ, m, k) where $m \leqslant \ell$. Fix i where $1 \leqslant i \leqslant m$.

Subcase (i): We have $k \geqslant \ell+i+1$.
The parking order of the prime is given by

$$(\underbrace{[a, a+\ell-1]}_{\ell \text{ terms}}, a, \underbrace{[a+\ell+1, a+\ell+i-1]}_{i-1 \text{ terms}}, a+\ell-m+i, \underbrace{[a+\ell+i+1, a+k+j-2]}_{k-\ell-i+j-2 \text{ terms}}, a+j-1)$$

for some $1 \leqslant j \leqslant \ell+i+1$. The total length of the prime is $k+j$.

Subcase (ii): $k < \ell+i+1$
The parking order of the prime is given by

$$(\underbrace{[a, a+\ell-1]}_{\ell \text{ terms}}, a, \underbrace{[a+\ell+1, a+\ell+i-1]}_{i-1 \text{ terms}}, a+\ell-m+i, \underbrace{[a+\ell+i+1, a+\ell+i+j-1]}_{j-1 \text{ terms}}, a+\ell+i+j-k)$$

for some $1 \leqslant j \leqslant k$. The total length of the prime is $\ell+i+j+1$.

Case 4. The displacement order is (ℓ, k, m) where $1 \leqslant m \leqslant \ell$.
The parking order of the prime is given by

$$(\underbrace{[a, a+\ell-1]}_{\ell \text{ terms}}, a, \underbrace{[a+\ell+1, a+i+k-2]}_{k-\ell+i-2 \text{ terms}}, a+i-1, \underbrace{[a+k+i, a+k+i+j-2]}_{j-1 \text{ terms}}, a+k+i+j-m-1)$$

for some $1 \leqslant j \leqslant m$. The total length of the prime is $k+i+j$.

Case 5. The displacement order is (k, ℓ, m) with $m \leqslant \ell \leqslant k$. Fix i where $1 \leqslant i \leqslant \ell$.
The parking order of the prime is given by

$$(\underbrace{[a, a+k-1]}_{k \text{ terms}}, a, \underbrace{[a+k+1, a+k+i-1]}_{i-1 \text{ terms}}, a+k-\ell+i, \underbrace{[a+k+i+1, a+k+i+j-1]}_{j-1 \text{ terms}}, a+k+i+j-m)$$

for some $1 \leqslant j \leqslant m$. The total length of the prime is $k+i+j+1$.

Case 6. The displacement order is (k, m, ℓ) with $\ell \leqslant k$. Fix $1 \leqslant i \leqslant m$.
The parking order of the prime is given by

$$(\underbrace{[a, a+k-1]}_{k \text{ terms}}, a, \underbrace{[a+k+1, a+k+i-1]}_{i-1 \text{ terms}}, a+k-m+i, \underbrace{[a+k+i+1, a+k+i+j-1]}_{j-1 \text{ terms}}, a+k+i+j-\ell)$$

for some $1 \leqslant j \leqslant \ell$. The total length of the prime is $k+i+j+1$.

Remark 4.22. The patterns above assume for clarity that when the first two entries of the displacement order are increasing, that the overall prime's smallest entry is a, the preference of the first displaced car. We account for instances where this is not the case in our enumerations, with what we call "starting values".

Proof of Theorem 4.21. See Appendix. □

As in the case of 2 displaced cars, we enumerate the number of strongly grouped prime parking functions with 3 displaced cars from the number of rearrangements of each pattern that preserve the displacement partition.

Theorem 4.23. *The number of strongly grouped prime parking functions with displacement partition $m+\ell+k$, where $1 \leqslant m < \ell < k$ unless otherwise stated, and a fixed displacement order, is given below.*

Case 1. The displacement order is (m,ℓ,k).

$$\sum_{j=1}^{\ell+1} \frac{(\ell-m)(k+j-1)!}{(m+1)(\ell+1)} + \sum_{i\in S_1}\sum_{j=1}^{\ell+i} \frac{(k+j-1)!}{(m+1)(\ell+i)} + \sum_{i\in S_2}\sum_{j=1}^{k} \frac{(\ell+i+j-1)!}{(m+1)(\ell+i)}$$

where $S_1 = [2, \max(m+1, k-\ell)]$ and $S_2 = [2, \max(m+1, k-\ell-1)]$.

Case 2. The displacement order is (m,k,ℓ) with $\ell \leqslant k$.

$$\sum_{j=1}^{\ell} \frac{(k-m)(k+j)!}{(m+1)(k+1)} + \sum_{j=1}^{\ell}\sum_{i=2}^{m+1} \frac{(k+i+j-1)!}{(m+1)(k+i)}$$

Case 3. The displacement order is (ℓ,m,k) where $m \leqslant \ell$.

$$\sum_{i\in S_1}\sum_{j=1}^{\ell+i+1} \frac{(k+j-1)!}{(\ell+1)(\ell+i+1)} + \sum_{i\in S_2}\sum_{j=1}^{k} \frac{(\ell+i+j)!}{(\ell+1)(\ell+i+1)}$$

where $S_1 = [1, \max(m, k-\ell-1)]$ and $S_2 = [1, \max(m, k-\ell-2)]$.

Case 4. The displacement order is (ℓ,k,m) where $1 \leqslant m \leqslant \ell$.

$$\sum_{j=1}^{m} \frac{(k-\ell)(k+j)!}{(\ell+1)(k+1)} + \sum_{j=1}^{m}\sum_{i=2}^{\ell+1} \frac{(k+i+j-1)!}{(\ell+1)(k+i)}$$

Case 5. The displacement order is (k,ℓ,m) with $m \leqslant \ell \leqslant k$.

$$\sum_{j=1}^{m}\sum_{i=1}^{\ell} \frac{(k+i+j)!}{(k+1)(k+i+1)}$$

Case 6. The displacement order is (k,m,ℓ) with $\ell \leqslant k$.

$$\sum_{j=1}^{\ell}\sum_{i=1}^{m} \frac{(k+i+j)!}{(k+1)(k+i+1)}$$

Remark 4.24. For displacement orders that end in the largest displacement term k, there are two separate summands, corresponding to whether or not additional cars are required to be added to attain displacement k. For displacement orders whose first two entries are increasing, the term corresponding to $i = 1$ is separated for terms corresponding to $i > 1$, based on consideration of "starting values".

Proof of Theorem 4.23. See Appendix. □

Now we combine enumerations for strongly and weakly grouped prime parking functions to count the number of prime parking functions of fixed length n with a fixed displacement order.

Proposition 4.25. *The number of prime parking functions of length n with displacement partition $k+\ell+m$ with certain displacement orders below is:*

Case 1. The displacement order is (k,m,ℓ) and $k+m+2 \leqslant n-1 \leqslant k+\ell$ and $k+3 \leqslant n \leqslant k+\ell+m+1$, where $\ell \geqslant m+2$.

$$k!m!\binom{n-1-(k+m)}{2}\binom{n-1}{m+1}\binom{n-1-(m+1)}{k+1}(n-1-(k+m+2))!+\sum_{i\in S_1}\frac{(n-1)!}{(k+1)(k+i+1)}$$

where $S_1 = [\max(1, n-(k+1)-\ell), \min(m, n-(k+1)-1)]$.

Case 2. The displacement order is (m,k,ℓ) with $\ell \leqslant k$ and $k+2 \leqslant n \leqslant k+\ell+m+1$.

$$\frac{(k-m)(n-1)!}{(m+1)(k+1)} * \mathbb{1}_c + \sum_{i\in S_2}\frac{(n-1)!}{(m+1)(k+i)}$$

where $S_2 = [\max(2, n-k-\ell), \min(m+1, n-k-1)]$ and $\mathbb{1}_c = 1$ when $n-k-\ell \leqslant 1$ and $\min(m+1, n-(k+1)) \geqslant 1$ and is 0 otherwise.

Case 3. The displacement order is (ℓ,k,m) with $m \leqslant \ell$ and $k+2 \leqslant n \leqslant k+\ell+m+1$.

$$\frac{(k-\ell)(n-1)!}{(\ell+1)(k+1)} * \mathbb{1}_d + \sum_{i\in S_3}\frac{(n-1)!}{(\ell+1)(k+i)}$$

where $S_3 = [\max(2, n-k-m), \min(\ell+1, n-k-1)]$ and $\mathbb{1}_d = 1$ when $n-k-m \leqslant 1$ and $\min(\ell+1, n-(k+1)) \geqslant 1$ and is 0 otherwise.

Case 4. The displacement order is (k,ℓ,m) with $m \leqslant \ell \leqslant k$ and $k+3 \leqslant n \leqslant k+\ell+m+1$.

$$\sum_{i\in S_4}\frac{(n-1)!}{(k+1)(k+i+1)}$$

Where $S_4 = [\max(1, n-(k+1)-m), \min(\ell, n-(k+1)-1)]$.

Proof. Consider the first case, when the displacement order is (k,m,ℓ). First let's consider the strongly grouped prime parking functions. The length of the pattern of this displacement order is $k+i+j+1$ where $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant \ell$. We have $i \geqslant 1$ and since $j \leqslant \ell$, we have $i \geqslant n-(k+1)-\ell$. Similarly, we have $i \leqslant m$ and since $j \geqslant 1$, we have $i \leqslant n-(k+1)-1$. Then as $n = k+i+j+1$, substituting $n-(k+1)-i$ for j gives the first term.

Now let's consider weakly grouped prime parking functions. Recall that in this case, the first $n-1$ entries form the disjoint $k+m$ displacement with displacement order (k,m), so their count is given by Lemma 4.19 using $n-1$ instead of n. The final statement in the first case follows from Remark 4.9.

The other cases follow from analogous reasoning, but in the cases after the first, only strongly grouped parking functions are possible.

As Remark 4.24 says, when the displacement order is increasing in its first two entries, we must break up the cases $i = 1$ and $i > 1$ separately. In the cases two and three above, the first term corresponds to the contribution from $i = 1$ if $i = 1$ is a valid value of i for this choice of n. In these cases, the indicator variable on the first summand is 1. If $i = 0$ is not a valid value of i for this value of n, then the indicator variable is 0 and the first summand is excluded from the count. $\square$

We complete the enumeration for displacement orders ending with k.

Proposition 4.26. *For prime parking functions with 3 cars displaced and final entry k in its displacement order, the number of prime parking functions of length n with displacement partition $k + \ell + m$ is given below.*

Case 1. The displacement order is (m, ℓ, k) and n satisfies $\max(\ell + m + 2, k) \leqslant n - 1 \leqslant k + m$, *and* $k \geqslant \ell + 2$.

$$\ell!m!\binom{n-1-(\ell+m)}{2}\binom{n-1}{m+1}\binom{n-1-(m+1)}{\ell+1}(n-1-(\ell+m+2))!+$$
$$\frac{(\ell-m)(n-1)!}{(m+1)(\ell+1)} * \mathbb{1}_{1\in S_1} + \sum_{i\in S_1|i>1}\frac{(n-1)!}{(m+1)(\ell+i)} + \sum_{i\in S_2}\frac{(n-1)!}{(m+1)(\ell+i)}$$

where S_1 and S_2 are the discrete intervals

$$S_1 = [\max(1, n-k-\ell), \min(m+1, k-\ell)]$$

$$S_2 = [\max(n-k-\ell, k-\ell+1), m+1]$$

Case 2. The displacement order is (ℓ, m, k) with $m \leqslant \ell$ and n satisfies $\max(\ell + m + 2, k) \leqslant n - 1 \leqslant k + \ell$.

$$\ell!m!\binom{n-1-(\ell+m)}{2}\binom{n-1}{m+1}\binom{n-1-(m+1)}{\ell+1}(n-1-(\ell+m+2))!+$$
$$\sum_{i\in S_1}\frac{(n-1)!}{(\ell+1)(\ell+i+1)} + \sum_{i\in S_2}\frac{(n-1)!}{(\ell+1)(\ell+i+1)}$$

where S_1 and S_2 are the discrete intervals

$$S_1 = [\max(1, n-k-\ell-1), \min(m, k-\ell-1)]$$
$$S_2 = [\max(n-k-\ell-1, k-\ell), m].$$

Proof. This follows from a combinatorial argument to count both the strongly and weakly grouped possible prime parking functions, considering separately the two subcases for the strongly grouped prime parking functions. $\square$

The above lemmas complete the enumeration of the number of prime parking functions of any length n with 3 displaced cars. Indeed, for a fixed length n, the number of prime parking functions with the fixed displacement partition is given by the sum of the above enumerations for each possible displacement order, including each term (there are 6 total) if and only if n satisfies the conditions of the case in the lemma. If n satisfies only the condition corresponding to the strongly grouped primes, then only include the summand corresponding to those primes.

4.4 Beyond Three Displaced Cars

When counting prime parking functions with 3 displaced cars, enumerating weakly grouped prime parking functions reduces to enumerations of disjoint parking functions (not prime) with 2 displaced cars. In general, a prime parking function with $m > 2$ displaced cars is either weakly or strongly grouped. Weakly grouped prime parking functions, after removing the last entry, revert back to ordinary parking functions with $m - 1$ displacements. Hence a recursive characterization of prime parking functions with a fixed displacement partition depends crucially on understanding strongly grouped prime parking functions since these do not revert back to counts of smaller lengths.

In the following theorem, we give a recursive pattern construction of strongly grouped prime parking functions with a fixed displacement order and any number of displaced cars.

Theorem 4.27. *Suppose a strongly grouped prime parking function α has displacement partition $\lambda = d_1 + d_2 + \cdots + d_m$, where the d_i's are weakly decreasing. Given a pattern of length L that represents the displacement of the first $m - 1$ cars (in any displacement order), where $m > 1$, the pattern of all m cars is below.*

Case 1. The final displaced car to park contributes displacement d_1 and $d_1 \geqslant L$.
Then the total pattern is obtained by appending the subpattern

$$(\underbrace{a + L, a + L + 1, \ldots, a + j + d_1 - 2}_{d_1 - L + j - 1 \text{ terms}}, a + j - 1)$$

for $1 \leqslant j \leqslant L$. The length of the total pattern is $d_1 + j$.

Case 2. The final displaced car to park contributes displacement $d_i \neq d_1$, or the final displaced car to park contributes displacement d_1 but $d_1 < L$.
Then the total pattern is obtained by appending the subpattern

$$(\underbrace{a + L, a + L + 1, \ldots, a + L + j - 2}_{j-1 \text{ terms}}, a + L + j - d_i - 1)$$

for $1 \leqslant j \leqslant d_i$. The length of the total pattern is $L + j$.

Proof of Theorem 4.27. See Appendix A. □

In the following theorem, we use the recursive pattern construction to develop a recursive enumeration for a fixed displacement partition, for any number of displaced cars.

Theorem 4.28. *Suppose a strongly grouped prime parking function α has displacement partition $d_1 + d_2 + \cdots + d_m$ where the d_i's are in strictly decreasing order, for $m > 2$. Choose any $m - 1$ entries in the displacement partition. Given a sum of the form in Theorem 4.23 representing the number of prime parking functions with the displacement partition of $m - 1$ terms (for each of the $(m - 1)!$ possible displacement orders), and length L of the pattern corresponding to the parking of the first $m - 1$ cars, the number of strongly grouped prime parking functions with displacement partition $d_1 + d_2 + \cdots + d_m$ can be obtained as follows:*

Case 1. The final displaced car to park contributes displacement d_1 (so d_1 is the one term excluded from the $m - 1$ preexisting count) and $d_1 \geqslant L$.

For each summand in the sum, add a variable j_m summing from 1 to L. Inside the summand, add the factor

$$\frac{(d_1 + j_m - 1)!}{L!}.$$

Case 2. *Otherwise, let d_i denote the displacement of the car excluded from the count of $m - 1$ cars.*
For each summand in the sum, add a variable j_m summing from 1 to d_i. Inside the summand, add the factor

$$\frac{(L + j_m - 1)!}{L!}.$$

Proof. First suppose that the final displaced car to park contributes a displacement d_1 and that $d_1 \geqslant L$. By Theorem 4.27, we are adding $d_1 - L + j_m - 1$ terms before the final car attempts to park, for variable $1 \leqslant j_m \leqslant d_1$. We notice that each of these $d_1 - L + j_m - 1$ terms are all parking in their desired slots, and so they can be rearranged anywhere before or at their current positions in the pattern. Hence there are $L + (d_1 - L + j - 1) = d_1 + j_m - 1$ choices for the first entry, $d_1 + j_m - 2$ for the second, and so on, ending with $L + 1$ choices for the last. This introduces the product

$$(d_1 + j_m - 1)(d_1 + j_m - 2) \dots (L + 1) = \frac{(d_1 + j_m - 1)!}{L!}.$$

If $j_2, \dots, j_{m-1}$ denote the counter variables in the sum for the chosen $m - 1$ cars, then this new factor must be added inside the sum to account for the fact that each distinct value of the ordered tuple $(j_2, j_3, \dots, j_{m-1}, j_m)$ represents a different prime parking function, and all prime parking functions with displacement partition $d_1 + \dots + d_m$ are counted when all possible values of $(j_2, j_3, \dots, j_{m-1}, j_m)$ are accounted for.

Now suppose that we are not in the previous case, and let d_i denote the displacement of the car excluded from the count of $m - 1$ cars. By Theorem 4.27, we are adding $j_m - 1$ terms before the final car attempts to park, for $1 \leqslant j_m \leqslant d_i$. Similar to the previous case, each of these $j_m - 1$ terms are all parking in their desired slots, and so we can be rearranged anywhere before or at their current positions in the pattern. Hence there are $L + j_m - 1$ choices for the first entry, $L + j_m - 2$ choices for the second, and so on, ending with $L + 1$ choices for the last entry. This introduces the factor

$$(L + j_m - 1)(L + j_m - 2) \cdots (L + 1) = \frac{(L + j_m - 1)!}{L!}$$

into the sum, for the same reasoning as in the first case. □

Example 4.29. We will use Theorem 4.28 to count the number of strongly grouped prime parking functions with four displaced cars and displacement order (m, k, ℓ, d) where $m < \ell < k < d$.

We build the count with 4 displaced cars from the enumeration for the first 3 displaced cars, which is given in Theorem 4.23. By Theorem 4.23, the number of strongly grouped prime prime parking functions with displacement partition $\lambda = k + \ell + m$ and displacement order (m, k, ℓ) is given by

$$\sum_{j=1}^{\ell} \frac{(k-m)(k+j)!}{(m+1)(k+1)} + \sum_{j=1}^{\ell} \sum_{i=2}^{m+1} \frac{(k+i+j-1)!}{(m+1)(k+i)}$$

with pattern length $k + i + j$.

Let's use Theorem 4.28 to count the number of strongly grouped prime parking functions with 4 displaced cars. Since d is the largest displacement term, there are two cases. The strongly grouped primes accounted for in the previous sum, with 3 displaced cars, have length $L = k + i + j$ for some choice of i and j. If $d \geqslant L$, then the number of strongly grouped primes with displacement order (m, k, ℓ, d) is given by

$$\sum_{j=1}^{\ell} \sum_{j_4=1}^{k+j+1} \frac{(k-m)(k+j)!(d+j_4-1)!}{(m+1)(k+1)(k+j+1)!} + \sum_{j=1}^{\ell} \sum_{i=2}^{m+1} \sum_{j_4=1}^{k+i+j} \frac{(k+i+j-1)!(d+j_4-1)!}{(m+1)(k+i)(k+i+j)!}$$

recalling that the first summand corresponds to $i = 1$. Alternatively, if $d < L$, then the count is given by

$$\sum_{j=1}^{\ell} \sum_{j_4=1}^{d} \frac{(k-m)(k+j)!(k+j+j_4)!}{(m+1)(k+1)(k+j+1)!} + \sum_{j=1}^{\ell} \sum_{i=2}^{m+1} \sum_{j_4=1}^{d} \frac{(k+i+j-1)!(k+i+j+j_4-1)!}{(m+1)(k+i)(k+i+j)!}$$

Combining the two cases, we have to make sure that each (i, j) pair is accounted for in the correct case. The number of strongly grouped prime parking functions with displacement order (m, k, ℓ, d) is

$$\sum_{(1,j)\in S_1} \sum_{j_4=1}^{k+j+1} \frac{(k-m)(k+j)!(d+j_4-1)!}{(m+1)(k+1)(k+j+1)!} + \sum_{\substack{(i,j)\in S_1 \\ i\geqslant 2}} \sum_{j_4=1}^{k+i+j} \frac{(k+i+j-1)!(d+j_4-1)!}{(m+1)(k+i)(k+i+j)!}$$

$$+ \sum_{(1,j)\in S_2} \sum_{j_4=1}^{d} \frac{(k-m)(k+j)!(k+1+j+j_4-1)!}{(m+1)(k+1)(k+j+1)!} + \sum_{\substack{(i,j)\in S_2 \\ i\geqslant 2}} \sum_{j_4=1}^{d} \frac{(k+i+j-1)!(k+i+j+j_4-1)!}{(m+1)(k+i)(k+i+j)!}$$

where $S_1 = \{(i, j) \mid i \in [m + 1], j \in [\ell], d \geqslant k + i + j\}$ corresponds to the first case and $S_2 = \{(i, j) \mid i \in [m + 1], j \in [\ell], d < k + i + j\}$ corresponds to the second.

This example shows that the number of strongly grouped prime parking functions with any given displacement partition and displacement order can be enumerated recursively, starting with the known counts with 2 or 3 cars being displaced.

Remark 4.30. Theorem 4.27 gives a recursive way to build the pattern of any number of cars being displaced in a strongly grouped prime parking function for displacement partition that has no equal terms; i.e., no cars have the same displacement. However, the pattern of any number of cars being displaced where the displacement partition has any number of equal terms can be reduced to the criterion for Theorem 4.27 based on Remark 4.9, where the displacement of $k + k$ is equivalent to the displacement pattern of $k + \ell$ with displacement order (k, ℓ), replacing ℓ with k. One strategy is to replace a sequence of equal displacement terms with a descending group of variables, while preserving the relative order between variable terms and outside terms.

As an example, consider the displacement partition $6 + 4 + 4 + 3 + 3 + 3 + 2$. Suppose we want to enumerate the number of prime parking functions with this displacement partition and with displacement order $(4, 6, 3, 3, 4, 3, 1)$. We know that displacement partition $4+4$ is equivalent to the pattern for displacement partition $k_1 + \ell_1$ where $1 \leqslant \ell_1 < k_1$ with displacement order (k_1, ℓ_1). We also know that the displacement partition $3 + 3 + 3$ is equivalent to the pattern for displacement partition $k_2 + \ell_2 + m_2$ where $1 \leqslant m_2 < \ell_2 < k_2$ with displacement order (k_2, ℓ_2, m_2). Hence we can express the total displacement order as $(k_1, 6, k_2, \ell_2, \ell_1, m_2, 2)$ with relations $2 < m_2 < \ell_2 < k_2 < \ell_1 < k_1 < 6$, and find its pattern (and count) with Theorem 4.27, replacing ℓ_1, k_1 with 4 and m_2, ℓ_2, k_2 with 3 in the pattern. Observe that the subsequence of terms in the displacement order corresponding to displacement 3 are descending, and the same is true for terms corresponding to 4.

Hence Theorem 4.28 recursively enumerates the number of strongly grouped prime parking functions with any displacement partition, after performing the above adjustment for repeated displacement terms as needed.

4.5 Prime Parking Functions with Displacement Partition $2 + 2 + \cdots + 2$

Consider prime parking functions where every displaced car is displaced by exactly 2 spots, so the displacement partition is $\lambda = 2 + \cdots + 2$ with $k > 0$ terms. What prime parking functions can produce this displacement partition? The displacement partition $\lambda = 2$ is produced by the prime 121. For the displacement partition $\lambda = 2 + 2$, there are two possibilities: 1212 and 12143. We notice that in the case of 1212, the prime is equivalent to starting from 121 and then appending a single entry, introducing no new undisplaced cars. In the second case, 12143 is equivalent to starting from 121 and then appending two entries, the first of which is an undisplaced car and the second of which gets displaced by 2 spots. We generalize this idea in the following lemma.

Lemma 4.31. *Given a prime parking function p_0 of length $L > 0$ that has displacement partition $2 + 2 + \cdots + 2$, with $k > 0$ terms, there are exactly two prime parking functions with displacement partition $2+2+\cdots+2$, with $k+1$ terms, whose first L entries are p_0. These two primes are given by appending $(L - 1)$ or alternatively the sequence $(L + 1, L)$, to p_0.*

Proof. First, recall that a prime parking function of length L parks in L consecutive spots, it's clear that in either case, the resulting parking function has displacement partition $2 + 2 + \cdots + 2$, with $k + 1$ terms. Also, in both cases the parking function remains prime: the first case case overlaps the prime p_0 at spots $L - 1$ and L, and similarly the second case overlaps the p_0 at spot L.

Since appending any preference that is less than $L-1$ would produce a displacement more than 2, then any possible appended sequence must have all preferences at least $L - 1$. The appended sequence must have some entry overlap p_0 to remain prime, so the possible places of overlap are $L - 1$ and L. Also, the last car in the appended sequence must be displaced (by 2 spots) in order for the parking function to remain prime. The only possibilities are $(L - 1)$ and $(L + 1, L)$. $\square$

With the previous lemma in mind, we can represent all possible parking functions that give rise to a displacement partition $\lambda = 2 + 2 + \cdots + 2$, with any number of terms, using a poset where the prime parking function p_1 is a parent of prime p_2 when p_1 arises from appending a single displacement of 2 to the end p_2. Figure 4.5 shows the first three levels of the poset. In general, the k-th level consists of the prime parking functions with displacement partition $\lambda = 2 + \cdots + 2$ with exactly k terms of 2. In the poset, left edges correspond to appendings of a single entry, while right edges correspond to appendings of two entries.

Given a prime on level k of the poset (produced by $k - 1$ appendings from 121), the unique path from 121 to the prime gives the sequence of appendings that took place. In other words, any

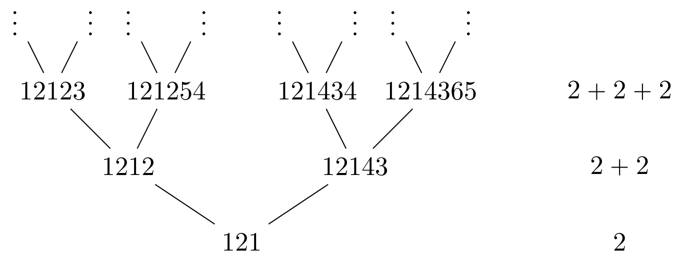

Figure 6: The Hasse Diagram of the poset showing primes with displacement partition $\lambda = 2+\cdots+2$, where the number of terms in the displacement partition is equal to the level on the poset.

prime parking function with displacement partition $2+2+\cdots+2$ with k terms is associated with a unique path from 121 to level k on the poset traversing $k-1$ edges.

We now can use the poset to develop an enumeration of prime parking functions with displacement partition $\lambda = 2+\cdots+2$. Suppose we store the sequence of $k-1$ displacements in a $(k-1)$-tuple $s = (s_1, s_2, \dots, s_{k-1}) \in [2]^{k-1}$ where $s_i = 1$ if the i-th corresponds to a left edge, and 2 if the appending corresponds to a right edge. Let $s' = (3+s_1, 3+s_1+s_2, \dots, 3+\sum_{i=1}^{k-1} s_k)$ be the $(k-1)$-tuple where s'_i denotes the sum of the first i elements of s and 3. Then the length of the prime after i appendings (after 121) is given by s'_i.

Proposition 4.32. *The number of prime parking functions with displacement partition* $2+2+\cdots+2$*, with* $k>1$ *terms, is*

$$\sum_{s\in\{1,2\}^{k-1}} 2 \prod_{\substack{i\in[k-1]\\ s_i=2}} (s'_i - 1)$$

where $s' = (3+s_1, 3+s_1+s_2, \dots, 3+\sum_{i=1}^{k-1} s_i)$ *is a* $(k-1)$*-tuple obtained from* $s \in \{1,2\}^{k-1}$*.*

Proof. As we've already explained, any prime parking function with displacement partition $\lambda = 2+\cdots+2$, with k terms, can be represented uniquely as a path from 121 to level k of the poset in Figure 4.5, with the edges traversed on the path encoding all of the sequences appended.

However, each prime on level k of the poset can be rearranged in more than one way while preserving the displacement partition. Firstly, the initial 121 in the pattern can be swapped to 211: the first two letters can be swapped in $2! = 2$ ways. Suppose the i-th appending is an appending by two entries. By Lemma 4.31, if L denotes length of the prime after $i-1$ displacements, the appended sequence is $(L+1, L)$. Since the first added entry represents a car parking without displacement, its preference can be moved to any of slots at or before its position, while preserving the displacement partition. Since s'_i records the length of the pattern after the appending, the number of possible positions for this undisplaced car is $s'_i - 1$. Accounting for the initial factor of 2 and all such rearrangements in the sequence s, each term in the sum gives the total number of prime parking functions with the desired displacement partition whose appendings are given by the prime s_i on the poset.

Since the position of the displaced car for the appending sequence $(L-1)$, as well as the position of the second added entry in the appending sequence $(L+1, L-1)$ are fixed, there are no ways to rearrange those preferences. Summing over all possible sequences in $\{1,2\}^{k-1}$, or equivalently, all paths from 121 to level k of the poset, gives the result.

□

Since appendings of the form $(L-1)$ add a single new letter to the prime whereas appendings of the form $(L+1, L)$ add two new letters, a sequence $s \in [2]^{k-1}$ completely determines the length of the corresponding prime parking function. Hence we can enumerate primes with a fixed total length.

Corollary 4.32.1. *The number of prime parking functions of length n with displacement partition $2+2+\cdots+2$, with $k > 1$ terms, is*

$$\sum_{\substack{s \in \{1,2\}^{k-1} \\ |\{i | s_i = 1\}| = 2k-n+1}} 2 \prod_{\substack{i \in [k-1] \\ s_i = 2}} (s_i' - 1)$$

for $k+2 \leqslant n \leqslant 2k+1$, and 0 otherwise.

Proof. We only want to consider sequences s with total length n. A sequence s with c_1 entries equal to 1 has $(k-1)-c_1$ entries equal to 2. Since the 1s in s correspond to appendings of a single entry, while 2s in s correspond to appendings of two entries, the total length of the prime corresponding to s is $3 + c_1 + 2(k-1-c_1) = 2k - c_1 + 1$. Hence we require $n = 2k - c_1 + 1$ or equivalently, $c_1 = 2k - n + 1$. The smallest possible length is when $c_1 = k-1$ and the maximum is when $c_1 = 0$. □

4.6 Prime Parking Functions with Displacement Partition $v + v + \cdots + v$, with k terms.

Generalizing the previous subsection, we now consider prime parking functions with displacement partition $v + v + \cdots + v$, with $k \geqslant 1$ terms, where $v > 1$. When $k = 1$, we have a permutation of $[v]$ followed by a 1. We begin by characterizing the possible ways to append new displacements to a preexisting prime.

Proposition 4.33. *Suppose we have a prime parking function p_0 of length $L > 0$ that has displacement partition $v + v + \cdots + v$ where $v > 1$ and with $k > 0$ terms. Then there are exactly v distinct subsequences that can be appended to p_0 to produce a prime parking function with displacement partition $v + v + \cdots + v$ with $k+1$ terms, if all but the last term of any subsequence are placed in increasing order.*

These possible subsequences, when all but the last of its terms are placed in increasing order, have the form $(\underbrace{L+1, L+2, \ldots, L+j}_{j \text{ terms}}, L+j-1-v)$ for some $0 \leqslant j \leqslant v-1$.

Proof. First, any of the possible subsequences in the lemma statement produce a parking function with displacement partition $v + v + \cdots + v$: the subsequence adds j cars and hence and the first $L + j$ spots are occupied before the last one parks, so the final car parks in spot $L + j$ which is 2 spots beyond its preference.

By Lemma 4.14, the last car to park in any prime parking function is displaced and parks in the last spot on the street. Since the sequence introduces exactly one new displaced car, the last entry and only the last entry in the sequence can represent a displaced car and the last car must occupy the last spot on the street. Hence any sequence consists of some number of undisplaced cars occupying a contiguous block of spots starting from $L + 1$, followed by a car displaced by v slots.

If there are j undisplaced cars in the sequence, the final entry must be exactly $L + 1 + j - v$ in order to have displacement v. To remain prime, the final car must overlap p_0, so its preference must be at most L, corresponding to adding $j = v - 1$ undisplaced cars, so $0 \leqslant j \leqslant v - 1$. After arranging the j undisplaced cars' preferences in increasing order, the lemma follows. $\square$

We can construct a poset analagous to the one in Figure 4.5. Each entry has v choices for appending a new displacement. The entries on level k of the poset give all possible patterns (in terms of appending sequences) that produce the displacement partition $v + v + \cdots + v$ with k terms.

For a prime parking function with this displacement partition, let's store the sequence of $k - 1$ displacements after the initial one in a $(k - 1)$-tuple $s = [v]^{k-1}$. The value s_i denotes how many letters were added to produce the i-th displacement after the initial one, so each $s_i \in [v]$. Let $s' = (v + 1 + s_1, v + 1 + s_2, \ldots, v + 1 + \sum_{i=1}^{k-1} s_i)$ be the $(k - 1)$-tuple where s'_i records the length of the pattern after i appendings after the initial one.

Theorem 4.34. *The number of prime parking functions with displacement partition $v + v + \cdots + v$, with k terms, is*

$$\sum_{s \in [v]^{k-1}} v! \prod_{\substack{i \in [k-1] \\ s_i > 1}} \frac{(s'_i - 1)!}{(s'_i - s_i)!}$$

with $s' = (v + 1 + s_1, v + 1 + s_2, \ldots, v + 1 + \sum_{i=1}^{k-1} s_i)$ as previously defined.

Proof. The proof is analogous to the the proof of Proposition 4.32. The only difference is that based on Proposition 4.33, the number of rearrangements for each pattern is now dependent on v instead of being fixed at 2.

The first v terms of the initial displacement in the pattern can be arranged in $v!$ ways. For an appending that adds more than $s_i > 1$ letters, the number of rearrangements of the $s_i - 1$ displaced cars is given by $(s'_i - 1)(s'_i - 2) \ldots (s'_i - s_i + 1) = \frac{(s'_i - 1)!}{(s'_i - s_i)!}$.

$\square$

Analogously to Corollary 4.32.1, a sequence s of $k - 1$ appendings (after the initial one) has each appending i adding s_i letters to the pattern. Hence the total length n of the pattern is given by $n = k + 1 + \sum_{i=1}^{k-1} i \cdot |\{j \mid s_j = i\}|$. As a result, we can enumerate prime parking functions of fixed length with displacement partition $v + v + \cdots + v$, with k terms.

Corollary 4.34.1. *The number of prime parking functions of length n with displacement partition $v + v + \cdots + v$, with k terms, is*

$$\sum_{\substack{s \in [v]^{k-1} \\ \ell(s) = n}} v! \prod_{\substack{i \in [k-1] \\ s_i > 1}} \frac{(s'_i - 1)!}{(s'_i - s_i)!}$$

for $v + k \leqslant n \leqslant vk + 1$, and 0 otherwise. For a sequence $s \in [v]^{k-1}$, s' is defined as in Theorem 4.34, and $\ell(s) = k + 1 + \sum_{i=1}^{k-1} i \cdot |\{j \mid s_j = i\}|$.

Proof. The proof follows from the same reasoning as in Corollary 4.32.1. The smallest possible length of such a prime parking function would be in the case where every s_i equals 1, producing a total length of $v + 1 + (k - 1) = v + k$. The largest possible length would have each s_i equal to v, producing a total length $v + 1 + v(k - 1) = vk + 1$. □

5 Displacement Program

Seeking to generalize results from previous sections, this section describes a method by which one can efficiently count all parking functions of a given length n and displacement partition λ. As parking functions can be partitioned into equivalence classes based on parking rearrangement (Lemma 2.3) and each parking function within an equivalence class has the same displacement partition, we can count the number of parking functions of length n and displacement partition λ by enumerating each equivalence class.

We identify the class representatives as follows. Note that since a parking-ordered parking function α of length n satisfies the condition $a_i \leqslant i$, α is componentwise less than or equal to $(1, 2, \ldots, n)$. This implies that for every parking-ordered parking function α, there exists a unique vector $d = (d_1, \ldots, d_n)$ such that $\alpha + d = (1, 2, \ldots, n)$. For any parking-ordered α, the corresponding vector d is always a valid displacement vector. In fact, this correspondence is a bijection: parking-ordered parking functions of length n are in a one-to-one correspondence with displacement vectors of length n. Therefore, finding the parking-ordered parking functions of length n and displacement partition λ amounts to finding all displacement vectors of length n whose nonzero entries are exactly those in λ. This leads to our first algorithm:

Displacement Vector Algorithm
Input: An integer n and an integer partition λ where $\lambda_i \leqslant n - i$.
Output: The displacement vectors of length n whose nonzero entries are the parts of λ.
Method: Observe that displacement vectors whose nonzero entries are the parts of λ are formed by rearrangements of λ with $n - |\lambda|$ 0's appended to it subject to the constraint that the ith number in the rearrangement is less than i. This observation forms the basis for the algorithm. Note that the first entry will always be 0 because the first car in a parking function cannot be displaced, so we can solely focus on the second through nth entries and append a 0 to the front of each displacement vector at the end of the algorithm.

Start with a list L of $n - |\lambda| - 1$ 0's. Then append a number of 1's equal to the number of 1's in λ to L and find all distinct permutations of L. Next, for each distinct permutation of L, remove the first entry and find all of the ways to insert the 2's in λ into the remaining entries. We must remove the first entry because 2 cannot be the second number in a displacement vector (remember that the first number in any of these permutations will end up being the second number in the displacement vector because we will add a 0 at the beginning). Then reunite each different insertion with its corresponding first entry. Do the same for 3, except remove the first two entries before inserting the 3's and replace both entries at the front after inserting the 3's without changing their order. Continue this process, removing and replacing one extra entry each time until you've inserted the largest part in λ. Finally, add a 0 at the beginning of each vector to obtain all displacement vectors of length n whose nonzero entries are the parts of λ. Thus, the number of displacement vectors are counted by the formula

$$\frac{\prod_{i=1}^{|\lambda|}(n-\lambda_i-i+1)}{\prod_{j=1}^{\lambda_1}(\#\text{ of } j\text{'s in } \lambda)!}.$$

The numerator counts the number of possible ways one can rearrange the nonzero entries in λ among n spots to form a displacement vector; the remaining unoccupied indices are filled by zeroes. For each entry in a displacement vector $d = (d_1, \dots, d_n)$, it must be the case that $d_i \leqslant i-1$ for $1 \leqslant i \leqslant n$. Therefore, λ_1 can only occupy indices $\lambda_1 + 1$ through n of d, allowing for $n - \lambda_1$ possibilities. For each λ_i thereafter, we have $n - \lambda_i - i + 1$ possibilities. Multiplying these together yields the numerator.

The denominator accounts for any repetitions in elements of the displacement partition λ. Since λ_1 is the largest value in an integer partition, we screen every value j between 1 and λ_1, checking if j has multiple repetitions in λ. For every one of these js, we divide by the number of those js factorial to rectify any overcounting from repetitions, which yields the denominator.

The function `displacement_vectors` in Appendix B shows how to explicitly construct displacement vectors of length n with nonzero values matching those in a displacement partition λ.

Example 5.1. Let $n = 6$ and $\lambda = (4, 2, 1, 1)$. We start by finding our initial list of 0's. As $|\lambda| = 4$, $n - |\lambda| - 1 = 1$, so we start with (0). Notice that λ has two 1's, so the distinct permutations with those 1's and the one 0 are

$$(0,1,1) \quad (1,0,1) \quad (1,1,0).$$

For 2, we separate the first entry from the rest of each partial displacement vector as such:

$$(0)(1,1) \quad (1)(0,1) \quad (1)(1,0).$$

Then we list all the ways to insert the 2 into the rightmost portion

$$\begin{array}{ccc} (0)(1,1,2) & (0)(1,2,1) & (0)(2,1,1) \\ (1)(0,1,2) & (1)(0,2,1) & (1)(2,0,1) \\ (1)(1,0,2) & (1)(1,2,0) & (1)(2,1,0), \end{array}$$

and reunite each different insertion with its corresponding first entry:

$$\begin{array}{ccc} (0,1,1,2) & (0,1,2,1) & (0,2,1,1) \\ (1,0,1,2) & (1,0,2,1) & (1,2,0,1) \\ (1,1,0,2) & (1,1,2,0) & (1,2,1,0). \end{array}$$

As λ has no 3's, removing and replacing the first two entries from each partial displacement vector changes nothing. For 4, we separate the first three entries from the rest of each partial displacement vector

$$\begin{array}{ccc} (0,1,1)(2) & (0,1,2)(1) & (0,2,1)(1) \\ (1,0,1)(2) & (1,0,2)(1) & (1,2,0)(1) \\ (1,1,0)(2) & (1,1,2)(0) & (1,2,1)(0). \end{array}$$

list all the ways to insert the 4 into the rightmost portion

$$\begin{array}{cccccc} (0,1,1)(2,4) & (0,1,1)(4,2) & (0,1,2)(1,4) & (0,1,2)(4,1) & (0,2,1)(1,4) & (0,2,1)(4,1) \\ (1,0,1)(2,4) & (1,0,1)(4,2) & (1,0,2)(1,4) & (1,0,2)(4,1) & (1,2,0)(1,4) & (1,2,0)(4,1) \\ (1,1,0)(2,4) & (1,1,0)(4,2) & (1,1,2)(0,4) & (1,1,2)(4,0) & (1,2,1)(0,4) & (1,2,1)(4,0), \end{array}$$

and reunite each different insertion with its corresponding leftmost portion:

$$\begin{array}{cccccc}
(0,1,1,2,4) & (0,1,1,4,2) & (0,1,2,1,4) & (0,1,2,4,1) & (0,2,1,1,4) & (0,2,1,4,1) \\
(1,0,1,2,4) & (1,0,1,4,2) & (1,0,2,1,4) & (1,0,2,4,1) & (1,2,0,1,4) & (1,2,0,4,1) \\
(1,1,0,2,4) & (1,1,0,4,2) & (1,1,2,0,4) & (1,1,2,4,0) & (1,2,1,0,4) & (1,2,1,4,0).
\end{array}$$

Finally, add a 0 at the beginning of each vector to recover all displacement vectors with length $n = 6$ and displacement partition $\lambda = (4, 2, 1, 1)$:

$$\begin{array}{cccccc}
(0,0,1,1,2,4) & (0,0,1,1,4,2) & (0,0,1,2,1,4) & (0,0,1,2,4,1) & (0,0,2,1,1,4) & (0,0,2,1,4,1) \\
(0,1,0,1,2,4) & (0,1,0,1,4,2) & (0,1,0,2,1,4) & (0,1,0,2,4,1) & (0,1,2,0,1,4) & (0,1,2,0,4,1) \\
(0,1,1,0,2,4) & (0,1,1,0,4,2) & (0,1,1,2,0,4) & (0,1,1,2,4,0) & (0,1,2,1,0,4) & (0,1,2,1,4,0).
\end{array}$$

According to the formula in the algorithm description,

$$\begin{aligned}
\frac{\prod_{i=1}^{|\lambda|}(n-\lambda_i-i+1)}{\prod_{j=1}^{\lambda_1}(\#\text{ of } j\text{'s in }\lambda)!} &= \frac{(6-4-1+1)(6-2-2+1)(6-1-3+1)(6-1-4+1)}{(2!)(1!)(0!)(1!)} \\
&= \frac{(2)(3)(3)(2)}{2} = 18,
\end{aligned}$$

which matches the 18 displacement vectors we constructed above.

To enumerate the parking functions in each equivalence class, we introduce the characteristic poset of a displacement vector.

Definition 5.2. The **characteristic poset** of a displacement vector of length n is the poset with elements in $[n]$ described by the following cover relation: car j is covered by car i if $d_i > 0$, $i > j \geqslant i - d_i$, and i is the smallest index for which both of these properties are true. Write car j is covered by car i as car $j <$ car i.

Definition 5.3. A **linear extension** of a poset is a total ordering on the nodes subject to the constraint that if $j < i$ in the poset, then $j < i$ in the total ordering.

Lemma 5.4. *The parking functions with a given parking-ordered order α are the linear extensions of $d(\alpha)$'s characteristic poset labeled with α's preferences, with $d(\alpha)$ the displacement vector of α.*

Proof. First, label the characteristic poset's nodes using the preferences of the parking-ordered parking function. Then suppose ℓ is a linear extension of the poset and read ℓ from its minimal node to its maximal node. Since ℓ comprises the same preferences as α, ℓ is some permutation of α. Since α is a parking function and permutations of parking functions are parking functions, ℓ is a parking function. As ℓ is a linear extension of $d(\alpha)$'s characteristic poset, the order of ℓ's preferences obeys $d(\alpha)$'s characteristic poset's cover relations. By the definition of the cover relations, when ℓ is arranged into its parking rearrangement ℓ', car i will have $d(\alpha)_i$ cars park in front of it starting at its preference. This means that ℓ' has the same displacement vector as α. Since α is also parking-ordered and each displacement vector corresponds to a unique parking-ordered parking function, $\ell' = \alpha$.

Suppose β is a parking function whose parking rearrangement is α. Then car i in β has $d(\alpha)_{\sigma(i)}$ cars park in or immediately after its preferred spot, β_i, before it parks. This means the order of the cars in β obey the cover relations that define the characteristic poset, and therefore this order is achievable through a linear extension. □

This leads to our second algorithm:

Linear Extensions Algorithm
Input: A displacement vector v
Output: The linear extensions of v's characteristic poset
Method: Initialize a list L, which will contain the cover relations for the characteristic poset. Iterate through the entries of v from left to right. For all $i \in [\text{length}(v)]$ and for all $j \in \{i - v_i, \dots i-1\}$, if j has not already been covered, add $j < i$ to the list L of cover relations. Notice that $\{i - v_i, \dots, i-1\}$ only contains positive integers as $v_i < i$ for all $v_i \in v$. By iterating from left to right, we guarantee that car i is the first that gets displaced by car j. Then, use the list of cover relations to construct the corresponding poset through Sage's `linear_extensions` and `cardinality` functions. By [4], these functions run in $O(n^2)$ time using Atkinson's algorithm to compute the number of linear extensions on a tree poset.

Example 5.5. Let $\alpha = (1, 2, 1, 4, 2)$, then the displacement vector is $d(\alpha) = (0, 0, 2, 0, 3)$. Car 3 is the first with nonzero displacement and its displacement is 2 so it covers cars 1 and 2, thus we obtain the cover relations car 1 $<$ car 3 and car 2 $<$ car 3. The next car with nonzero displacement is car 5 and its displacement is 3, so it covers cars 3 and 4. Note that even though $3 > 0$ and $2 \geqslant 5 - 3$, car 5 does not cover car 2 because car 3 already does. This means car 3 $<$ car 5 and car 4 $<$ car 5 are the rest of the cover relations that define the poset. Figure 7 is the Hasse diagram for the characteristic poset of α's displacement vector.

$$\begin{tikzcd}
 & & 5 \arrow[dl, dash] \arrow[dr, dash] & \\
 & 3 \arrow[dl, dash] \arrow[dr, dash] & & 4 \\
1 & & 2 & .
\end{tikzcd}$$

Figure 7: The Hasse diagram for the characteristic poset of $d(\alpha)$

Relabeling the nodes with each car's preference gives

$$\begin{tikzcd}
 & & 2 \arrow[dl, dash] \arrow[dr, dash] & \\
 & 1 \arrow[dl, dash] \arrow[dr, dash] & & 4 \\
1 & & 2 & .
\end{tikzcd}$$

The linear extensions of this poset are

$$\begin{tikzcd}
2 \arrow[d, dash] & 2 \arrow[d, dash] & 2 \arrow[d, dash] & 2 \arrow[d, dash] & 2 \arrow[d, dash] & 2 \arrow[d, dash] & 2 \arrow[d, dash] & 2 \arrow[d, dash] \\
4 \arrow[d, dash] & 1 \arrow[d, dash] & 1 \arrow[d, dash] & 1 \arrow[d, dash] & 4 \arrow[d, dash] & 1 \arrow[d, dash] & 1 \arrow[d, dash] & 1 \arrow[d, dash] \\
1 \arrow[d, dash] & 4 \arrow[d, dash] & 2 \arrow[d, dash] & 2 \arrow[d, dash] & 1 \arrow[d, dash] & 4 \arrow[d, dash] & 1 \arrow[d, dash] & 1 \arrow[d, dash] \\
2 \arrow[d, dash] & 2 \arrow[d, dash] & 4 \arrow[d, dash] & 1 \arrow[d, dash] & 1 \arrow[d, dash] & 1 \arrow[d, dash] & 4 \arrow[d, dash] & 2 \arrow[d, dash] \\
1 & 1 & 1 & 4 & 2 & 2 & 2 & 4 .
\end{tikzcd}$$

The parking functions with partition-preserving order $(1,2,1,4,2)$ are

$$\begin{matrix} (1,2,1,4,2) & (1,2,4,1,2) & (1,4,2,1,2) & (4,1,2,1,2) \\ (2,1,1,4,2) & (2,1,4,1,2) & (2,4,1,1,2) & (4,2,1,1,2). \end{matrix}$$

Combining the displacement vector and linear extension algorithms yields an algorithm whose import is summarized by Theorem 5.6.

Theorem 5.6. *Let* $\mathrm{PF}_n(\lambda)$ *be the set of parking functions of length* n *and displacement partition* λ. *Let* $\text{displacement\_vectors}(n, \lambda)$ *be the set of displacement vectors of length* n *whose nonzero entries are the parts of* λ. *Let* $\text{lin\_extensions\_of\_disp\_vector}(v)$ *be the set of linear extensions of a displacement vector's characteristic poset. Then*

$$|\,\mathrm{PF}_n(\lambda)| = \sum_{v \in \text{displacement\_vectors}(n,\lambda)} |\,\text{lin\_extensions\_of\_disp\_vector}(v)|.$$

Proof. The result follows from Lemma 2.2, Lemma 2.3, and Lemma 5.4. □

5.1 Runtime Analysis

In this section, we analyze the runtime of the code in Appendix B, which outlines the algorithm that counts all the parking functions of length n with displacement partition λ. A full description of the functions in Appendix B is given in Section 5. Throughout this analysis, n represents the length of the parking function inputted into the `pf` function.

We first analyze the function `merge_preserving_order` which returns all possible combinations of two lists such that the order of the first list is preserved in the merged list. Let a be the length of `list1` and b the length of `list2`. The majority of the computation arises from the nested `for` loops beginning on line `21`. The first `for` loop iterates $\binom{a+b}{a}$ times (the output of `indices` on line `18` has that many entries). The second `for` loop iterates over $a+b$ entries with all other lines yielding constant time. Using Stirling's Approximation, one can verify that

$$(a+b)\binom{a+b}{a} = \frac{(a+b)^{\frac{3}{2}}(a+b)^{a+b}}{\sqrt{2\pi ab}\cdot a^a b^b}. \tag{$*$}$$

We will use the expression $(*)$ on the final line to approximate runtime when the `merge_preserving_order` function is used.

The `displacement_vectors` function has worst case $O(\sqrt{n}4^n)$ runtime. The bulk of the runtime begins at the `for` loop on line `60`. The `for` loop runs over the largest part of the partition, which is $n-1$ in the worst case, yielding approximately $O(n)$ iterations for the outermost `for` loop. The first inner `for` loop on line `64` iterates over the elements of `intermediate_vectors2`. At the first iteration, `intermediate_vectors2` has one element. At every step thereafter, `intermediate_vectors2` is cleared, and then populated with `len(merge_preserving_order(second_part_split, [part]*number_of_part))` elements, as seen by line `72`. In the worst case scenario, the partition consists of $n-1$ 1s. This would imply that the inner `for` loop would iterate approximately $\binom{n+n}{n}$ times. Plugging in $a=n$ and $b=n$ into equation $*$, we obtain

$$\frac{(2n)^{\frac{3}{2}}(2n)^{2n}}{2n\sqrt{\pi}\cdot n^{2n}} = \frac{\sqrt{2n}\cdot 4^n}{\sqrt{\pi}}.$$

Therefore, the worst case runtime within each iteration is $O(\sqrt{n}4^n)$. However, as the only unique part of the partition was 1, the inner `for` loop will run in constant time for every other iteration of the outer `for` loop as `number_of_part` is 0. The remaining `for` loops are negligible with respect to $O(\sqrt{n}4^n)$, so the `displacement_vectors` function runs in $O(\sqrt{n}4^n)$ time.

The `lin_extensions_of_disp_vector` function runs in $O(n^3)$ time. The outer `for` loop iterates over the n entries in the displacement vector. The number of iterations of the inner `for` loop is given by the value of the i-th displacement, which is bounded above by $n-1$, giving a runtime of $O(n^2)$ thus far. The `if` statement on line `101` iterates over the values in `cover_relations`, which has a worst case length of $n-1$. Therefore, the if statement runs in $O(n)$, so the nested `for` loops run in $O(n^3)$ runtime. Outside of the loops, the cost of finding the linear extensions of the poset using Atkinson's algorithm is $O(n^2)$. Hence the overall runtime of this function is $O(n^3)$.

Finally, we analyze the `pf` function. The second `for` loop on line `127` is what contributes non-negligible runtime. We iterate at most $O(\sqrt{n}4^n)$ times as calculated by the analysis of the `displacement_vectors` function. Within each iteration, we compute the `lin_extensions_of_disp_vector` function, which runs in $O(n^3)$ runtime. The final runtime is encapsulated in the following theorem.

Theorem 5.7. *The runtime of the* `pf` *function is* $\leqslant O(n^{7/2}4^n)$. □

Sage generates all parking functions of length n by first listing all nondecreasing parking functions (of which there are nth Catalan number many of these) and then permuting all its elements while checking if each is a valid parking function [1]. To find all parking functions of this length n with a displacement partition of λ, one must vet each generated parking function for this specific λ. We will briefly show that our program for generating parking functions of length n and displacement partition λ is far superior (in terms of runtime) to the naive method described above.

Asymptotically, the nth Catalan number C_n approaches

$$C_n \approx \frac{4^n}{\sqrt{\pi}n^{3/2}}.$$

By Stirling's Approximation,

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n,$$

and multiplying these two yields the product

$$\sqrt{2}\left(\frac{4}{e}\right)^n n^{n-1} \geqslant (n+1)^{n-1}.$$

Screening for the displacement partition λ when generating the parking functions requires a runtime of $\Omega(n)$ or greater, but we choose to forgo this term. Hence, the naive implementation will run no faster than a runtime of $\Omega((n+1)^{n-1})$, far slower than the worst case runtime of $O(n^{7/2}4^n)$ for `pf` in Theorem 5.7.

A Some Combinatorial Proofs

Proof of Lemma 4.21. There are three cars displaced, one by m, another by ℓ, and another by k. Let c_m be the car displaced by m, c_ℓ be the car displaced by ℓ, and c_k be the car displaced by k. By assumption all three displacements happen in a single prime, and each displaced car overlaps the previous' prime.

First, suppose the the displacement order is (m, ℓ, k).

After c_ℓ parks, the first $\ell+i$ slots are taken. We must consider the value $\ell+i$. If $k > \ell+i$, then cars must be added in order for c_k to have displacement k. This is because even if c_k attempts to park in slot a, it will only have displacement $\ell+i < k$. Indeed, slots $a+\ell+i$ through $a+k-1$ must be occupied, which is $k-(\ell+i) > 0$ slots. Suppose c_k's preference is the j-th slot in the prime, so $j \geqslant 1$, and $j \leqslant m+1$ to remain a single prime. Since c_k has displacement k, it must park in slot $a+j+k-1$, so slots $a+k$ through $a+k+j-2$ must be occupied. Hence in total, slots $a+\ell+i$ through $a+j+k-2$ must be added, which is $k+j-\ell-i-1$ terms. Then c_k parks with preference $a+j-1$. The total length of the pattern is $m+(\ell-m+i-2)+2+(k-\ell-i+j-1)+1 = k+j$. Also, when $k = \ell+i$ then this same reasoning applies, noting that $k-(\ell+i) = 0$.

If $k < \ell+i$ then no cars are required to be added for c_k to park. Since slots a through $a+\ell+i-1$ are occupied, the next available slot is $a+\ell+i$, so c_k's preference must be at least $a+\ell+i-k$. If p is the preference of c_k, let $j = p-(a+\ell+i-k)+1$, so $j \geqslant 1$. Since the largest preference c_k can have while remaining in a single prime is $a+\ell+i-1$, $j \leqslant k$. In order for c_k to have displacement k with its preference, it must park in slot $p+k = a+\ell+i+j-1$, so the slots $a+\ell+i$ through $a+\ell+i+j-2$, which is $j-1$ slots, must be taken. Then c_k parks with preference $p = a+\ell+i+j-k-1$. The length of the pattern is $\ell+i+j$. This completes case 1.

Now suppose that the displacement order is (m, k, ℓ). After c_k parks, the first $k+i$ slots, slots a through $a+k+i-1$ are occupied. Since $k+i > \ell$, the smallest preference that c_ℓ can have is not a, but $a+k+i-\ell$. If p is the preference of c_ℓ, let $j = p-(a+k+i-\ell)+1$, so $j \geqslant 1$. To remain a single prime, $p \leqslant a+k+i-1$, so $j \leqslant \ell$. Since c_ℓ has displacement ℓ, it must park in slot $p+\ell = a+k+i+j-1$, hence slots $a+k+i$ through $a+k+i+j-2$, which is $j-1$ terms, must be added. Then c_ℓ parks with preference $p = a+k+i+j-\ell-1$. The length of the pattern is $k+i+j$. Hence case 2 follows.

The remaining cases follow from identical reasoning as the two cases above. Case 3 has two subcases, based on whether cars are required to be added or not, so its proof is analagous to the proof of Case 1. The remaining cases follow from an argument analagous to the proof of Case 2.

The statements allowing for equality within certain cases account for the patterns for a displacement partition with some equal terms. These follow from Remark 4.9, that displacement of $k+k$ is identical to the displacement partition $k+\ell$ where displacement $k > \ell$ happens first, then replacing ℓ with k in the pattern. $\square$

Proof of Theorem 4.23. The general approach is to use Lemma 4.21 to characterize the pattern of each displacement order, and then analyze which entries in the pattern can be permuted while preserving the displacement partition.

For the displacement order (k, ℓ, m), the first k terms can be rearranged in any of its $k!$ permutations while still preserving the displacement partition. Likewise, the $i-1$ terms immediately after the second a but before the term $a+k-\ell+i$ can be inserted anywhere before or at their current slots. Hence there are $k+1+i-1 = k+i$ choices for the first term, $k+i-1$ choices for the second, and so on, ending with $k+2$ choices where the $i-1$th term can go, giving a factor of $(k+2)(k+3)\dots(k+i) = \frac{(k+i)!}{(k+1)!}$. Then, the $j-1$ terms after the second $a+k-m+i$ and before the last entry can be inserted anywhere before or at their current slots. There are $k+i+1+j-1 = k+i+j$ choices for the first, and so on, with $k+i+2$ choices for the $j-1$th term, giving a factor of $(k+i+2)(k+i+3)\dots(k+i+j) = \frac{(k+i+j)!}{(k+i+1)!}$. After combining and summing for the possible values of i and j, the case of displacement order (k, ℓ, m) holds. This reasoning also works for (k, m, ℓ), and the subcase of (ℓ, m, k) where no cars are required to be added. For the subcase of (ℓ, m, k) where $k \geqslant \ell + i + 1$, then cars must be added. We follow the same argument, except in the last step, we are adding $k - \ell - i + j - 2$ terms instead of $j - 1$. Hence after the first $\ell + i + 1$ terms have been added, there are $\ell + i + 1 + (k - \ell - i + j - 2) = k + j - 1$ choices for the first term, $k + j$ for the second, and so on, ending with $\ell + i + 2$ choices for the last term. This gives the factor $\frac{(k+j-1)!}{(\ell+i+1)!}$. Following the rest of the logic but replacing this term gives the result for the subcase.

For the displacement order (m, k, ℓ), we consider the subcases $i = 1$ and $i > 1$ separately, since as explained in the proof of Theorem 4.10, having $i = 1$ allows for different "starting values". First, consider $i = 1$. In the first k terms, the subpattern of length $m+1$ which produces the displacement m must appear, so there are $\frac{k!}{(m+1)!}$ ways to select which slots are occupied by this subpattern and arrange the entries outside of it. Also, as in the proof of Theorem 4.10, there are $(k-m)$ choices for the "starting value". Then, we notice that the $j-1$ terms immediately before the final term can all be inserted anywhere before or at their current positions. There are $k + i + (j - 1) = k + i + j - 1$ choices for the first entry, $k+i+j-2$ choices for the second, and so on, ending with $k+i+1$ choices for the last entry, giving a factor of $\frac{(k+i+j-1)!}{(k+i)!}$. Substituting $i = 1$ into this expression, combining and summing over the possible values of j gives the first summand in the case. Now consider $i > 1$, so we can assume that the starting value is a. The first m terms can be rearranged in any order, giving a factor of $m!$. The $k - m + i - 2$ terms immediately after the second a can be inserted anywhere before or at their current slots. Hence there are $m+1+k-m+i-2 = k+i-1$ choices for the first entry in this subpattern, $k+i$ choices for the second, and so on, ending with $m+2$ choices for the last entry in the subpattern, giving a factor of $(m + 2)(m + 3) \dots (k + i - 1) = \frac{(k+i-1)!}{(m+1)!}$. Also, the $j-1$ terms immediately before the final entry can be inserted anywhere before or at their current slots. By the same reasoning as the $i = 1$ subcase, this introduces a factor of $\frac{(k+i+j-1)!}{(k+i)!}$. Combining and summing over the possible values of i and j gives the second summand in the case. We also can apply this exact same reasoning to any displacement order whose two terms in the displacement order are increasing, and where no cars are required to be added. Replacing m with the first car in the order as appropriate, we obtain the counts for the displacement order (ℓ, k, m) and the subcase of (m, ℓ, k) where no cars are required to be added, namely when $k < \ell + i$.

For displacement order (m, ℓ, k) with $k \geqslant \ell + i$, we follow the same argument, except in the last step, we are adding $k - \ell - i + j - 1$ terms instead of $\ell - m + i - 2$. Hence after the first $\ell + i$ terms have been added, there are $\ell + i + k - \ell - i + j - 1 = k + j - 1$ choices for the first car, $k + j - 2$ for the second, and so on, ending with $\ell + i + 1$ choices for the last, giving a factor of $\frac{(k+j-1)!}{(\ell+i)!}$. Breaking up the cases $i = 1$ and $i > 1$, following the previous logic, gives the result.

The claims that match the cases to counts of $k+\ell+m$ where there are some equal terms follows from Remark 4.9. □

Proof of Theorem 4.27. First, consider the case where d_1 is the one car remaining to park. Then slots a through $a + L - 1$ are occupied, so without adding any cars, the maximum displacement that can be attained is L.

If $d_1 > L$ then cars must be added in order to attain displacement d_1. In particular, slots $a + L$ through $a + d_1 - 1$ must be added, which is $d_1 - L$ slots. Suppose that the last car has preference p, and let $j = p - a + 1$, so $j \geqslant 1$. To remain a single prime, $p \leqslant a + L - 1$ so $j \leqslant L$. Since the last car has preference p, it must park in slot $p + d_1 = a + j + d_1 - 1$ in order to have displacement d_1. Hence the slots up to and including $a + j + d_1 - 2$ must be taken. So in total, slots $a + L$ through $a + j + d_1 - 2$, which is $j + d_1 - \ell - 1$ slots, must be added. Then the last car parks with preference

$p = a + j - 1$. The length of the total pattern is $L + d_1 - L + j - 1 + 1 = d_1 + j$. (If $d_1 = L$ then this exact argument holds, and the general formula of $d_1 - L + j - 1$ terms is just just $j - 1$). Hence case 1 follows.

If $d < L$ then no cars are required to be added. Slots a through $a + L - 1$ are taken, so the next available slot is $a + L$. Hence the minimum preference p that the last car can have is $a + L - d_1$ in order to have displacement d_1. Let $j = p - (a + L - d_1) + 1$, so $j \geqslant 1$. To remain a single prime, $p \leqslant a + L - 1$, so $j \leqslant (a + L - 1) + (a + L - d_1) + 1 = d_1$. In order to have displacement d_1, the last car must park in slot $p + d_1 = (a + L + j - d_1) - 1 + d_1 = a + L + j - 1$, so the slots $a + L$ through $a + L + j - 2$, which is $j - 1$ slots, must be added. Then the last car parks with preference $p = a + L + j - d_1 - 1$. The total length of the pattern is $L + j - 1 + 1 = L + j$. Hence the pattern agrees with case 2.

Now let's consider the case where the last car to park is $d_i \neq d_1$. In particular, the length of the subpattern for the first $m - 1$ cars is at least $d_1 + 1 > d_i$, we know that we are not required to add any cars in order for d_i to park. The slots a through $a + L - 1$ are occupied, and since the next available slot is $a + L$, the smallest preference p that the last car can have is $a + L - d_i$. Let $j = p - (a + L - d_i) + 1$, so $j \geqslant 1$. Since we require $p \leqslant a + L - 1$ for the parking function to remain a single prime, this implies $j \leqslant d_i$. In order for the last car to have displacement d_i, it must park in slot $p + d_i = (a + L + j - d_i - 1) + d_i = a + L + j - 1$. Hence slots $a + L$ through $a + L + j - 2$, which is $j - 1$ slots, must be added. Then the last car parks with preference $p = a + L + j - d_i - 1$. The length of the total pattern is $L + (j - 1) + 1 = L + j$. Hence the case follows. □

B Python/Sage Scripts

This section contains the code used to implement Theorem 5.6. Sage version 9.6 and Python 3.7.9 were used for the following code. The function `pf(n, partition)` runs the code which outputs the total number of parking functions with a length n and displacement partition λ.

```
import numpy as np
import itertools

def merge_preserving_order(list1, list2):
    """
    Returns the merging of list1 and list2 such that list1's order is preserved in the
        merged list.

      Parameters:
         list1 (list): A list of nonnegative integers
         list2 (list): A list of one nonnegative integer repeated

      Returns:
         list_final (list): The merged list
    """
    list_final = []
    sequence = list(range(len(list1) + len(list2)))
    #all possible lists of length len(list1) from 0 to length of sequence - 1
    indices = list(itertools.combinations(sequence, len(list1)))

```

```python
    #loop over these indices
    for i in range(len(indices)):
        temp = []
        list1_count = 0 #index in list1
        #building the temp array
        for j in range(len(list1) + len(list2)):
            #append the nonzero value
            if j in indices[i]:
                temp.append(list1[list1_count])
                list1_count += 1
            #append a zero if did not encounter a nonzero value
            else:
                temp.append(list2[0])
        list_final.append(temp)
    return list_final

def displacement_vectors(n, partition):
    """
    Return all possible displacement vectors of length n which can be formed from the
        values in the displacement partition. See section 5 Displacement Program for an
        explanation of the algorithm.

        Parameters:
            n (int): A positive integer
            partition (list): A nonincreasing list of positive integers representing an
                integer partition

        Returns:
            intermediate_vectors2 (list): All displacement vectors of length n formed from
                partition
    """
    #if the partition is empty, initialize it with 0
    if not partition:
        partition = [0]
    #if the parking function is length 1, it can only have the 0 displacement vector
    if n == 1:
        return [[0]]
    #adds zeroes to the displacement partition until it has length n - 1
    partition_with_zeroes = np.pad(partition, (0, n - len(partition) - 1), 'constant')
    intermediate_vectors1 = [] # creating new displacment vectors at the current step

    # all displacement vectors from the previous step will go in this list
    intermediate_vectors2 = [[x for x in partition_with_zeroes if x == 0]]

    #looping from 1 to the largest value in partition
    for part in range(1, partition_with_zeroes[0] + 1):
        #counting the number of each part in the partition
        number_of_part = np.count_nonzero(partition_with_zeroes == part)
        for item in intermediate_vectors2:
            #for each item in intermediate_vectors2, we want to insert our part number
            #throughout item and then append that to intermediate_vectors1. However
            #we only want to do this after ignoring the first item - 1 terms.
            first_part_split = item[:part - 1] #splits off the first item - 1 terms
            second_part_split = item[part - 1:] #gives the remaining terms

            #merges the second list with the new parts while preserving the order of second
                list in merged_dps
            merged_dps = merge_preserving_order(second_part_split, [part]*number_of_part)
            #reappends the first part to each new list in merged_dps
            for merged_dp in merged_dps:
                intermediate_vectors1.append(first_part_split + merged_dp)
        #setup for the next value in the displacement partition
        intermediate_vectors2 = intermediate_vectors1
        intermediate_vectors1 = []
    #insert a 0 at the beginning of the displacement vectors and return
    for dp in intermediate_vectors2:
        dp.insert(0, 0)
    return intermediate_vectors2

def lin_extensions_of_disp_vector(disp_vector):
    """
    Return the number of linear extensions of the characteristic poset of the displacement
        vector. See section 5 Displacement Program for an explanation of the algorithm.

        Parameters:
            n (int): A positive integer
            partition (list): A nonincreasing list of positive integers representing an
                integer partition

        Returns:
            intermediate_vectors2 (list): All displacement vectors of length n formed from
                partition
    """
    cover_relations = []
    #loop over the displacement vector
    for i in range(len(disp_vector)):
        #loop from 0 to value of disp_vector[i]
        for j in range(int(disp_vector[i])):
            #check if j is in first element of elements of cover_relations
            if i - j - int(1) not in [row[int(0)] for row in cover_relations]:
                #j has not been covered, so add it to the cover relations
                cover_relations.append([i - j - int(1), i])
    #creating the poset in sage with nodes as elements in the displacement vector and cover
         relations
    poset = Poset((list(range(len(disp_vector))), cover_relations), cover_relations = True)
    #return the number of linear extensions of that poset
    return poset.linear_extensions().cardinality()

def pf(n, partition):
    """
    Returns the number of parking functions with a length n and displacement partition.

        Parameters:
            n (int): A positive integer
            partition (list): A nonincreasing list of positive integers representing an
                integer partition

        Returns:
            counter (int): The number of parking functions of length n and displacement
                partition
    """
    counter = 0
    #Sorts the partition in nonincreasing order if it was inputted incorrectly
    partition.sort(reverse = True)
    #Checks if the partition is valid by the condition that partition[i] >= n - i
    for i in range(len(partition)):
        if partition[i] >= n - i:
            return 0
    for disp_vector in displacement_vectors(n, partition):
        counter = counter + lin_extensions_of_disp_vector(disp_vector)
    return counter
```

Acknowledgements

The authors would like to thank Susanna Fishel for her continued support throughout this project. This material is based upon work supported by the National Science Foundation under Grant No. DMS-1929284 while the authors were in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the Summer@ICERM 2022 program.

References

[1] *Sagemath parking function documentation*, accessed September 19, 2022.

[2] Y. Aguillon, D. Alvarenga, P.E. Harris, S. Kotapati, J.C. Martinez Mori, C.D. Monroe, Z. Saylor, C. Tieu, and D.A. Williams III, *On parking functions and the tower of hanoi*, American Mathematical Monthly (2022).

[3] Drew Armstrong, Nicholas A Loehr, and Gregory S Warrington, *Rational parking functions and catalan numbers*, Annals of Combinatorics **20** (2016), no. 1, 21–58.

[4] Mike D Atkinson, *On computing the number of linear extensions of a tree*, Order **7** (1990), no. 1, 23–25.

[5] E. W. Bowen, personal communication to Sloane, N. J. A. (1976).

[6] Melody Bruce, Michael Dougherty, Max Hlavacek, Ryo Kudo, and Ian Kit Nicolas, *A decomposition of parking functions by undesired spaces*, The Electronic Journal of Combinatorics **23** (2016).

[7] Emma Colaric, Ryan DeMuse, Jeremy L Martin, and Mei Yin, *Interval parking functions*, Advances in Applied Mathematics **123** (2021), 102129.

[8] Louis Comtet, *Advanced combinatorics*, Springer, 1974.

[9] Alan G. Konheim and Benjamin Weiss, *An occupancy discipline and applications*, Siam Journal on Applied Mathematics - SIAMAM **14** (1966).

[10] I. Gessel and S. Seo, *A refinement of cayley's formula for trees*, Electric Journal of Combinatorics **11** (2006), R27.

[11] Ira Gessel and Richard P Stanley, *Stirling polynomials*, Journal of Combinatorial Theory, Series A **24** (1978), no. 1, 24–33.

[12] Louis H Kalikow, *Enumeration of parking functions, allowable permutation pairs, and labeled trees*, Brandeis University, 1999.

[13] Ronald Pyke, *The supremum and infimum of the poisson process*, Ann. Math. Statist. **30** (1959), no. 2, 568–576.

[14] John Riordan, *Ballots and trees*, J. Combinatorial Theory **6** (1969), 408–411. MR 234843

[15] Richard P. Stanley, *Enumerative combinatorics. vol. ii*, Cambridge University Press, 1999.

[16] Catherine H. Yan, *Parking functions*, Handbook of enumerative combinatorics **440** (2015), 835–893.