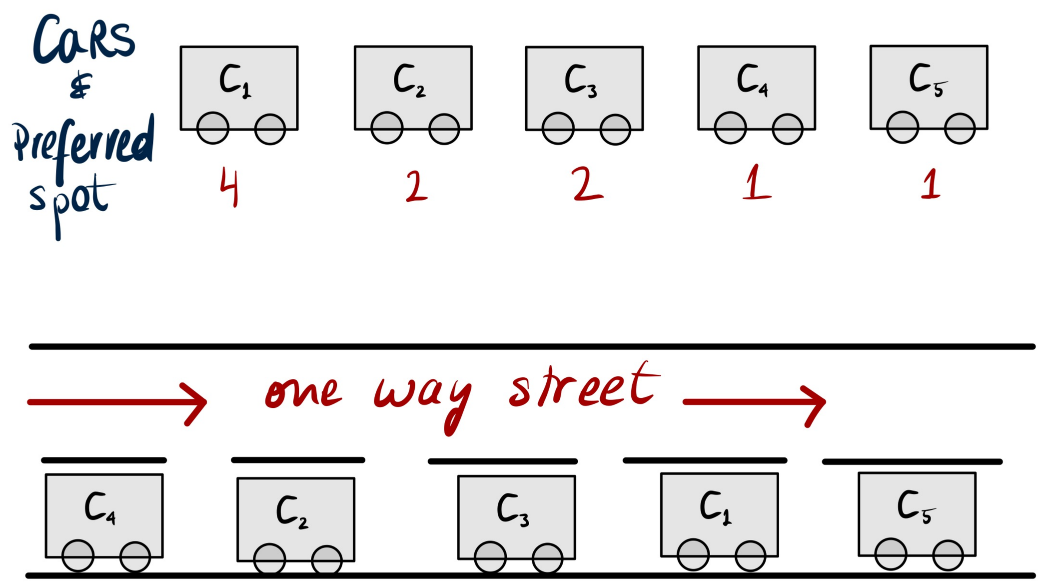
Counting k-Naples parking functions through permutations and the k-Naples `area` statistic

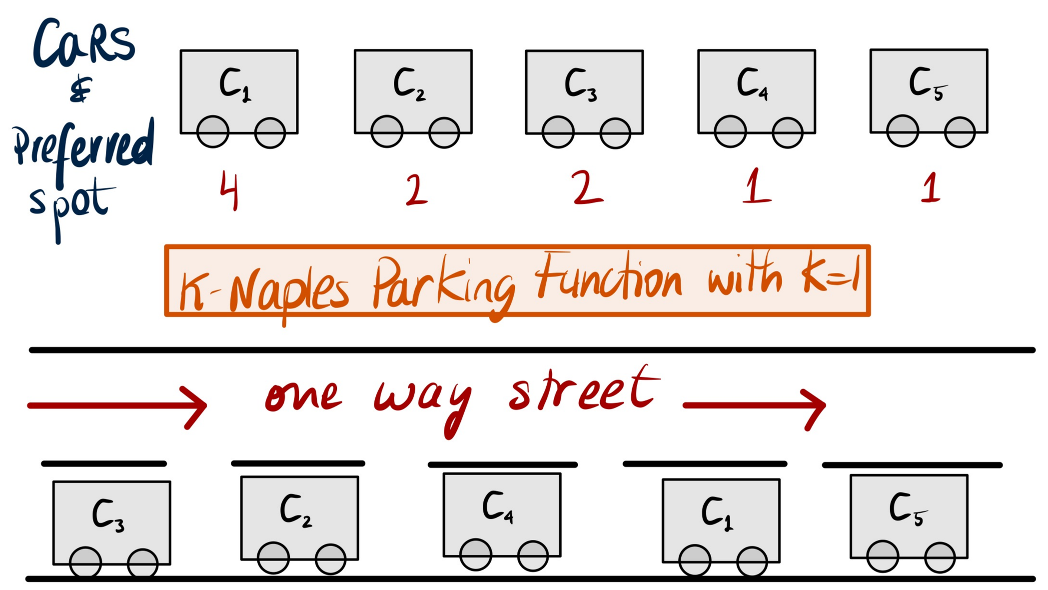
Christo Keller (UMass Amherst) Eunice Sukarto (UC Berkeley)

joint work with L. Colmenarejo, P. Harris, Z. Jones, A. Ramos Rodríguez, and A. R. Vindas-Meléndez (arXiv:2009.01124)

Parking functions [1]

- A **parking function** of length n is an n-tuple $s_i = (s_1, s_2, \dots, s_n)$ in $[n] = \{1, 2, \dots, n\}$ the increasing rearrangement s_i' of which satisfies the sentence $s_i' \leq i$ for all $i \in [n]$.
- Parking functions can be characterized by the **parking rule**, which tries to park n cars on a one-way street with n parking spots based on an n-tuple $s_i \in [n]^n$:
  Imagine n cars travel down a one-way street with n parking spots. Each car prefers a spot, which it attempts to park in. If the spot is empty, it parks there and succeeds; otherwise, it continues down the road until it finds an empty spot.
- For example, the 5-tuple $(4, 2, 2, 1, 1)$ is a parking function of length 5.
  - The increasing rearrangement $(1, 1, 2, 2, 4)$ satisfies the initial definition.
  - All 5 cars will park according to the algorithm in the second bullet point.
- But for instance the 5-tuple $(4, 5, 2, 4, 1)$ fails to be a parking function of length 5.
  - The increasing rearrangement $(1, 2, 4, 4, 5)$ fails at the third position!
  - When the fourth car tries to park, it will find no spot available.
- A clever argument shows that the number of parking functions of length n is $(n+1)^{n-1}$.
  - This gives, for instance, a bijection with rooted trees on $n+1$ vertices, by Cayley's formula.
  - This in turn gives a bijection with rooted forests on n vertices (just remove the roots).
  - This also counts monomials in the Hilbert series of diagonal harmonics and arrangement in the Shi hyperplane regions, among other combinatorial connections (eg. to the Pitman-Stanley polytope).





k-Naples parking functions

- The k-**Naples parking functions** generalize the parking functions by abstracting on the parking rule.
- As each car parks, if it finds its spot taken, before it checks ahead of itself, it checks first the spot behind it, then the spot two behind it, and so on up to k spots. (You can imagine this as a sort of generalized parallel parking.)
- Now we can call the parking functions the 0-Naples. For example, $(1, 3, 3)$ fails to be 0-Naples, but it is 1-Naples.
- Christensen et al [2] gave a recursion for the number of k-Naples parking functions of length n:

$$|PF_{n+1,k}| = \sum_{i=0}^{n} \binom{n}{i} \min\{(i+1)+k, n+1\} |PF_{i,k}| (n-i+1)^{n-i-1}$$

- Specializing to $k = 0$ and using the formula for the number of parking functions, we get the corollary

$$(n+1)^{n-1} = |PF_{n,0}| = \sum_{i=0}^{n-1} \binom{n-1}{i} (i+1)^i \, (n-i)^{n-i-2}$$

A closed form formula

- Define $\varphi_k : PF_{n,k} \to S_n$ by $\varphi(a_1, a_2, \dots, a_n) = s_1 s_2 \cdots s_n$, where parking spot i is occupied by the s_i^{th} car.
- For example, if $\alpha = (4, 2, 2, 4, 1)$, then in **one-line notation** $\varphi_1(\alpha) = 32415$ since the cars park like so:

spot 1	spot 2	spot 3	spot 4	spot 5
c_3	c_2	c_4	c_1	c_5

- Given a permutation $\sigma = s_1 \cdots s_n \in S_n$, let
  - $\texttt{left}_k(i;\sigma)$ = length of longest subsequence $s_j \cdots s_{i-1}$ such that $s_t < s_i$, for all $j \leq t < i$.
  - $\texttt{right}_k(i;\sigma)$ = length of longest subsequence $s_i \cdots s_r$ such that $r \leq i+k$ and $s_t \leq s_i$ for all $i \leq t \leq r$.
- Define the ℓ_k function counting choices for a_i

$$\ell_k(i;\sigma) = \begin{cases} \texttt{left}_k(i;\sigma) + \texttt{right}_k(i;\sigma) & \text{if } \texttt{left}_k(i;\sigma) = i-1 \\ \max(\texttt{left}_k(i;\sigma) - k, 0) + \texttt{right}_k(i;\sigma) & \text{if } \texttt{left}_k(i;\sigma) < i-1. \end{cases}$$

- For example, $n = 5$, $k = 2$, and $\sigma = 51423 \in S_5$. Then
  - $\texttt{left}_2(1;\sigma) = \texttt{left}_2(2;\sigma) = \texttt{left}_2(4;\sigma) = 0$, $\texttt{left}_2(3;\sigma) = \texttt{left}_2(5;\sigma) = 1$.
    $\texttt{right}_2(1;\sigma) = \texttt{right}_2(3;\sigma) = 3$, $\texttt{right}_2(2;\sigma) = \texttt{right}_2(4;\sigma) = \texttt{right}_2(5;\sigma) = 1$.
  - Therefore, $\ell_2(1;\sigma) = 3$, $\ell_2(2;\sigma) = 1$, $\ell_2(3;\sigma) = \max(1-2, 0) + 3 = 3$, $\ell_2(4;\sigma) = 1$, and $\ell_2(5;\sigma) = \max(1-2, 0) + 1 = 1$. This means there are $3 * 3 = 9$ such parking functions.
- For each $\sigma \in S_n$, there are $|\varphi_k^{-1}(\sigma)| = \prod_{i=1}^{n} \ell_k(i;\sigma)$ k-Naples parking functions mapping to it.

Theorem: Counting the number of k-Naples parking functions

For all $n \geq 1$ and $0 \leq k \leq n-1$, we have the formula

$$|PF_{n,k}| = \sum_{\sigma \in S_n} \left(\prod_{i=1}^{n} \ell_k(i;\sigma) \right).$$

A logarithmic generating function

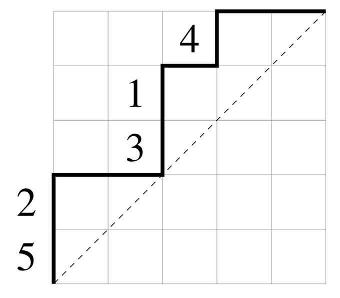
Theorem

Let $c_{n,i}$ be the number of permutations in S_n with fiber of size i under ϕ_0. Then

$$G_n(q) = \sum_{i=1}^{n!} c_{n,i} q^{\ln(i)} = \sum_{i=0}^{n-1} \binom{n-1}{i} q^{\ln(i+1)} G_i(q) G_{n-1-i}(q)$$

Statistical distributions giving q-analogues

- A q-**analogue** $[Q]_q$ of a Q is a formula in q with $\lim_{q\to 0}[Q]_q = Q$.
- A **statistic** f on a set S is a function $f : S \to \mathbf{Z}$ (Eg. the divisor function.)
- The **distribution** $D(f;q)$ of a statistic f on S is the power series $\sum_{s\in S} q^{f(s)}$.
- Parking functions can be thought of as **labelled Dyck paths**. See Figure 1.
- The `area` of a parking function is the number of boxes so $\texttt{area}(31341) = 3$.



Labelled Dyck path for $(3, 1, 3, 4, 1)$.

Theorem

$$D(\texttt{area};q) = \sum_{f \in PF_{n,0}} q^{\texttt{area}(f)} = \sum_{\sigma \in S_n} \left(\prod_{i=1}^{n} [\ell_k(i;\sigma)]_q \right)$$

where $[-]_q$ is the usual q-analog $[m]_q = 1 + q + \cdots + q^{m-1}$.

- Further, we found a statistic $\texttt{area}_k$ on the k-Naples parking functions generalizing the theorem to $k > 0$:

$$\texttt{area}_k(f) = \sum_{i=1}^{n} [n - i + \texttt{right}_k(i;\sigma)(\phi_k(f)) - f_i]$$

  which specializes to $\texttt{area}(f) = \sum_{i=1}^{n} [n - i - f_i + 1]$ when $k = 0$ (recall the definition of `right`).

Bibliography

[1] R. P. Stanley. *Parking functions*. Online at http://www-math.mit.edu/ rstan/transparencies/parking.pdf.

[2] A. Christensen, P. E. Harris, Z. Jones, M. Loving, A. R. Rodríguez, J. Rennie, and G. R. Kirby. A generalization of parking functions allowing backwards movement. *The Electronic Journal of Combinatorics*, 27(1):33, 2020.