Sampling from defect d preference lists

The main idea of this way of sampling from defect d preference lists is to use the last break point as a way of dividing up the preference lists into two parts, each of which can be generated uniformly more easily. One half is a parking function which can be generated uniformly at random using the circular argument with $n+1$ spots [2, 4]. The other half is a new type of parking function that I am calling a p-prime parking function. If we want to generate a defect d preference list, we will need to be able to generate a d-prime parking function. Note that for $d = 1$, this reduces to just generating a prime parking function, which can be achieved with a circular argument with $n-1$ spots [4]

Prime parking functions. There are several equivalent ways of thinking of prime parking functions for $n = m$:

- No break points. That raises the question, what is a break point? The main way to understand a break point is that no car attempts to park in spot k and then must move to the next spot. For $n = m$ and $d = 0$, there are more tricks which we can easily use. A break point is a value k such that k cars want to park in the first k spots. Equivalently if π' is the increasing rearrangement of π, then $\pi'_k = k$.
- If you remove one of the cars which wants the first spot, the result is still a parking function
- Increasing rearrangement of π' satisfies $\pi'_i < i$ (the normal condition is $\pi'_i \leq i$ for parking functions)

p-prime parking functions. A p-prime parking function is a prime parking function where all of the preferences come from $[n-p]$ rather than from $[n]$ (ie no cars want to park in the last p spots).

Note that a 1-prime parking functions is the same as a normal prime parking function. This is useful since to generate a prime parking function, there is a circular argument with $n-1$ spots [4]

Let $|PPF_{n,p}|$ be the number of p-parking functions. Note that $|PPF_{n,1}| = |PPF_n| = (n-1)^{n-1}$. This result is referenced in this paper [4] and is an exercise on page 95ish (solution on 141) [3].

I calculated a couple counts using inclusion-exclusion like arguments:

$$|PPF_{n,2}| = (n-1)^{n-1} - n(n-2)^{n-2}$$
$$|PPF_{n,3}| = (n-1)^{n-1} - 2n(n-2)^{n-2} + n(n-1)(n-3)^{n-3} - \binom{n}{2}(n-3)^{n-3}$$

For $|PPF_{n,2}|$, the term that is subtracted represents the number of prime parking functions for n which include the value $n-1$ as a preference. This can be thought of as the number of different shuffles of a prime parking functions on $n-1$ with the value $n-1$.
For $|PPF_{n,3}|$: The second term is associated with shuffles of $n-1$ and $n-2$ with prime

parking functions of length $n-1$; The third term is associated with shuffles of $n-1, n-2$ with prime parking functions of length $n-2$; the last term is associated with shuffles of $n-2, n-2$ with prime parking functions of length $n-2$

While these terms are helpful for counting, they are less useful for sampling since they are using an inclusion-exculsion type argument, but at least they can give a sense for how repeated sampling would perform in the limit as n gets large

To determine the way in which you should sample from the location of the last break point in order to sample uniformly from defect d preference lists, I worked out a helpful generating function relating two useful things - defect and distance of the last break point from the end. Let $F(x,y)$ be a generating function where the exponent on x is the location of the last break point from the end of the parking function and the exponent on y is the defect. Then

$$\begin{aligned} F(x,y) &= \sum_{\pi \in [n]^m} x^{s(\pi)} y^{d(\pi)} \\ &= \sum_{d=0}^{m-1} \sum_{s=1}^{m-d} \binom{m}{s+d} (n-m+1+d)(n-s+1)^{m-s-d-1} |PPF_{n=s+d,p=d}| x^s y^d \\ &\quad + (n-m) n^{m-1} x^0 d^0 \qquad \text{Added 6/30 to include perference lists where } s=0 \end{aligned}$$

This comes from shuffling a parking function for $m-(s+d)$ cars and $n-s$ spots with a d-prime parking function for $s+d$ cars

This expression lets you count the total defective parking functions by plugging in 1 for x and looking at the dth term for x. This matches the results in the table in Cameron [1]. Verified for $d=0, 2 \leq n=m \leq 10$, $d=1, 2 \leq n=m \leq 10$ and $d=2, 2 \leq n=m \leq 10$.

Here is a desmos that I made because I was curious what the distribution that you draw from for s looked like, and how it behaves as n gets bigger. It doesn't tend towards uniform - the ends seem to have larger values than the middle. I would appreciate another pair of eyes looking at these distributions to see if there is an obvious kind of distribution that they might be

The sampling procedure

1. Choose the value of s (the last break point in the preference list) according to the distribution of possible values for s given defect d, so

$$\begin{aligned} P(s=s_0) &= \frac{\frac{1}{d!s_0!} \frac{\partial^{s_0}}{\partial x^{s_0}} \frac{\partial^d}{\partial y^d} F(1,y)\Big|_{y=0,x=0}}{\frac{1}{d!} \frac{\partial^d F(1,y)}{\partial y^d}\Big|_{y=0}} \\ &= \frac{\binom{m}{s_0+d}(n-m+1+d)(n-s_0+1)^{m-s_0-d-1}|PPF_{n=s_0+d,p=d}|}{|P_{n,m,d}|} \end{aligned}$$

   (excessive calculus included because I finally realized why calculus comes up in the context of generating functions)

2. Choose a parking function for $n-(s+d)$ cars and $n-s$ spots uniformly at random using the circular argument. Let us call this parking function π

3. Choose a $d-$prime parking function uniformly at random. Let us call this preference τ. Note this step is still tricky, but here are some useful tricks

   - For $d=1$, we can use the circular argument with $n-1$ cars to find a prime parking function [4][3]
   - For $s=1$, all cars must choose the same spot
   - For $s=2$, 2 cars must pick the first spot, and the rest must split between the two spots somehow
   - For $s=3$, maybe you could split into cases??
   - Maybe you could show that for big enough n, just repeating 1 until you get something that works isn't the worst idea ever?

4. Shuffle your two parking functions π and τ

To Do

I got pulled into thinking about how to randomly sample because it was very exciting that this idea worked out somewhat well. I still need to

- Work on putting the questions you are leaving to the side written down in an email draft
- Add to and update your list of permutation statistics (use notes from [4] in 6/19 notes) and index of last break point and number of break points (should define more carefully in general).

Reading

- Ian's Thesis
- Paper about random walks (would you double check that this paper sent)

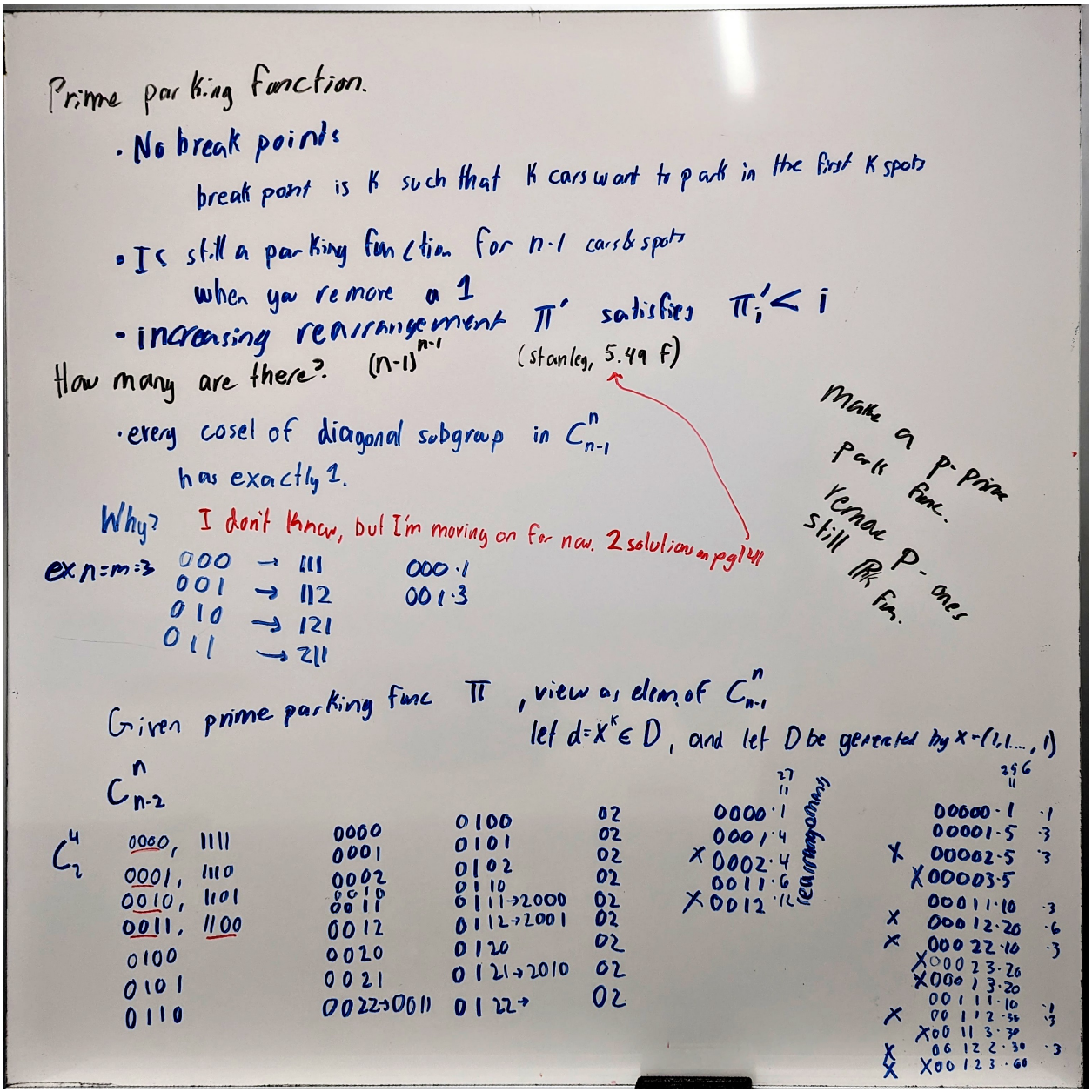
Code

- Write code to do random walks on preference lists and make plots of statistics of interest
- Write code to sample from defect d preference lists (at least for some of them... will have to work out the sticky step 3 a bit more)
- Write code to measure the distance of a statistic or distribution from uniform (and make plots especially for the diaconis and hicks like plots)

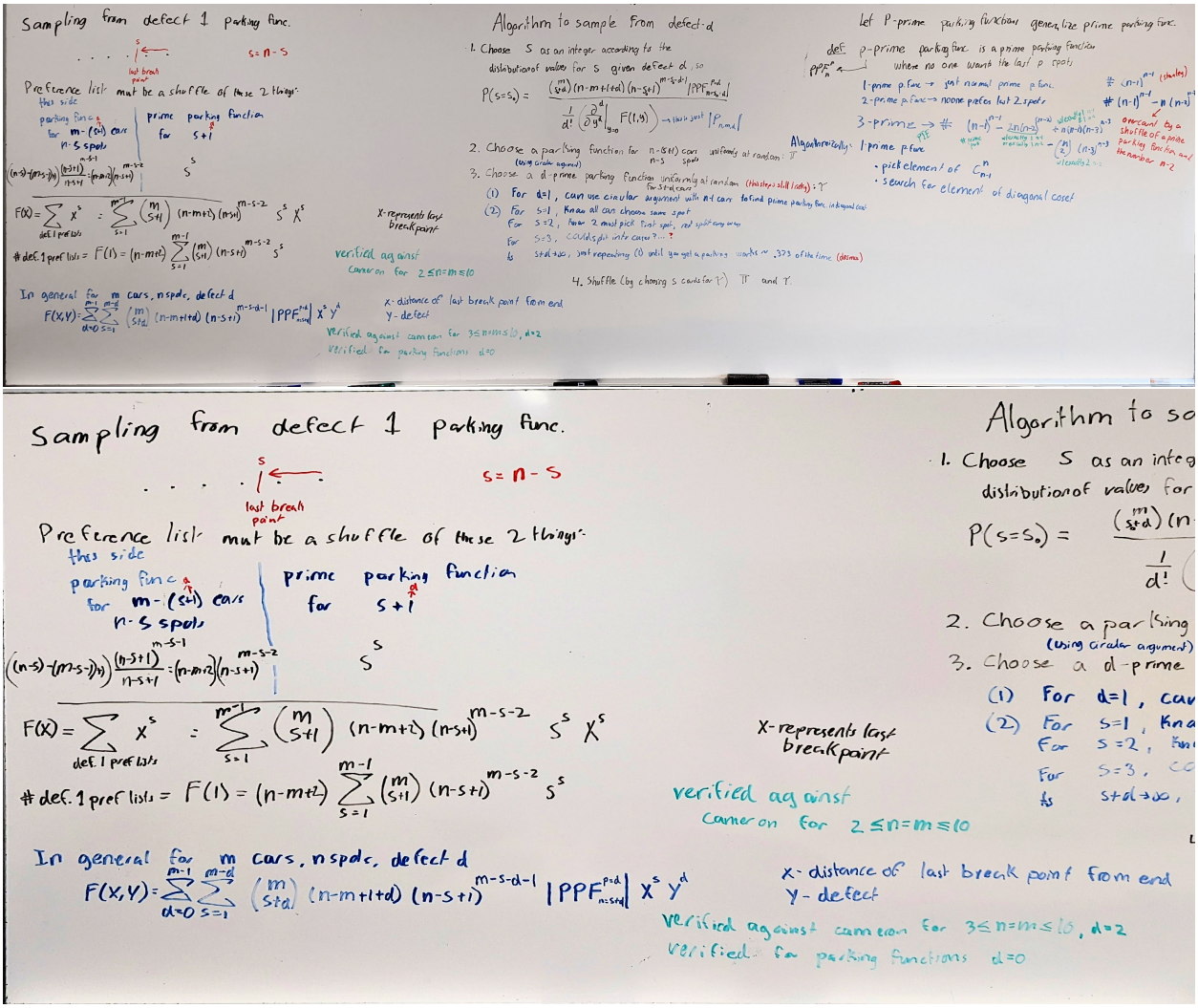
Things to think about

- You have convenient computational evidence that some of the statistics are somewhat local... prove it (haha I already have see write up from 6/12)
- Random walks! (motivation for why you might care about things being somewhat local)
- foundational functions? keep these in mind and see if they make life easier
- If you run linear probing on a random preference list, what kind of distribution does that give you on permutations?

Whiteboard pictures



This is the full whiteboard, and the three following photos are the three(ish) sections of the board



Algorithm to sample from defect-d

1. Choose s as an integer according to the distribution of values for s given defect d, so

$$P(s=s_0) = \frac{\binom{m}{s_0+d}(n-m+1+d)(n-s_0+1)^{m-s_0-d-1}\left|PPF^{p=d}_{n=s_0+d}\right|}{\frac{1}{d!}\left(\frac{\partial^d}{\partial y^d}\Big|_{y=0} F(1,y)\right)} \rightarrow \text{this is just } \left|P_{n,m,d}\right|$$

2. Choose a parking function for $n-(s+1)$ cars, $n-s$ spots (using circular argument) uniformly at random: π

3. Choose a d-prime parking function uniformly at random (this step's still tricky): γ for $s+d$ cars
   - (1) For $d=1$, can use circular argument with $n-1$ cars to find prime parking func. in diagonal coset
   - (2) For $s=1$, know all cars choose same spot
     For $s=2$, know 2 must pick first spot, rest split easy ways
     For $s=3$, could split into cases? ... ?
     As $s+d \to \infty$, just repeating (1) until you get a parking works $\sim .373$ of the time (desmos)

$n \leq 10$

4. Shuffle (by choosing s cards for γ) π and γ.

Let

def p-p PPF^p_n ←

1-p

2-p

3-p

Algorithmically: 1-pr

Let P-prime parking functions generalize prime parking func.

def p-prime parking func is a prime parking function where no one wants the last p spots PPF^p_n ←

1-prime p. func → just normal prime p. func.

2-prime p. func → noone prefers last 2 spots

3-prime → # PIE $(n-1)^{n-1} - \underbrace{2n(n-2)^{(n-2)}}_{\text{w/ exactly 1 } n-1 \text{ or exactly 1 } n-2} + \underbrace{n(n-1)(n-3)^{n-3}}_{\text{w/ exactly 1 } n-1 \text{ \& 1 } n-2} - \underbrace{\binom{n}{2}(n-3)^{n-3}}_{\text{w/ exactly 2 } n-2}$

($\#$ prime parks)

$\#\ (n-1)^{n-1}$ (stanley)

$\#\ (n-1)^{n-1} - n(n-2)^{n-2}$ ↙ overcount by a shuffle of a prime parking function and the number $n-2$

Algorithmically: 1-prime p. func

- pick element of C^n_{n-1}
- search for element of diagonal coset

om: π

s still tricky): γ

rime parking func. in diagonal coset

References

[1] Peter J. Cameron, Daniel Johannsen, Thomas Prellberg, and Pascal Schweitzer. Counting defective parking functions, 2008.

[2] Persi Diaconis and Angela Hicks. Probabilizing parking functions. *Advances in Applied Mathematics*, 89:125–155, 2017.

[3] Richard P. Stanley and Sergey Fomin. *Enumerative Combinatorics*, volume 2 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1999.

[4] Richard P. Stanley and Mei Yin. Some enumerative properties of parking functions, 2023.