Local parking procedures on the integers

Philippe Nadeau

Local parking procedures on the integers *

Philippe Nadeau†[1]

[1]Univ Lyon, CNRS, Université Claude Bernard Lyon 1, UMR 5208, Institut Camille Jordan, 43 Blvd. du 11 Novembre 1918, F-69622 Villeurbanne Cedex, France

Abstract

We introduce a large class of parking procedures on $\mathbb{Z}$ generalizing the classical one. This class is characterized by natural local constraints that the procedures must satisfy. We uncover some nice combinatorics attached to such procedures, including a certain universal enumeration formula.

1 Introduction

Classical parking functions (or words) are one of the fundamental objects of enumerative and algebraic combinatorics, connected to various structures such as noncrossing partitions, hyperplane arrangements, and many others: see for instance the survey [8] and references therein. The corresponding parking procedure on $\mathbb{Z}$ was originally defined as a very simple hashing procedure [4].

We recall their definition informally: r cars want to park on an empty street where the spots are labeled by $1, 2, \ldots, r$ from left to right. The cars arrive successively, and the ith car has a preferred spot v_i. If its spot is available, it parks there, and if not it parks in the nearest available spot on the right. The function $v : i \mapsto v_i$ is called a *parking function* if at the end, all cars managed to park.

The main result is that the number of parking functions is given by the simple formula $(r+1)^{r-1}$. Another standard result is the following characterization: $v : i \mapsto v_i$ is a parking function if and only if for any $k = 1, \ldots, r$, there are at least k indices i such that $1 \leq v_i \leq k$.

In ongoing joint work with Vasu Tewari [6], the authors defined procedures that could be rephrased as certain parking[1] algorithms themselves. This gave the idea to define a general mathematical framework for these procedures, which is the content of this abstract.

We first describe the local procedures $\mathcal{P}$ that we want to consider. We will then see that the enumeration $(r+1)^{r-1}$ is in a sense universal for our parking procedures, see Corollary 9. We also describe some natural connection with the combinatorics of binary trees via a natural encoding, and finally define certain extensions of our model.

2 Our setup

We first slightly extend the sequence of desired spots from the cars. Let S be any set, and consider the alphabet $A = \mathbb{Z} \times S$. The set S represents some extra information: in terms of cars, one might consider its brand or color. We let $\mathrm{val} : A = \mathbb{Z} \times S \to \mathbb{Z}$ be the projection to the first factor, which represents the desired spot. As we will see the second factor in S will be carried along nicely in our construction, and we will focus mostly on the values.

*The full version of this work will be published elsewhere. This research is partially supported by the project ANR19-CE48-011-01 (COMBINÉ)

†Email: nadeau@math.univ-lyon1.fr.

[1]We use parking in a loose fashion in this whole abstract, as we do not focus on any practical application of the procedures but rather on the combinatorial structures that they give rise to.

Remark 1. *When S is a singleton $S = \{\bullet\}$, we identify $A \cong \mathbb{Z}$ and* val *is the identity.*

A sequence of incoming cars is given by a word $a_1 a_2 \dots a_r$ with $a_i \in A = \mathbb{Z} \times S$, read from left to right. The ith car wants to park in $\mathrm{val}(a_i)$.

We want to describe certain functions

$$\mathcal{P} : A^* \to \mathrm{Fin}(\mathbb{Z}) = \{I \subset \mathbb{Z} \mid \#I < +\infty\}.$$

$\mathcal{P}(a_1 \cdots a_r)$ is the subset of occupied spots after cars with preferences $a_1, \dots, a_r \in \mathbb{Z} \times S$ have arrived successively. We define our parking procedures on $\mathbb{Z}$ –instead of a finite or semiinfinite interval– in order to get rid of any boundary effect. For an element of $\mathrm{Fin}(\mathbb{Z})$, its *(maximal) intervals* are its connected components as an induced graph of $\mathbb{Z}$.

Remark 2. *More generally, one can require $\mathcal{P}$ to be a* partial *function, that is, to be defined on a subset $L \subset A^*$. It is reasonable to require that L be closed under deleting a letter at any position, that is, that L be closed under taking subwords. For example, one can forbid certain letters in A to occur more than a certain number of times.*

In our models *all cars are able to park, and cars do not move again once they're parked*: $\mathcal{P}(\epsilon) = \emptyset$, and for any $W \in A^*, a \in A$ we have $\mathcal{P}(Wa) = \mathcal{P}(W) \sqcup \{i\}$ for a certain $i \in \mathbb{Z}$.

We denote $\mathrm{ls}(Wa) := i$, which is the spot where the last car parks. If $W = a_1 \cdots a_r$, then the function $\pi^W : \{1, \dots, r\} \to \mathcal{P}(W), i \mapsto \mathrm{ls}(a_1 a_2 \cdots a_i)$ is a bijection.

We now define the first two requirements for $\mathcal{P}$, which can be summarized as: If your desired spot is available, park there, otherwise park to the nearest spot either to the left or to the right.

- **(Lucky parking)** If $\mathrm{val}(a) \notin \mathcal{P}(W)$, then $\mathrm{ls}(Wa) = \mathrm{val}(a)$;
- **(Local move)** If $\mathrm{val}(a) \in \mathcal{P}(W)$, let $[t, u]$ be the maximal interval of $\mathcal{P}(W)$ such that $\mathrm{val}(a) \in [t, u]$. Then $\mathrm{ls}(Wa) \in \{t - 1, u + 1\}$.

To define $\mathcal{P}$, it thus suffices to determine a "rule" that picks either $\mathrm{ls}(Wa) = t - 1$, *left*, or $\mathrm{ls}(Wa) = u + 1$, *right*, whenever $\mathrm{val}(a) \in \mathcal{P}(W)$, that is when the desired spot is occupied. We will require such a rule to be local in some specific way described by the last two conditions.

Remark 3. *Note that a recent variation of parking functions, k-Naples functions [1], do not enter our framework. The rule there is to back up k spots before finding the first available spot on the right, which sometimes contradicts the previous requirements.*

We now come to the remaining two constraints on the functions $\mathcal{P}$. Informally, we require invariance under translation (shift), and that left/right decisions must depend only on the subsequence of cars that parked on the encountered interval.

Mathematically speaking, let $\tau : i \mapsto i + 1$ be the shift on $\mathbb{Z}$. It extends to subsets of $\mathbb{Z}$. Also, if $a = (i, s)$, define $\tau(a) = (i + 1, s)$ and extend to words.

- **(Shift invariance)** For any W, $\mathcal{P}(\tau(W)) = \tau(\mathcal{P}(W))$.
- **(Local decision)** Consider an interval I that is maximal in $\mathcal{P}(W)$ and $\mathcal{P}(W')$ for two words W and W', such that in both words the subword corresponding to the cars parked in I is identical. Then for any a such that $\mathrm{val}(a) \in I$, $\mathrm{ls}(Wa) = \mathrm{ls}(W'a)$.

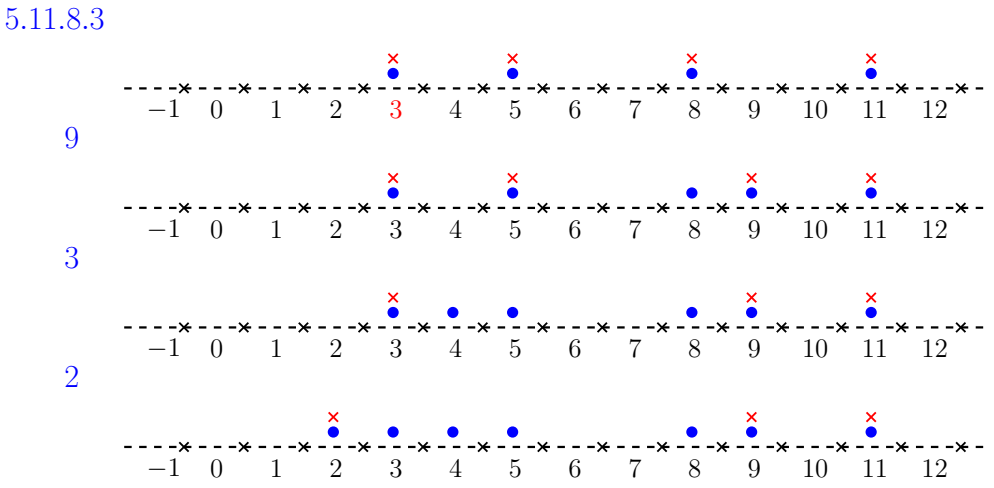
Definition 4. *We declare a parking procedure $\mathcal{P}$ to be* local *if it satisfies **Lucky parking, Local move, Shift invariance** and **Local decision**.*

All these requirements may not seem completely natural at first sight, but their motivation and use will become clearer in the next sections.

Example 5. *The classical parking procedure $\mathcal{P}^{usual}$ is obtained in the standard case $A = \mathbb{Z}$ by always picking the nearest spot to the right.*

Example 6. *The CS-parking procedure*[2] *$\mathcal{P}^{CS}$ will be introduced and studied in [6]. We consider the standard case $A = \mathbb{Z}$. To define $\mathcal{P}^{CS}$ inductively, let W, a, I such that I is a maximal interval of $\mathcal{P}^{CS}(W)$ and $a \in I$. Let $j \in \{1, \dots, r\}$ be maximal such that $a_j \in I$ ("last car that parked on the interval"). Then if $a < a_j$ park left of I, and if $a \geq a_j$ park right: this defines $\mathcal{P}^{CS}(Wa)$.*

So one needs to remember, for each maximal interval, where the last car that parked there had wanted *to park. We indicate this with a red cross above certain cars (= blue dots) in the illustration below for the word $W = 5.11.8.3.9.3.2$, which shows $\mathcal{P}^{CS}(W) = \{2, 3, 4, 5, 8, 9, 11\}$.*



3 Enumeration

We consider $\mathcal{P}$ any local parking procedure, here and in the rest of this abstract.

Definition 7. *A word $a_1 \cdots a_r$ is a* $\mathcal{P}$-parking word *if $\mathcal{P}(a_1 \cdots a_r) = \{1, \dots, r\}$.*

Let $Park(\mathcal{P})$ be the set of parking words for $\$\mathcal{P}$. Classical parking functions/words correspond clearly to the case $\mathcal{P}^{usual}$. The case of $\mathcal{P}^{CS}$-parking words is of particular importance in [6], and was one of the motivations for the present abstract.

Here we fix $i \leq r$. Let $A^i_{[r+1]}$ be the set of words of length i and letters with values in $[r+1] = \{1, \dots, r+1\}$.

Cyclic parking We define $\mathcal{P}^{(r)}$ to be the cyclic version of $\mathcal{P}$, as follows: It is defined on $A^i_{[r+1]}$ for any $i \leq r$ and its image is a subset of $\mathbb{Z}/(r+1)\mathbb{Z}$ of size i. Suppose $\mathcal{P}^{(r)}(W)$ is defined for $i < r$, and pick a letter a with $\mathrm{val}(a) \in [r+1]$. If $\mathrm{val}(a) \notin \mathcal{P}^{(r)}(W)$, park it at $\mathrm{val}(a)$. If not, one checks readily that the conditions **Shift invariance** and **Local decision** allow us to determine unambiguously to park left or right, based on any lift of $\mathcal{P}^{(r)}(W)$ to $\mathbb{Z}$, and thus to define $\mathcal{P}^{(r)}(Wa)$.

If $W \in A^i_{[r+1]}$ and $k \in \{1, \dots, r+1\}$, define $W[k] \in A^i_{[r+1]}$ to be the word obtained by replacing each letter (a, s) with $((\tau^k(a) \mod r+1, s)$. We have the following easy lemma:

[2] The full procedure has extra information $S = \mathbb{N}$ and we only allow words on $A = \mathbb{Z} \times \mathbb{N}$ without repeated letters, in the sense of Remark 2. Letters in $A = \mathbb{Z} \times \mathbb{N}$ are then ordered lexicographically for comparison (note that $a = a_j$ does not occur in this setting).

Lemma 8 ("Pollak's argument"). *Let $W \in A^r_{[r+1]}$ and $k \in \{1, \dots, r+1\}$.*
Then $\mathcal{P}^{(r)}(W) = \mathbb{Z}/(r+1)\mathbb{Z} - \{k\}$ if and only if $W[k]$ is in $Park(\mathcal{P})$.

Now for any $\mathcal{G} \subset A_r^{r+1}$, define $\mathcal{G}[k] = \{W[k] \mid W \in \mathcal{G}\}$. From the previous lemma one obtains:

$$\sum_{k=0}^{r} Park(\mathcal{G}[k]) = |\mathcal{G}|. \tag{1}$$

In particular, if $\mathcal{G}$ is stable under cyclic shift, that is $\mathcal{G} = \mathcal{G}[1]$ and thus $\mathcal{G} = \mathcal{G}[k]$ for all k, we have $Park(\mathcal{G}) = \frac{|\mathcal{G}|}{r+1}$. In the standard case $A = \mathbb{Z}$, we can take $\mathcal{G} = \{1, \dots, r+1\}^r$ and get:

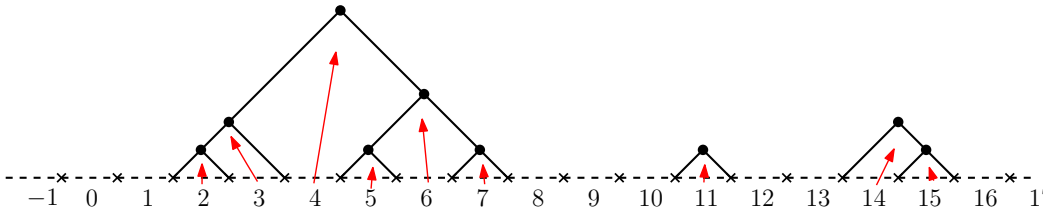
Corollary 9. *For any standard parking procedure $\mathcal{P}$, the number of $\mathcal{P}$-parking words of length r is given by $(r+1)^{r-1}$.*

This shows that the enumeration of parking functions is *universal* for all such standard procedures. For instance, the CS-parking functions are enumerated by $(r+1)^{r-1}$. Note that however there does not seem to be any nice characterization of them, in contrast to the one for usual parking functions recalled in the introduction. More generally, the properties of $\mathcal{P}$-parking functions may differ vastly depending on the procedures, even if their enumeration is the same.

4 Encoding with binary trees

Clearly the parking procedures $\mathcal{P}$ are not injective in general: even if we know π^W, which encodes occupied spots for each prefix of a word W, we cannot reconstruct W. Here we will define a general *lift* of the parking procedure that completely encodes the word W.

Recall that a (finite, plane, binary) *tree* is defined recursively as either empty or consisting of a node, a left subtree and a right subtree. Its size is its number of nodes. These are well-known to be in bijection with *complete* binary trees by attaching extra leaves. A forest is then usually defined as a set of trees. Here we will always mean *indexed* forests, as illustrated below: each binary tree is first completed, and naturally attached to an interval of $\mathbb{Z}$, and these intervals must be the maximal intervals of a finite subset of $\mathbb{Z}$ called the *support* of the forest. The support in the example is $\{2, 3, 4, 5, 6, 7, 11, 14, 15\}$. Elements of the support correspond bijectively to nodes of the forest, as illustrated by the arrows in the figure: this is the *canonical labeling* of the nodes.



Given a procedure $\mathcal{P}$ and a word W, we will recursively attach a pair (P, Q) of labeled forests with the same underlying forest[3] of size r. P will be labeled by the multiset of letters of W, while Q is *decreasing*: it has labels $\{1, \dots, r\}$ and each node has greater label than all its descendants.

Construction Assume $(P', Q') = \widehat{\mathcal{P}}(W)$ where W has size r. By induction we have that the common support of $\widehat{\mathcal{P}}(W)$ is the subset $\mathcal{P}(W)$ of size r. Consider a letter a, and let $i = \mathrm{ls}(Wa)$. We distinguish three cases in order to define P:

1. *$\mathcal{P}(Wa)$ has one more interval than $\mathcal{P}(W)$.* This interval is necessarily the singleton $\{i\}$. In this case add to P' a single node tree canonically labeled by i, and label it by a: this defines P.

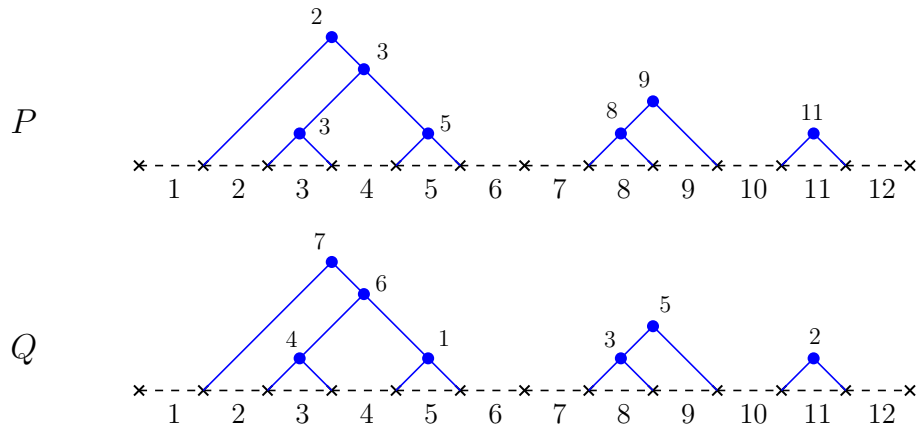
[3] Equivalently, one could of course gather the two labelings on the underlying shape.

2. $\mathcal{P}(Wa)$ *has one fewer interval than* $\mathcal{P}(W)$. So i is adjacent to two intervals of $\mathcal{P}(W)$, corresponding inductively to two trees of $\widehat{\mathcal{P}}(W)$. We add a new root labeled a in P' and attach to it the two subtrees. This gives P, in which the new node has canonical labeling i.
3. $\mathcal{P}(Wa)$ *and* $\mathcal{P}(W)$ *have the same number of intervals.* In this case i is on the left or right of an interval of $\mathcal{P}(W)$, but not adjacent to any other interval. We add a new root labeled a in P' and attach to it the subtree corresponding to the interval: it is attached as a right subtree if i is on the left, and as a left subtree if i is on the right.

In all cases, Q is obtained from Q' by labeling the new node by $r+1$. It is immediate by that Q is decreasing.

Definition 10. *The encoding* $\widehat{\mathcal{P}} : A^* \mapsto (P, Q)$ *is called the* $\mathcal{P}$*-correspondence.*

Here is this correspondence for $\mathcal{P}^{CS}$ with the word $W = 5.11.8.3.9.3.2$ from our previous example:



The encoding $\widehat{\mathcal{P}}$ can be thought of as recording the full history of the parking process. Let us list some immediate properties for any local procedure $\mathcal{P}$:

- A word is $\mathcal{P}$-parking if and only if it results in a single tree with support $\{1, \ldots, r\}$ via the $\mathcal{P}$-correspondence.
- The $\mathcal{P}$-correspondence is injective: the word W can be reconstructed by reading off the labels of P in the increasing order given by the labeling of Q.
- The canonical labeling of a node is the spot where the corresponding car ended up parking.

Note that the conditions **Shift invariance** and **Local decision** are not necessary for this construction and properties. Roughly speaking, they determine what kind of P-labelings may occur in the image.

The correspondence is particularly nice if for any P in the image, all Q with same underlying tree give a pair (P, Q) in the image of $\widehat{\mathcal{P}}$. In that case the image of $\widehat{\mathcal{P}}$ is determined once we know what labeled trees P can occur.

This is the case for our running examples: For $\mathcal{P}^{usual}$, a valid labeling of a node is the canonical label of one of its left descendants (or its own). For $\mathcal{P}^{CS}$, a valid labeling of a node has to be weakly larger than the label of its left child, and strictly smaller than the label of its right child (if such children are not present, use instead the canonical label of the node).

5 Extensions

5.1 Probabilistic parking

To add probabilities to the setting, one can ask how likely it is for a word to be parking. One can also study properties of $\mathcal{P}$-parking words picked uniformly at random for instance. In the classical case this

is very natural from the hashing viewpoint, see [2, 3].

It is also possible to probabilize the procedures themselves, by having $\mathcal{P}$ associate to each word W a finite probability measure on $\mathrm{Fin}(\mathbb{Z})$. The local requirements extend naturally to define such procedures, as do most results given in this abstract. One may then ask: what is the probability of a fixed W to be parking ?

Such a procedure is defined at the end of [4]: one flips a (fixed) coin to determine to park left or right. Another possibility is to do a random walk on the interval that is hit, so that parking on the left/right depends on the interval size, and where one desires to park. This leads to the rich combinatorics of *remixed Eulerian numbers*, introduced by the author with Vasu Tewari [7].

5.2 Special subclasses

A natural subclass of parking procedures consists of those where the parking decisions only depend on the set of occupied spots:

Definition 11. *A parking procedure $\mathcal{P}$ is called* Markovian *if there exists a function $M : \mathrm{Fin}(\mathbb{Z}) \times A \to \mathrm{Fin}(\mathbb{Z})$ such that $\mathcal{P}(a_1 \cdots a_r a) = M(\mathcal{P}(a_1 \cdots a_r), a)$ for any letters $a_1, \cdots, a_r, a$.*

Clearly $\mathcal{P}^{usual}$ is Markovian, while $\mathcal{P}^{CS}$ is not (one needs the red crosses in the figure).

Definition 12. *A parking procedure $\mathcal{P}$ is called* abelian *if $\mathcal{P}(a_1 \cdots a_r) = \mathcal{P}(a_{\sigma_1} \cdots a_{\sigma_r})$ for any letters $a_1, \ldots, a_r$ and any permutation σ.*

$\mathcal{P}^{usual}$ is abelian, as can be seen immediately from the characterization of parking words in the introduction. $\mathcal{P}^{CS}$ is not, as 112 is parking while 121 is not. It would be interesting to determine other abelian procedures, as these are probably quite rare. One can for instance show that if a standard procedure $\mathcal{P}$ is abelian and Markovian, it is either $\mathcal{P}^{usual}$ or the symmetric "always go left" version.

References

[1] A. Christensen, P. E. Harris, Z. Jones, M. Loving, A. Ramos Rodriguez, J. Rennie, and G. Rojas Kirby, A generalization of parking functions allowing backward movement, *Electron. J. Combin.* **27(1)** (2020), #P1.33.

[2] P. Diaconis and A. Hicks, Probabilizing parking functions, *Adv. in Appl. Math.* **89** (2017), 125–155.

[3] P. Flajolet, P. Poblete and A. Viola , On the Analysis of Linear Probing Hashing, *Algorithmica.* **22** (1998), 490–515.

[4] A.G. Konheim and B. Weiss, An occupancy discipline and applications, *SIAM J. Appl. Math.* **14** (1966), 1266–1274.

[5] R. P. Stanley and J. Pitman. A polytope related to empirical distributions, plane trees, parking functions, and the associahedron. *Discrete Comput. Geom.* **27(4)**, (2002), 603–634.

[6] P. Nadeau and V. Tewari, Schubert coefficients for the permutahedral variety, *in preparation.*

[7] P. Nadeau and V. Tewari, A q-deformation of an algebra of Klyachko and Macdonald's reduced word formula, preprint, 2021, `arxiv:2106.03828`.

[8] C. H. Yan, Parking functions, in *Handbook of enumerative combinatorics*, Discrete Math. Appl. (Boca Raton), CRC Press, Boca Raton, FL, 2015, 835–893.