GRAPH SEARCHING AND A GENERALIZED PARKING FUNCTION

A Dissertation

by

DIMITRIJE NENAD KOSTIĆ

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2007

Major Subject: Mathematics

GRAPH SEARCHING AND A GENERALIZED PARKING FUNCTION

A Dissertation

by

DIMITRIJE NENAD KOSTIĆ

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2007

Major Subject: Mathematics

ABSTRACT

Graph Searching and a Generalized Parking Function. (August 2007)

Dimitrije Nenad Kostić, B.A., Lawrence University;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Catherine H. Yan

Parking functions have been a focus of mathematical research since the mid-1970s. Various generalizations have been introduced since the mid-1990s and deep relationships between these and other areas of mathematics have been discovered. Here, we introduced a new generalization, the *G-multiparking function*, where G is a simple graph on a totally ordered vertex set $\{1, 2, \ldots, n\}$. We give an algorithm that converts a G-multiparking function into a rooted spanning forest of G by using a graph searching technique to build a sequence $F_1, F_2, \ldots, F_n$, where each F_i is a subforest of G and F_n is a spanning forest of G. We also give another algorithm that converts a rooted spanning forest of G to a G-multiparking function and prove that the resulting functions (between the sets of G-multiparking functions and rooted spanning forests of G) are bijections and inverses of each other. Each of these two algorithms relies on a *choice function* ζ, which is a function from the set of pairs (F, W) (where F is a subforest of G and W is a set of some of the leaves of F) into W. There are many possible choice functions for a given graph, each giving formality to the concept of choosing how a graph searching algorithm should procede at each step (i.e. if the algorithm has reached F_i, then F_{i+1} is some forest on the vertex set $V(F_i) \cup \{\zeta(F_i, W)\}$ for some W).

We also define *F-redundant edges*, which are edges whose removal from G does not affect the execution of either algorithm mentioned above. This concept leads to a categorization of the edges of G, which in turn provides a new formula for the Tutte polynomial of G and other enumerative results.

To Mom, Dad, and Bogdan

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Catherine Yan, for the thoughtful mentoring and care she took in guiding my research, as well as for her financial support. Many thanks to Rob Ellis and Jeremy Martin for our invigorating conversations and for their friendship. Professors Paula Tretkoff and Matt Papanikolas helped me discover a love of number theory that, although it may not be evident in this work, captivates me still and I am grateful to both of them. I thank Professor Maurice Rojas for his kindness, friendship, for his support and for patiently teaching me about algebraic geometry. I warmly remember my four years at Lawrence University where I first learned how to learn, and the many professors who taught me so many important lessons. Thanks go to the Budapest Semesters in Mathematics for four of the most enjoyable months of my life. Finally, I wish to thank *Square One Television*, a long-defunct but excellent television program, which aired on PBS for several years in the late '80s and early '90s, for first introducing me to the enigmatic wonders of mathematics.

Mom and dad have been an enthusiastic and unwavering source of support of every kind. For more than twenty years of formal education the two of you have stood stridently in my corner, cheering my every success and pushing me along after every setback. Bogdan has been a source of precocious and down-to-earth wisdom one rarely finds in a younger brother. I am also grateful to my cousins Simon Kostić and Draginja Jeftović, to my aunt Gordana and uncles Simeon and Miki, and the departed members of my family who would have been proud.

Over the years I have been fortunate to call many admirable people good friends, but few have had the constancy or the loyalty that my good friends from high school, Steve Gartz, John Hinrichsen, Ryan Kelly, Nishant Parulekar, Nick Schmerr, Joe Stansbery, and Elton Wong, have had. The seven of you have been a tremendous source of encouragement and support for (collectively) more than a hundred years. Each of you and your families have given me the sort of genuine friendship that, I am convinced, few people are lucky enough to have from one person.

I am also grateful that several friends from Lawrence have maintained an interest in me, namely June Erwin and Mike Passow, Joanna Messer, Elizabeth Gimpel, and Adam Lake.

I am deeply grateful to Jim Ruffo, whose plainspoken good advice is as valuable to me as the warm and trusting friendship we have developed. I am glad to have met and befriended Matthias Bühlmaier, Tanja Djokić, Dariush Ehsani, Mike Flanagan, Jenny and Eddie Fuselier, Ashraf Ibrahim, Alexis Koskan, Goran Latisko, Svenja Lowitzsch, Fabien Marpeau, Mariana Pereira, Johannes Ruf, Kristen Russell, Maša Šuković, Gabriel Tucci, Valentina Vega, and George Welch and his family. If I am a good person, it is because I am so fortunate to be surrounded by friends like you.

I wish to thank the staff of the Texas A&M Math Department, particularly Sherry Floyd and Monique Stewart, for accommodating my many administrative and clerical needs so well and so patiently throughout my graduate studies and for the generous financial support they have provided me.

Finally, I wish to thank Bertrand Russell and Noam Chomsky, two men whose writings have had a profound effect on me.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

CHAPTER I

BACKGROUND AND INTRODUCTION

The purpose of this chapter is to introduce classical parking functions, their generalizations, and survey major results in this field. We will generally eschew proofs, except where they are short and provide some intuition.

A. Notation

The following notation will be employed throughout this dissertation. We let $\mathbb{N}$ denote the non-negative integers and $\mathbb{P}$ the positive integers. We use $[n]$ to denote $\{1, 2, \ldots, n\}$. The symmetric group on n letters will be denoted S_n.

B. Classical Parking Functions

A *parking function* of length n is a sequence $(a_1, a_2, \ldots, a_n)$ of non-negative integers for which there exists a permutation $\pi \in S_n$ such that $0 \leq a_{\pi(i)} \leq i-1$ for all indexes i. In other words, if we choose a permutation π such that $a_{\pi(1)} \leq a_{\pi(2)} \leq \ldots \leq a_{\pi(n)}$ then we have $(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}) \leq (0, 1, \ldots, n-1)$. This increasing rearrangement is sometimes called the *order statistic* of the sequence. (Some authors prefer to let the a_i be positive; that is, to define $1 \leq a_{\pi(i)} \leq i$. This is an unimportant distinction, and everything that will be said below can easily be modified to fit this terminology.) It is immediate from the definition that any rearrangement of the entries of a parking function is also a parking function. The fact that these entries are not necessarily unique makes counting the number of parking functions of a given length more difficult

The journal model is Advances in Computational Mathematics.

than counting the size of the symmetric group S_n, but we will soon see that this is still not very difficult.

A more engaging way to think of parking functions (and one which justifies the nomenclature) is as follows. Consider a one-way street with n empty parking spaces. Suppose n cars enter the street, single file, and each driver independently chooses a spot in which she would like to park. (The i^{th} driver's preferred parking spot is designated a_i.) The first driver to enter the street may simply park wherever she chooses. Every subsequent driver will drive up the street to the desired spot, and park there if it is unoccupied. Otherwise, she must drive further up the street and park in the first vacant spot she comes across. If no such spot is available, then she will have to leave. Then, $(a_1, a_2, \ldots, a_n)$ is a *parking function* if every driver can find a parking spot. Computer scientists might view this as a linear probe with no "wrap-around".

This concept was first defined in 1966 by Konheim and Weiss in [15].

Example 1. If $n = 2$, then there are three parking functions: $(0, 0)$, $(1, 0)$, and $(0, 1)$. If $n = 3$, then there are sixteen parking functions: $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$, $(0, 0, 2)$, $(0, 2, 0)$, $(2, 0, 0)$, $(0, 1, 1)$, $(1, 0, 1)$, $(1, 1, 0)$, $(0, 1, 2)$, $(0, 2, 1)$, $(1, 0, 2)$, $(1, 2, 0)$, $(2, 0, 1)$, and $(2, 1, 0)$.

The attentive combinatorialist will guess (correctly) that the number of parking functions of length n is given by the *Cayley number* $(n + 1)^{n-1}$. There are several ways to prove this fact, and it happens to be a special case of theorem 12 in section II. There is a simple way to prove this fact. Imagine that the one-way street is circular, and that there is an $n+1^{st}$ parking spot between the 1^{st} and n^{th} spots. We allow our n cars to prefer any one of these $n+1$ spots. Since now the cars can "wrap around", all n cars will always be able to park, and there will be exactly one empty spot. This empty spot can be any one of the $n+1$ spots, and if it is the $n+1^{st}$ spot we inserted, then our preference sequence is a parking function. Equivalently, if $(a_1, a_2, \ldots, a_n)$ is the preference sequence, then there exists a unique i between 1 and $n+1$ such that $(a_1+i, a_2+i, \ldots, a_n+i)$ is a parking function. As we have $(n+1)^n$ possible preference sequences, and they are grouped into equivalence classes each of size $n+1$, we have exactly $\frac{(n+1)^n}{n+1} = (n+1)^{n-1}$ parking functions.

The Cayley numbers are more famous for enumerating another sequence of sets. Let $V = V(G)$ be a set and let $E = E(G)$ be a set of subsets of V, each of size 2. The set V is called the set of *vertices* and E the set of *edges*. The ordered pair $(V(G), E(G))$ is called the *graph* G. When V is visualized as a set of dots in the plane and E as lines connecting these dots, a number of other graph-theoretic notions become easy to understand. A *path* between two vertices is a sequence of edges from one to the other. A graph is *connected* if between every pair of vertices there is a path. A *cycle* in G is a path that begins and ends at the same vertex. A *tree* is a connected graph with no cycles. In fact, the following classical theorem provides multiple ways to think of a tree.

Theorem 1. Any two of the following conditions implies the third, and trees are exactly the graphs exhibiting them.
(1) G is connected.
(2) G has no cycles.
(3) $|V(G)| - 1 = |E(G)|$.

Below in Figure 1 we illustrate all the trees on three and four vertices. The vertices are labelled, but we have suppressed the labels.

It is a classical result that the Cayley numbers enumerate the set of trees on $n+1$ vertices. In the mid-1970's several papers, such as [10], exploited this fact to

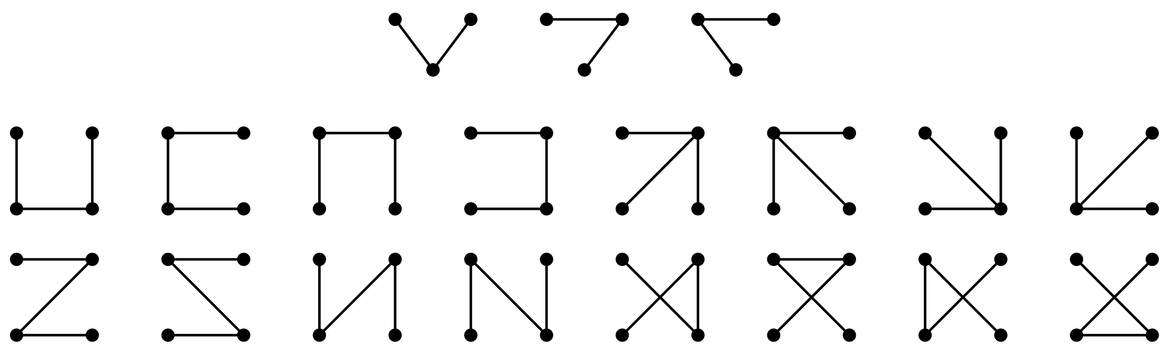


Fig. 1. All labelled trees on three and four vertices.

create bijections between the set of parking functions of length n and trees on $n+1$ vertices.

A fascinating connection exists between classical parking functions and the theory of noncrossing partitions. If $S = \{1, 2, \ldots, n\}$, a *partition* of S is a set of subsets $P = \{A_1, A_2, \ldots, A_k\}$ of S which are pairwise disjoint and whose union is S. A partition is *noncrossing* if whenever $a, c \in A_i$ and $b, d \in A_j$ have the property that $a < b < c < d$, we have $A_i = A_j$. The noncrossing partitions of a set S have a canonical poset order (often called the order given by *refinement*): if P_1 and P_2 are noncrossing partitions of S, then $P_1 \leq_{NC} P_2$ if for every $A \in P_1$ there exists $B \in P_2$ such that $A \subseteq B$.

If $S = \{1, 2, 3, 4\}$ then there is only one crossing partition: $13|24$. Below we illustrate the lattice of noncrossing partitions of S under the order $\leq_{NC}$.

For any $S = [n]$, $\mathrm{NC}(n)$ always has a maximal element $\hat{1}$ (given by the partition $P = 123\ldots n$) and a minimal element $\hat{0}$ (given by the partition $P = 1|2|\ldots|n$). In Figure 2, one can easily see that there are 16 paths (more commonly called *maximal chains*) between $\hat{0}$ and $\hat{1}$.

In fact, in [27] Stanley discovered a bijective labelling of the maximal chains of $\mathrm{NC}(n+1)$ using the parking functions of length n. Noncrossing partitions play an

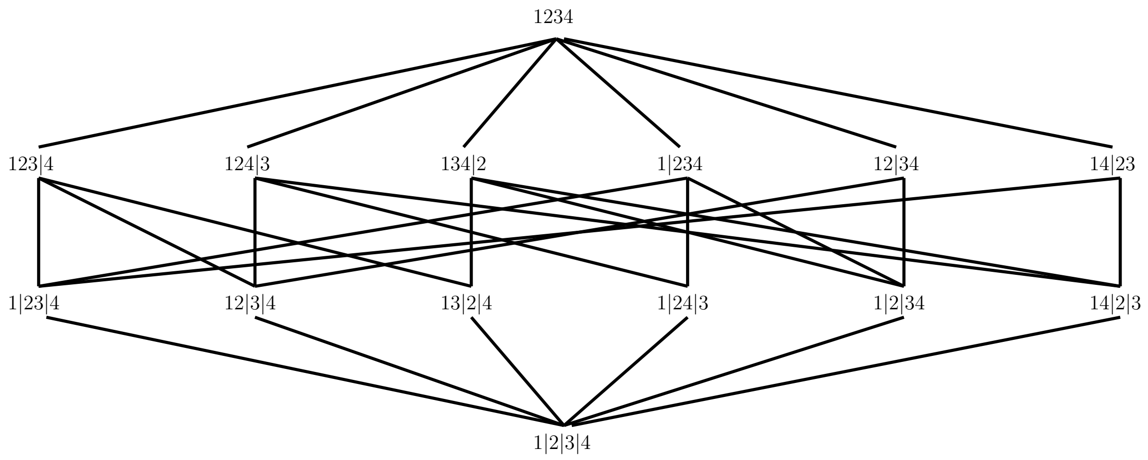


Fig. 2. The lattice NC(4).

important role in *free probability theory*, which studies a certain analog of independence for noncommutative random variables. The *free cumulant* functional can be defined as

$$\mu(X_1, X_2, \ldots, X_n) := \sum_P \prod_{B \in P} \kappa(X_i | i \in B)$$

where the product ranges over all noncrossing partitions P of $[n]$.

C. Generalizations

At least as a branch of combinatorics, the theory of parking functions lay mostly dormant between the mid-1970's and the early 1990's. Interest in the theory was revitalized with the introduction of a number of generalizations, each with its own set of distinct and interesting results, some with implications reaching far beyond combinatorics. Several of these generalizations will resurface in subsequent chapters, and this section will also outline some of the main results from the theory of generalized parking functions.

1. x-Parking Functions

Let $x = (x_1, x_2, \ldots, x_n) \in \mathbb{N}^n$. An *$x$-parking function* is a sequence $(a_1, a_2, \ldots, a_n)$ whose order statistic satisfies $1 \le a_{\pi(i)} \le x_1 + \ldots + x_i$. It is clear that if $1 = x_1 = x_2 = \ldots = x_n$, then the x-parking functions are exactly the parking functions of length n, and that there are no x-parking functions if $x_1 = 0$. This definition was first given by Pitman and Stanley in [23]. They proved that if $P_n(x)$ denotes the number of x-parking functions and $\mathrm{park}(n)$ denotes the set of ordinary parking functions of length n, then

$$P_n(x) = \sum_{(a_1,\ldots,a_n)\in \mathrm{park}(n)} x_{a_1} x_{a_2} \ldots x_{a_n} = n! V_n(x)$$

where $V_n(x)$ is the volume of a certain polytope called the *associahedron.* An important special case of this identity is

$$P_n(1, q, q^2, \ldots, q^{n-1}) = \sum_{(a_1,\ldots,a_n)\in \mathrm{park}(n)} q^{a_1+a_2+\ldots+a_n-n}$$

The *inversion enumerator* of the complete graph on the vertex set $\{1, 2, \ldots, n\}$ is defined as

$$I_n(q) := \sum_T q^{\mathrm{inv}(T)}$$

where the sum ranges over all spanning trees T of K_n and where $\mathrm{inv}(T)$ is the number of edges $\{i, j\} \in E(T)$ such that $i > j$ and i is closer to the vertex 1 along the unique path in T from 1 to i to j. A result of Kreweras in [16] implies that

$$n! V_n(1, q, q^2, \ldots, q^{n-1}) = q^{\binom{n}{2}} I_n(1/q)$$

We pause here to note some of the classical results on the inversion enumerator which can be generalized (see section 2).

Theorem 2.

1. ([14], [16], [22]) $I_n(1+q) = \sum_{G} q^{|E(G)|-n}$
   where G ranges over all connected simple graphs on $n+1$ labelled vertices.
2. ([12]) $\displaystyle\sum_{n\geq 0} I_n(q)(q-1)^n \frac{x^n}{n!} = \frac{\sum_{n\geq 0} q^{\binom{n+1}{2}} \frac{x^n}{n!}}{\sum_{n\geq 0} q^{\binom{n}{2}} \frac{x^n}{n!}}$
3. ([16]) $q^{\binom{n}{2}} I_n(1/q) = \displaystyle\sum_{(a_1,\ldots,a_n)} q^{a_1+\ldots+a_n}$
   where $(a_1, \ldots, a_n)$ ranges over all parking functions of length n.
4. ([16]) Given $I_1(q) := 1$, the inversion enumerator satisfies the recurrence
   $I_n(q) = \displaystyle\sum_{i=0}^{n} \binom{n}{i} \left(\sum_{j=0}^{i} q^j \right) I_i(q) I_{n-i}(q)$

Another interesting formula Stanley and Pitman were able to prove was that if $l, k \geq 1$ and $x = (l, k, k, \ldots, k)$ is a vector of length n then $P_n(x) = l(l+nk)^{n-1}$. In [31], Yan found an interesting interpretation of this formula. A *rooted k-forest* on the vertex set $\{1, 2, \ldots, n\}$ is a rooted forest whose edges are each colored in one of k colors. Then, there is a bijection between the x-parking functions and the sequences $(\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k)$ where each $\mathcal{T}_i$ is a rooted k-forest, $\mathcal{T}_i$ and $\mathcal{T}_j$ are disjoint if and only if $i \neq j$, and the union of the vertex sets of the $\mathcal{T}_i$ is $\{1, 2, \ldots, n\}$.

2. k-Parking Functions

This particular generalization will not play a large role in the subsequent chapters of this dissertation, but we include discussion here because their relationship to hyperplane arrangements is an interesting and important one.

Let $k \in \mathbb{N}$. A *k-parking function* of length n is a sequence $(a_1, a_2, \ldots, a_n)$ whose order statistic satisfies $1 \leq a_{\pi(i)} \leq ki$. Thus, the 1-parking functions of length n are exactly the usual parking functions of length n.

In [28], Stanley showed that k-parking functions are closely related to a certain hyperplane arrangement. If $n, k \geq 1$ the *extended Shi arrangement* S_n^k is the collection of hyperplanes given by the equations

$$x_i - x_j = -k+1, -k+2, \ldots, k \text{ where } 1 \leq i < j \leq n$$

Example 2. The Shi arrangement S_3^2, viewed from different perspectives. Note that each hyperplane depicted extends infinitely.

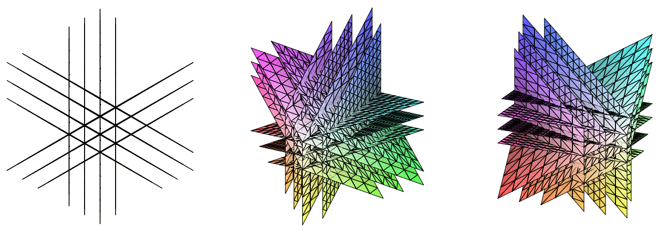


Fig. 3. Some views of S_3^2.

We define the *regions* of a hyperplane arrangements to be the path-connected components of the complement (in the ambient space $\mathbb{R}^n$). (One can easily see these regions in the leftmost image in Example 2.) It turns out that there is a natural way to label the regions cut out by these hyperplanes with the k-parking functions. We choose one of these chambers, the one whose points satisfy the inequalities $x_1 > x_2 > \ldots > x_n > x_1 - 1$, to be the "base" region and we label it R_0. There is then a notion of distance between a region R and R_0: given a straight line connecting a point in R and a point in R_0, let $d(R)$ be the number of hyperplanes the line intersects. It is not difficult to see that $d(R)$ is well-defined and since S_n^k has only finitely many regions, $d(R) < \infty$. We now define labels λ for each region R of S_n^k. First set $\lambda(R_0) := (0, 0, \ldots, 0)$. If R' is a labelled region separated from an unlabelled region

R by the hyperplane $x_i - x_j = m$, then

$$\lambda(R) := \begin{cases} \lambda(R') + e_i & \text{if } m > 0 \\ \lambda(R') + e_j & \text{if } m \leq 0 \end{cases}$$

It turns out that the $\lambda(R)$ are exactly the k-parking functions of length n, giving a bijection.

This correspondence, while interesting, reveals little about k-parking functions as combinatorial objects. Recall that in section 1 above, we noted that

$$I_n(q) = \sum_{(a_1,\ldots,a_n)} q^{a_1+\ldots+a_n-n}$$

where $I_n(q)$ is the inversion enumerator of K_n. We now introduce a generalization of the inversion enumerator. If F is a rooted b-forest, as defined in section 1, set

$$l(F) := \text{inv}(F) + \sum_{(v,e)} \kappa(e)$$

where $\text{inv}(F)$ is the total number of inversions (disregarding the color of the edges) on the trees of F, where the sum ranges over all pairs (v, e) where e is an edge on the unique path between the vertex v and the root of the tree of F on which it sits, and where $\kappa(e)$ is the color of the edge e. We then define the k-inversion enumerator

$$I_n^k(q) := \sum_F q^{l(F)}$$

where the sum ranges over all rooted k-forests F whose vertex set is $[n]$.

Theorem 3. The following hold for any $k \geq 1$.

1. ([28]) $I_n^k(1+q) = \sum_G q^{e(G)+r(G)-n}$ where G ranges over multirooted b-graphs on [n]

2. ([28]) $\sum_{n\geq 0} I_n^k(q)(q-1)^n \frac{x^n}{n!} = \frac{\sum_{n\geq 0} q^{k\binom{n}{2}+n} \frac{x^n}{n!}}{\sum_{n\geq 0} q^{k\binom{n}{2}} \frac{x^n}{n!}}$
3. ([28]) $q^{k\binom{n}{2}} I_n(1/q) = \sum_{(a_1,\ldots,a_n)} q^{a_1+\ldots+a_n}$ where $(a_1,\ldots,a_n)$ ranges over all k-parking functions.
4. ([30]) Given $I_1^k(q) := 1$, the k-inversion enumerator satisfies the recurrence $I_{n+1}^k(q) = \sum_{a_0+a_1+\ldots+a_k=n} \binom{n}{a_0, a_1, \ldots, a_k} q^{a_1+2a_2+\ldots+(k-1)a_{k-1}} \left(\sum_{i=0}^{n-a_k} q^i \right) \prod_{i=0}^{b} I_{a_i}^k(q)$ where $a_0, a_1, \ldots, a_k$ are non-negative integers.

These identites generalize the classical results for $I_n(q)$ (see the corresponding results in theorem 2 above).

Let NC_{n+1}^k be set of noncrossing partitions where k divides the size of every block. This set also has a poset structure, again under the refinement order (note that it is not a lattice if $k > 1$, since it has no $\hat{0}$ element). In [27], Stanley proved that k-parking functions enumerate the maximal chains in NC_{n+1}^k, by a labelling scheme similar to that for ordinary parking functions.

3. (p, q)-Parking Functions

In [7], Cori and Poulalhon proposed a new generalized parking function apparently intended to modify the parking analogy introduced above. Given positive integers p and q, a (p,q)-*sequence* is a sequence $(u,v) = (u_1,\ldots,u_p,v_1,\ldots,v_q)$ such that for all $i \leq p$ and $j \leq q$, $0 \leq u_i \leq q$ and $0 \leq v_i \leq p$. Clearly, there exists a partial order $\leq_{(p,q)}$ on (p,q)-sequences in which $(u_1,v_1) \leq_{p,q} (u_2,v_2)$ if $u_1 \leq u_2$ and $v_1 \leq v_2$ in the usual lexicographic order. Given any permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_p, \sigma_{p+1},\ldots,\sigma_{p+q} \in S_{p+q}$, we can define a (p,q)-sequence $(u^\sigma, v^\sigma) = (u_1^\sigma,\ldots,u_p^\sigma, v_{p+1}^\sigma,\ldots,v_{p+q}^\sigma)$ where

$$u_i^\sigma := |\{\sigma_{p+j} \mid 1 \leq j \leq q,\ \sigma_{p+j} < \sigma_i\}| \quad \text{if } 1 \leq i \leq p$$
$$v_i^\sigma := |\{\sigma_j \mid 1 \leq j \leq p,\ \sigma_j < \sigma_i\}| \quad \text{if } p+1 \leq i \leq p+q$$

A *(p,q)-parking function* is a (p,q)-sequence (u,v) such that there exists $\sigma \in S_{p+q}$ with $(u,v) \leq_{p,q} (u^\sigma, v^\sigma)$. The permutation σ giving (u^σ, v^σ) is called a *certificate* for (u,v). An example will help demystify these definitions.

Example 3. Let $p=4$ and $q=3$. To see that $(u,v) = (0001341)$ is a $(4,3)$-parking function, we need to find a $\sigma \in S_{4+3} = S_7$ such that $(u,v) \leq_{4,3} (u^\sigma, v^\sigma)$. Consider $\sigma = 1524763 \in S_7$. Using the above formulae, we can see that $(u^\sigma, v^\sigma) = (0101442)$, and since $(0001341) \leq_{4,3} (0101442)$, we have found a (p,q)-parking function.

The permutation σ in the above example is not unique; we could also have chosen 1524367, 1245736, or many other permutations. In fact, an entire family of certificates for (u,v) can be obtained from the usual action of $S_p \times S_q$ on 1524763. It is not difficult to see that the set of (p,q)-parking functions is invariant under the action of $S_p \times S_q$.

Cori and Poulalhon also introduced an interesting parking analogy. Consider $p+q$ cars, p of them blue and q of them red, all in line to park on a one-way street. The (p,q)-sequence $(u,v) = (u_1, \ldots, u_p, v_{p+1}, \ldots, v_{p+q})$ is a (p,q)-parking function if there exists a way for these cars to park so that the i^{th} blue car has exactly u_i red cars parked behind him, and the j^{th} red car has exactly v_{p+j} blue cars parked behind him.

Despite the apparently contrived definitions given, (p,q)-parking functions are closely related to the classical parking functions via the following two propositions, both proven in [7].

Proposition 4. A (p,q)-sequence (u,v) is a (p,q)-parking function if and only if the concatenation of u and v is a parking function of length $p+q$.

Proposition 5. A sequence u of length n is a parking function if and only if (u,u) is an (n,n)-parking function.

Cori and Poulalhon also prove that there are $(p+q+1)(p+1)^{q-1}(q+1)^{p-1}$ (p,q)-parking functions and $\frac{1}{p+q+1}\binom{p+q+1}{p}\binom{p+q+1}{q}$ (p,q)-parking functions whose values are strictly increasing. (In algebraic terms, the action of $S_p \times S_q$ on the set of (p,q)-parking functions induces exactly $\frac{1}{p+q+1}\binom{p+q+1}{p}\binom{p+q+1}{q}$ equivalence classes.)

4. G-Parking Functions

This generalization is perhaps the most abstract of the four, and also the one that will be of most interest in the upcoming chapters, so we will devote more space to it here. Postnikov and Shapiro introduced in [24] the following generalization. A *directed graph* $G = (V(G), E(G))$ is a graph in which each element of $E(G)$ is a length-2 vector (i,j). (Edges are thus thought of as being arrows from i to j.) We define the *outdegree* of a vertex i in a subset $U \subseteq V(G)$ to be $\mathbb{O}_U(i) := \#\{j \notin U | (i,j) \in E(G)\}$. Let $V(G) = \{0, 1, 2, \ldots, n\}$. A function $f : V(G) \to \mathbb{N}$ into the non-negative integers is called a *G-parking function* if for every $U \subseteq \{1, 2, \ldots, n\}$ there exists an $i \in U$ such that $f(i) < \mathbb{O}_U(i)$. (Later, in chapter II, we will define G-parking functions for undirected graphs.) If $i \in U$ exhibits $f(i) < \mathbb{O}_U(i)$, we say i is *well-behaved in U*. The vertex 0 will often be called the *root* of the function f.

Example 4. The illustration below depicts a G-parking function f for the illustrated digraph. The vertices are labelled $f(i), \boldsymbol{i}$ where i is the vertex label. The vertex 0 is simply labelled as such.

The main enumerative result on G-parking functions is that they are in bijection to the (labelled) spanning trees of G. Postnikov and Shapiro proved this indirectly, by putting G-parking functions in bijection to another set of objects known as *critical configurations* (and also by other names), which will be explained in more detail in chapter III, section B. Work of Gabrielov in [11] proved that critical configurations

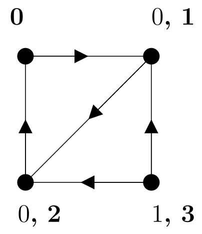


Fig. 4. An example of a G-parking function.

are in bijection to spanning trees. Chebikin and Pylyavvskyy gave an algorithm in [5] which, subject to a certain parameter called a *proper set of tree orders*, outputs an explicit bijection between the set of G-parking functions and the set of spanning trees of G. Changing the proper set of tree orders will in general induce a different bijection.

For parking functions generalized to graphs, such as the G-parking (and, as we will later see, the G-multiparking function and the $(\mathbf{X}, \mathbf{Y})$–parking function), there is a *burning algorithm* to determine, in $O(\#V(G))$ number of operations, whether a vertex function is a generalized parking function. The algorithm was originally stated in [9] and later adapted to this scenario in [5]. A burning algorithm can be thought of as any procedure which establishes an ascending chain $A_1 \subsetneq A_2 \subsetneq \ldots \subsetneq A_{\#V(G)} = V(G)$ of vertex subsets, each containing a well-behaved (in the appropriate sense) vertex. In order for the existence of such a chain to prove that *every* vertex subset has a well-behaved vertex, an analog of the following lemma must hold true.

Lemma 6. If $v \in W \subseteq U$ and v is well-behaved in U, then v is also well-behaved in W.

Proof. Note that $\mathbb{O}_U(v) \leq \mathbb{O}_W(v)$ and therefore $x_{\mathbb{O}_U(v)} \leq x_{\mathbb{O}_W(v)}$ and $y_{\mathbb{O}_U(v)} \leq y_{\mathbb{O}_W(v)}$. The conclusion follows. $\square$

One begins the algorithm by finding a well-behaved vertex i_1 in $V(G) - \{0\}$. If it does not exist, the function is not a G-parking function. If it does, then find a well-behaved vertex i_2 in $V(G) - \{0, i_1\}$. If it does not exist, the function is not a G-parking function. If it does, then find a well-behaved vertex i_3 in $V(G) - \{0, i_1, i_2\}$, and so forth until we have "removed" all vertices from consideration. Our function is a G-parking function if and only if at the end of this process we have removed all vertices from consideration. More formally,

Proposition 7. A vertex function is a G-parking function if and only if there exists an ordering $\pi(1), \pi(2), \ldots, \pi(n)$ of the vertices of a graph G such that for every j, $\pi(j)$ is well-behaved in $U_j := \{\pi(j), \ldots, \pi(n)\}$. (The permutation π will be called a *certificate* for f.)

Proof. It is clear from the definition that for any G-parking function some order π exists. Suppose, conversely, that π exists for some vertex function. If $U \subseteq V(G) - \{0\}$, let k be the maximal index such that $U \subseteq U_k := \{\pi(k), \ldots, \pi(n)\}$. This implies that $\pi(k) \in U$ and lemma 6 implies that $\pi(k)$ is well-behaved in U. Since U was arbitrary, the function is a G-parking function. □

Let $\mathcal{R} := \mathbb{K}[x_1, x_2, \ldots, x_n]$ be a polynomial ring. Define the monomial ideal $\mathcal{I}_G := \langle m_I | \emptyset \subsetneq I \subseteq [n] \rangle$, where $m_I := \prod_{i \in I} x_i^{\mathbb{O}_U(i)}$. We can also define another monomial ideal $\mathcal{J}_G := \langle p_I | \emptyset \subsetneq I \subseteq [n] \rangle$, where $p_I := (\sum_{i \in I} x_i)^{D_I}$ and $D_I := \sum_{i \in I} \mathbb{O}_U(i)$. One of Postnikov and Shapiro's main results is that the algebras $\mathcal{R}/\mathcal{I}_G$ and $\mathcal{R}/\mathcal{J}_G$ are finite dimensional as linear spaces over $\mathbb{K}$ and their dimensions both equal the number $\tau(G)$ of spanning trees of G.

CHAPTER II

G-MULTIPARKING FUNCTIONS AND SPANNING FORESTS

A. G-Multiparking Functions

In the previous section, we defined G-parking functions where G was a directed graph. Here, for various technical reasons, we will consider G to be an undirected graph unless otherwise noted. For any subset $U \subseteq V(G)$, and vertex $v \in U$, we define $\mathbb{O}_U(v)$ to be the cardinality of the set $\{\{v, w\} \in E(G) | w \notin U\}$. Here $E(G)$ is the set of edges of G.

Definition 1. Let G be a simple graph with $V(G) = [n]$. A *G-multiparking function* is a function $f : V(G) = [n] \to \mathbb{N} \cup \{\infty\}$, such that for every $U \subseteq V(G)$ either **(A)** i is the vertex of smallest index in U, (written as $i = \min(U)$), and $f(i) = \infty$, or **(B)** there exists a vertex $i \in U$ such that $0 \leq f(v_i) < \mathbb{O}_U(i)$.

The vertices which satisfy $f(i) = \infty$ in **(A)** will be called *roots of f* and those that satisfy **(B)** (in U) are said to be *well-behaved* in U, and **(A)** and **(B)** will be used to refer, respectively, to these conditions hereafter. Note that vertex 1 is always a root. The G-multiparking functions with only one root (which is necessarily vertex 1) are in obvious bijection (in fact, the differences are only notational) to the G-parking functions, as defined by Postnikov and Shapiro.

B. Algorithms

In this section, we construct bijections between the set $\mathcal{MP}_G$ of G-multiparking functions and the set $\mathcal{F}_G$ of spanning forests of G. For simplicity, here we assume G

is a simple graph with $V(G) = [n]$. A *sub-forest* F of G is a subgraph of G without cycles. A leaf of F is a vertex $v \in V(F)$ with degree 1 in F. Denote the set of leaves of F by Leaf(F). Let $\prod$ be the set of all ordered pairs (F, W) such that F is a sub-forest of G, and $\emptyset \neq W \subseteq \text{Leaf}(F)$. A *choice function* γ is a function from $\prod$ to $V(G)$ such that $\gamma(F, W) \in W$. Examples of various choice functions will be given in section C.

Fix a choice function γ. Given a G-multiparking function $f \in \mathcal{MP}_G$, we define an algorithm to find a spanning forest $F \in \mathcal{F}_G$. Explicitly, we define quadruples $(\text{val}_i, P_i, Q_i, F_i)$ recursively for $i = 0, 1, \ldots, n$, where $\text{val}_i : V(G) \to \mathbb{Z}$ is the *value function*, P_i is the set of *processed* vertices, Q_i is the set of vertices *to be processed*, and F_i is a subforest of G with $V(F_i) = P_i \cup Q_i$, $Q_i \subseteq \text{Leaf}(F_i)$ or Q_i consists of an isolated vertex of F_i.

Algorithm A.

- **Step 1: initial condition.** Let $\text{val}_0 = f$, P_0 be empty, and $F_0 = Q_0 = \{1\}$.

- **Step 2: choose a new vertex v.** At time $i \geq 1$, let $v = \gamma(F_{i-1}, Q_{i-1})$, where γ is the choice function.

- **Step 3: process vertex v.** For every vertex w adjacent to v and $w \notin P_{i-1}$, set $\text{val}_i(w) = \text{val}_{i-1}(w) - 1$. For any other vertex u, set $\text{val}_i(u) = \text{val}_{i-1}(u)$. Let $N = \{w | \text{val}_i(w) = -1, \text{val}_{i-1}(w) \neq -1\}$. Update P_i, Q_i and F_i by letting $P_i = P_{i-1} \cup \{v\}$, $Q_i = Q_{i-1} \cup N \setminus \{v\}$ if $Q_{i-1} \cup N \setminus \{v\} \neq \emptyset$, otherwise $Q_i = \{u\}$ where u is the vertex of the lowest-index in $[n] - P_i$. Let F_i be a graph on $P_i \cup Q_i$ whose edges are obtained from those of F_{i-1} by joining edges $\{w, v\}$ for each $w \in N$. We say that the vertex v is processed at time i.

Iterate steps 2-3 until $i = n$. We must have $P_n = [n]$ and $Q_n = \emptyset$. Define $\Phi = \Phi_{\gamma,G} : \mathcal{MP}_G \to \mathcal{F}_G$ by letting $\Phi(f) = F_n$.

If an edge $\{v, w\}$ is added to the forest F_i as described in Step 3, we say that w *is found by* v, and v is the *parent* of w, if $v \in P_{i-1}$. (In this paper, the parent of vertex v will be frequently denoted v^p.) By Step 3, a vertex w is in Q_i because either it is found by some v that has been processed, and $\{v, w\}$ is the only edge of F_i that has w as an endpoint, or w is the lowest-index vertex in $[n] - P_i$ and is an isolated vertex of F_i. Also, it is clear that each F_i is a forest, since every edge $\{u, w\}$ in $F_i \setminus F_{i-1}$ has one endpoint in $V(F_i) \setminus V(F_{i-1})$. Hence $\gamma(F_i, Q_i)$ is well-defined and thus we have a well-defined map Φ from $\mathcal{MP}_G$ to $\mathcal{F}_G$. The following proposition describes the role played by the roots of a G-multiparking function f.

Proposition 8. Let f be a G-multiparking function. Each tree component T of $\Phi(f)$ has exactly one vertex v with $f(v) = \infty$. In particular, v is the least vertex of T.

Proof. In the algorithm A the value for a root of f never changes, as $\infty - 1 = \infty$. Each nonroot vertex w of T is found by some other vertex v, and $\{v, w\}$ is an edge of T. As any tree has one more vertex than its number of edges, it has exactly one vertex without a parent. By the definition of Algorithm A, this must be a root of f.

To show that the root is the least vertex in each component, let $r_1 < r_2 < \cdots < r_k$ be the roots of f and suppose $T_1, T_2, \ldots, T_k$ are the trees of $F = \Phi(f)$, where $r_i \in T_i$. Let T_j be the tree of smallest index j such that there is a $v \in T_j$ with $v < r_j$. Then $j > 1$ since the vertex 1 is always a root. Define $U := V(T_j \cup T_{j+1} \cup \ldots \cup T_k)$. U is thus a proper subset of $V(G) = [n]$. By assumption, the vertex of least index in U is not a root. Therefore, U must contain a well-behaved vertex; that is, a vertex v such that $0 \leq f(v) < \mathbb{O}_U(v)$. Note that all the edges counted by $\mathbb{O}_U(v)$ lead to vertices in the trees $T_1, T_2, \ldots, T_{j-1}$. By the structure of algorithm A, all the vertices in the first $j-1$ trees are processed before the parent of v is processed. But this means that by the time A processes all the vertices in the first $j-1$ trees, $\mathrm{val}_i(v) = f(v) - \mathbb{O}_U(v) \leq -1$, so v should be adjacent to some vertex in one of the first $j-1$ trees. This is a contradiction. $\square$

From the above proof we also see that the forest $F = \Phi(f)$ is built tree by tree by the algorithm A. That is, if T_i and T_j are tree components of F with roots r_i, r_j and $r_i < r_j$, then every vertex of tree T_i is processed before any vertex of T_j.

To show that Φ is a bijection, we define a new algorithm to find a G-multiparking function for any given spanning forest, and prove that it gives the inverse map of Φ.

Let G be a graph on $[n]$ with a spanning forest F. Let $T_1, \ldots, T_k$ be the trees of F with respective minimal vertices $r_1 = 1 < r_2 < \cdots < r_k$.

Algorithm B.

- **Step 1. Determine the process order π.** Define a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n)) = (v_1 v_2 \ldots v_n)$ on the vertices of G as follows. First, $v_1 = 1$. Assuming $v_1, v_2, \ldots, v_i$ are determined,
  - **Case (1)** If there is no edge of F connecting vertices in $V_i = \{v_1, v_2, \ldots, v_i\}$ to vertices outside V_i, let v_{i+1} be the vertex of smallest index not already in V_i;
  - **Case (2)** Otherwise, let $W = \{v \notin V_i : v \text{ is adjacent to some vertices in } V_i\}$, and F' be the forest obtained by restricting F to $V_i \cup W$. Let $v_{i+1} = \gamma(F', W)$.

  (Hereafter, when discussing process orders, we will write v_i as $\pi(i)$.)

- **Step 2. Define a G-multiparking function $f = f_F$.** Set $f(r_1) = f(r_2) = \cdots = f(r_k) = \infty$. For any other vertex v, let r_v be the minimal vertex in the tree containing v, and $v, v^p, u_1, \ldots, u_t, r_v$ be the unique path from v to r_v. Set $f(v)$ to be the cardinality of the set $\{v_j | (v, v_j) \in E(G),\ \pi^{-1}(v_j) < \pi^{-1}(v^p)\}$.

To verify that a function $f = f_F$ defined in this way is a G-multiparking function, we need the following lemma.

Lemma 9. Let $f : V(G) \to \mathbb{N} \cup \{\infty\}$ be a function. If $v \in U \subseteq V(G)$ obeys property **(A)** or property **(B)** and W is a subset of U containing v, then v obeys the same property in W.

Proof. If $f(v) = \infty$ and v is the smallest vertex in U, then clearly it will still be the smallest vertex in W. If v is well-behaved in U, then $0 \leq f(v) < \mathbb{O}_U(v)$ and as $W \subseteq U$, we have $\mathbb{O}_U(v) \leq \mathbb{O}_W(v)$. Thus v is well-behaved in W. □

The *burning algorithm* was developed by Dhar in [9] to determine if a function on the vertex set of a graph had a property called *recurrence.* An equivalent description for G-parking functions is given in [5]: We mark vertices of G starting with the root 1. At each iteration of the algorithm, we mark all vertices v that have more marked neighbors than the value of the function at v. The function is a G-parking function if and only if all vertices are marked when this process terminates. Here we extend the burning algorithm to G-multiparking functions, and write it in a linear form.

Proposition 10. A vertex function is a G-multiparking function if and only if there exists an ordering $\pi(1), \pi(2), \ldots, \pi(n)$ of the vertices of a graph G such that for every j, $\pi(j)$ satisfies either condition **(A)** or condition **(B)** in $U_j := \{\pi(j), \ldots, \pi(n)\}$.

Proof. We say that the vertices can be "thrown out" in the order $\pi(1), \pi(2), \ldots, \pi(n)$ if they satisfy the condition described in the proposition. By the definition of G-multiparking function, it is clear that for a G-multiparking function, vertices can be thrown out in some order.

Conversely, suppose that for a vertex function $f : V(G) \to \mathbb{N} \cup \{\infty\}$ the vertices of G can be thrown out in a particular order $\pi(1), \pi(2), \ldots, \pi(n)$. For any subset U of

$V(G)$, let k be the maximal index such that $U \subseteq U_k = \{\pi(k), \ldots, \pi(n)\}$. This implies $\pi(k) \in U$. But $\pi(k)$ satisfies either condition **(A)** or condition **(B)** in U_k. By Lemma 9, $\pi(k)$ satisfies either condition **(A)** or condition **(B)** in U. Since U is arbitrary, f is a G-multiparking function. $\square$

Proposition 11. The Algorithm B, when applied to a spanning forest of G, yields a G-multiparking function $f = f_F$.

Proof. Let π be the permutation defined in Step 1 of Algorithm B. We show that the vertices can be thrown out in the order $\pi(1), \pi(2), \ldots, \pi(n)$. As $\pi(1) = 1$, the vertex $\pi(1)$ clearly can be thrown out. Suppose $\pi(1), \ldots, \pi(k-1)$ can be thrown out, and consider $\pi(k)$.

If $f(\pi(k)) = \infty$, by Case (1) of step 1, $\pi(k)$ is the smallest vertex not in $\{\pi(1), \ldots, \pi(k-1)\}$. Thus it can be thrown out.

If $f(\pi(k)) \neq \infty$, there is an edge of the forest F connecting $\pi(k)$ to a vertex w in $\{\pi(1), \ldots, \pi(k-1)\}$. Suppose $w = \pi(t)$ where $t < k$. By definition of f, there are exactly $f(\pi(k))$ edges connecting $\pi(k)$ to the set $\{\pi(1), \ldots, \pi(t-1)\}$. Hence $f(\pi(k)) < \mathbb{O}_{\{\pi(k),\ldots,\pi(n)\}}(\pi(k))$. Thus $\pi(k)$ can be thrown out as well. By induction the vertices of G can be thrown out in the order $\pi(1), \pi(2), \ldots, \pi(n)$. $\square$

Define $\Psi_{\gamma,G} : \mathcal{F}_G \to \mathcal{MP}_G$ by letting $\Psi_{\gamma,G}(f) = f_F$. Now we show that $\Phi = \Phi_{\gamma,G}$ and $\Psi = \Psi_{\gamma,G}$ are inverses of each other.

Theorem 12. $\Psi(\Phi(f)) = f$ for any $f \in \mathcal{MP}_G$ and $\Phi(\Psi(F)) = F$ for any $F \in \mathcal{F}_G$.

Proof. First, if $f \in \mathcal{MP}_G$ and $F = \Phi(f)$, then by Prop. 8 the roots of f are exactly the minimal vertices in each tree component of F. Those in turn are roots for $\Psi(F)$. In applying algorithm B to F, we note that the order $\pi = v_1 v_2 \ldots v_n$ is exactly the order in which vertices of G will be processed when running algorithm A on f. That is, $P_i = \{v_1, \ldots, v_i\}$, and v_{i+1} is not a root of f, then Q_i is the set of vertices which are adjacent (via edges in F) to those in P_i. By the construction of algorithm A, a vertex w is found by v if and only if there are $f(w)$ many edges connecting w to vertices that are processed before v, or equivalently, to vertices u with $\pi^{-1}(u) < \pi^{-1}(v)$. Since in $\Phi(f)$, $v = w^p$, we have $\Psi(\Phi(f)) = f$.

Conversely, we prove that $\Phi(\Psi(F)) = F$ by showing that $\Phi(\Psi(F))$ and F have the same set of edges. First note that the minimal vertices of the tree components of F are exactly the roots of $f = \Psi(F)$, which then are the minimal elements of trees in $\Phi(f)$. Edges of F are of the form $\{v, v^p\}$, where v is not a minimal vertex in its tree component. We now show that when applying algorithm A to $\Psi(F)$, vertex v is found by v^p. Note that $f(v) = |\{v_j | (v, v_j) \in E(G), \pi^{-1}(v_j) < \pi^{-1}(v^p)\}|$. In the implementation of algorithm A, the *valuation* on v drops by 1 for each adjacent vertex that is processed before v. When it is v^p's turn to be processed, $val_i(v)$ drops from 0 to -1. Thus v^p finds v, and $\{v, v^p\}$ is an edge of $\Phi(\Psi(F))$. $\square$

Since the roots of the G-multiparking function correspond exactly to the minimal vertices in the tree components of the corresponding forest, in the following we will refer to those vertices as *roots of the forest*.

C. Examples of the Bijections

The bijections $\Phi_{\gamma,G}$ and $\Psi_{\gamma,G}$, as defined above via algorithms A and B, allow a good deal of freedom in implementation. In algorithm A, as long as γ is well-defined at every iteration of Step 2, one can obtain val_{i+1}, P_{i+1}, Q_{i+1} and F_{i+1} and proceed. Recall that γ is a function from $\prod$, the set of ordered pairs (F, W), to $V(G)$ such that $\gamma(F, W) \in W$, where F is any sub-forest of G (not necessarily spanning) and W is a non-empty subset of $\text{Leaf}(F)$ or consists of an isolated point of F.

When restricting to G-parking functions, (i.e., G-multiparking functions with only one root), the descriptions of the bijections Φ and Ψ are basically the same as the ones given by Chebikin and Pylyavskyy [5], where the corresponding substructures in G are spanning trees. However our family of bijections, each defined on a choice function γ, is more general than the ones in [5], which rely on a *proper set of tree orders.* A proper set of tree orders is a set $\Pi(G) = \{\pi(T) : T \text{ is a subtree of } G\}$ of linear orders on the vertices of T, such that for any $v \in T$, $v <_{\pi(T)} v^p$, and if T' is a subtree of T containing the least vertex, $\pi(T')$ is a suborder of $\pi(T)$. Our algorithms do not require there to be a linear order on the vertices of each subtree. In fact, for a spanning tree T of a connected graph G, the proper tree order $\pi(T)$, if it exists, must be the same as the one defined in Step 1 of algorithm B. But in general, for two spanning trees T and T' with a common subtree t, the restrictions of $\pi(T)$ and $\pi(T')$ to vertices of t may not agree. Hence in general the choice function cannot be described in terms of proper sets of tree orders. In addition, our description of the map Φ, in terms of a dynamic process, provides a much clearer way to understand the bijection, and leads to a natural classification of the edges of G which plays an important role in connection with the Tutte polynomial (c.f.§4).

Different choice functions γ will induce different bijections between $\mathcal{MP}_G$ and $\mathcal{F}_G$. In this section we give several examples of choice functions that have combinatorial significance. In Example 1 we explain how to translate a proper set of tree orders into a choice function. Hence the family of bijections defined in [5] can be viewed as a subfamily of our bijections restricted to G-parking functions. The next three examples have appeared in [5]. We list them here for their combinatorial significance. Example 5 is the combination of breadth-first search with the Q-sets equipped with certain data structures. It is the one used to establish connections with Tutte polynomial in §4. The last example illustrates a case where γ cannot be expressed as a proper set of tree orders. We illustrate the corresponding map $\Phi_{\gamma,G}$ for examples 2–6 on the graph G in Figure 5. A G-multiparking function f is indicated by "$i/f(i)$" on vertices, where i is the vertex label.

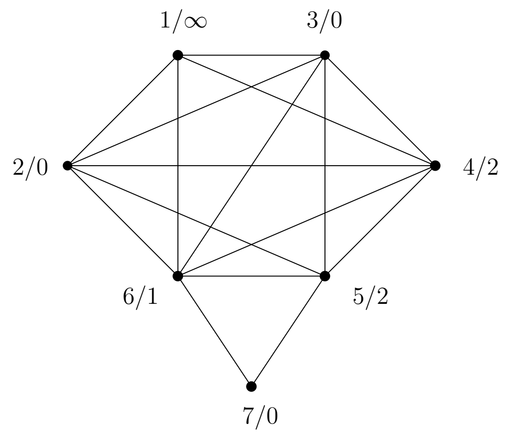


Fig. 5. A graph and a multiparking function.

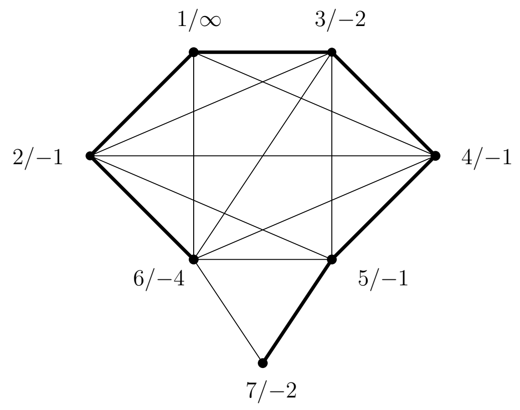
In each example, we will show the resulting spanning forest by darkened edges in G. Again each vertex will be labeled by a pair i/j, where i is the vertex labels, and $j = \mathrm{val}_n(i)$, where $n = 7$. Beneath that, a table will record the sets Q_t and P_t for each time t. In each Q_t, the vertex listed first is the next to be processed.

Example 1. γ with a proper set of tree orders.

We define the choice function that corresponds to a proper set of tree orders. Here we should generalize to the proper set of forest orders, i.e., a set of orders $\pi(F)$, defined on the set of vertices for each subforest F of G, such that for any $v \in F$, $v <_{\pi(F)} v^p$, and if F' is a subforest of F with the same minimal vertex in each tree component, $\pi(F')$ is a suborder of $\pi(F)$. In this case, define $\gamma(F, W) = v$ where v is the minimal element in W under the order $\pi(F)$. Examples 2–4 are special cases of this kind.

Example 2. γ with a given vertex ranking.

Given a vertex ranking $\sigma \in S_n$ define $\gamma_\sigma(F, W) := v$, where v is the vertex in W with minimal ranking. In particular, if σ is the identity permutation, then the vertex processing order is the *vertex-adding order* of [5]. In this case, in Step 2 of algorithm A, we choose v to be the least vertex in Q_{i-1} and process it at time i. The output of algorithm A is given below in Figure 6.



t	0	1	2	3	4	5	6	7
Q_t	{1}	{2,3}	{3,6}	{4,6}	{5,6}	{6,7}	{7}	$\emptyset$
P_t	$\emptyset$	{1}	{1,2}	{1,2,3}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5,6}	{1,2,3,4,5,6,7}

Fig. 6. The spanning forest determined by the given vertex ranking.

Example 3. γ with *depth-first search order.* The depth-first search order is the order in which vertices of a forest are visited when performing the depth-first search, which is also known as the preorder traversal. Given a forest F with tree components $T_1, T_2, \ldots, T_k$, where $1 = r_1 < r_2 < \cdots < r_k$ are the corresponding roots, the order $<_{df}$ is defined as follows. (1) For any $v \in T_i$, $w \in T_j$ and $i < j$, $v <_{df} w$. (2) For any $v \neq r_i$, $v^p <_{df} v$. (3) If $v^p = w^p$ and $v < w$, $v <_{df} w$. (4) For any v, let $F[v]$ be the subtree of F rooted at v. If $v \in F[v']$, $w \in F[w']$ and $v' <_{df} w'$, then $v <_{df} w$. For example, the depth-first search order on the tree in Figure 7 is $1 <_{df} 2 <_{df} 3 <_{df} 6 <_{df} 4 <_{df} 5$.

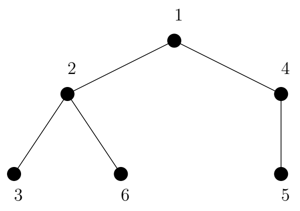
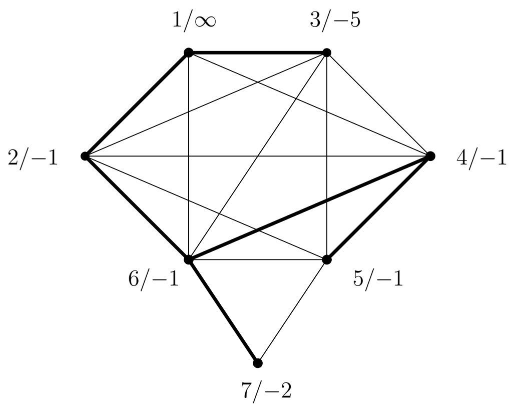


Fig. 7. A tree with 6 vertices.

The choice function γ_{df} with depth-first search order is then defined as $\gamma_{df}(F, W) = v$ where v is the minimal element of W under the depth-first search order $<_{df}$ of F. The output of algorithm A is given below in Figure 8.



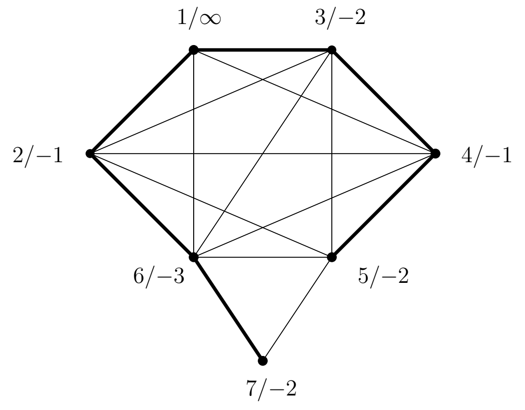
t	0	1	2	3	4	5	6	7
Q_t	{1}	{2,3}	{6,3}	{4,3,7}	{5,3,7}	{7,3}	{3}	∅
P_t	∅	{1}	{1,2}	{1,2,6}	{1,2,4,6}	{1,2,4,5,6}	{1,2,4,5,6,7}	{1,2,3,4,5,6,7}

Fig. 8. The spanning forest determined by the depth-first search order.

Example 4. γ with *breadth-first search order.* Breadth-first search is another commonly used tree traversal in computer science. Given a forest F, whose tree components are T_i with roots r_i, $(1 \le i \le k)$, and $1 = r_1 < r_2 < \cdots < r_k$, the order $<_{bf}$ is defined as follows. (1) For any $v \in T_i$, $w \in T_j$ and $i < j$, $v <_{bf} w$. (2) Within tree T_i, for each $v \in T_i$, let *height* $h_{T_i}(v)$ of v be the number of edges in the unique path from v to the root r_i. We set $v <_{bf} w$ if $h_{T_i}(v) < h_{T_i}(w)$, or else if $h_{T_i}(v) = h_{T_i}(w)$ and $v < w$. For example, the the breadth-first search order for the tree in Figure 7 is

$1 <_{bf} 2 <_{bf} 4 <_{bf} 3 <_{bf} 5 <_{bf} 6$.

The choice function γ_{bf} with breadth-first search order is defined as $\gamma_{bf}(F, W) = v$ where v is the minimal element of W under the breadth-first search order $<_{bf}$ of F. The output of algorithm A is given below in Figure 9.



t	0	1	2	3	4	5	6	7
Q_t	{1}	{2,3}	{3,6}	{4,6}	{6,5}	{5,7}	{7}	$\emptyset$
P_t	$\emptyset$	{1}	{1,2}	{1,2,3}	{1,2,3,4}	{1,2,3,4,6}	{1,2,3,4,5,6}	{1,2,3,4,5,6,7}

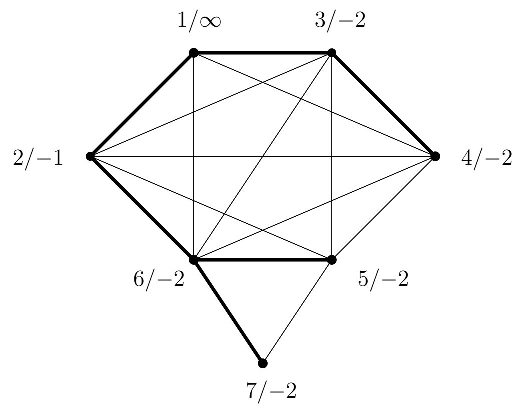
Fig. 9. The spanning forest determined by the breadth-first search order.

Example 5. Breadth-first search with a data structure on Q_i. In this case, new vertices enter the set Q_i in a certain order, and some intrinsic data structure on Q_i decides which vertex of Q_i is to be processed in the next step. A typical example is that of breadth-first search with a queue, in which case each Q_i is an ordered set, (i.e., the stage of a queue at time i). New vertices enter Q_i in numerical order, and γ chooses the vertex that entered the queue earliest.

This example can also be defined by a modified breadth-first search order, which we call *breadth-first order with a queue*, and denote by $<_{bf,q}$. Given a forest F, whose tree components are T_i with root r_i, $(1 \le i \le k)$, and $1 = r_1 < r_2 < \cdots < r_k$, the order $<_{bf,q}$ is defined as follows. (1) For any $v \in T_i$, $w \in T_j$ and $i < j$, $v <_{bf,q} w$. (2) Within tree T_i, the root r_i is minimal under $<_{bf,q}$. (3) $v <_{bf,q} w$ if $v^p <_{bf,q} w^p$. (4)

If $v^p = w^p$ and $v < w$, $v <_{bf,q} w$. For example, the breadth-first search order with a queue for the tree in Figure 7 is $1 <_{bf,q} 2 <_{bf,q} 4 <_{bf,q} 3 <_{bf,q} 6 <_{bf,q} 5$.

The choice function γ associated with this order is denoted by $\gamma_{bf,q}$, and is used in §4. The following is the output of algorithm A with $\gamma_{bf,q}$ on the graph in Figure 5.



t	0	1	2	3	4	5	6	7
Q_t	(1)	(2,3)	(3,6)	(6,4)	(4,5,7)	(5,7)	(7)	$\emptyset$
P_t	$\emptyset$	{1}	{1,2}	{1,2,3}	{1,2,3,6}	{1,2,3,4,6}	{1,2,3,4,5,6}	{1,2,3,4,5,6,7}

Fig. 10. The spanning forest determined by the breadth-first search with queue order.

Another typical structure is to let Q_i be the stage of a stack at time i, that is, it pops out the vertex that last entered. We can also combine the other vertex orders with a queue or stack for the Q-sets.

Example 6. A choice function γ that cannot be defined by a proper set of tree orders.

Let

$$\gamma(F, W) = \begin{cases} x & \text{if } W = \{x\}, \\ \text{the second minimal vertex of } W, & \text{if } |W| \geq 2. \end{cases}$$

Then in Figure 11, the order on the left tree is 156342, and the one on the right tree is 153462, which do not agree on the subtree consisting of vertices 1356. Hence it can not be defined via a proper set of tree orders.

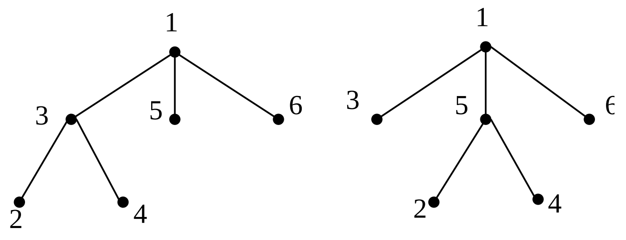


Fig. 11. A tree that cannot be defined by a proper set of tree orders.

D. External Activity and the Tutte Polynomial

1. F-redundant Edges

A forest F on $[n]$ may appear as a subgraph of different graphs, and a vertex function f may be a G-multiparking function for different graphs. In this section we characterize the set of graphs which share the same pair (F, f). Again let G be a simple graph on $[n]$, and fix a choice function γ. For a spanning forest F of G, let $f = \Psi_{\gamma,G}(F)$. We say an edge e of $G - F$ is *F-redundant* if $\Psi_{\gamma,G-\{e\}}(F) = f$. Note that we only need to use the value of γ on (F', W) where F' is a sub-forest of F. Hence $\Psi_{\gamma,G-\{e\}}(F)$ is well-defined.

Let π be the order defined in Step 1 of Algorithm B. Note that π only depends on F, not the underlying graph G. Recall that v^p denotes the parent vertex of vertex v in some spanning forest. We have the following proposition.

Proposition 13. An edge $e = \{v, w\}$ of G is F-redundant if and only if e is one of the following types:

1. Both v and w are roots of F.
2. v is a root and w is a non-root of F, and $\pi^{-1}(w) < \pi^{-1}(v)$.
3. v and w are non-roots and $\pi^{-1}(v^p) < \pi^{-1}(w) < \pi^{-1}(v)$. In this case v and w

must lie in the same tree of F.

Proof. We first show that each edge of the above three types are F-redundant. Since for any root r of the forest F, $f(r) = \infty$, the edges of the first two types play no role in defining the function f. And clearly those edges are not in F. Hence they are F-redundant.

For edge (v, w) of type 3, clearly it cannot be an edge of F. Since $f(v) = \#\{v_j | (v, v_j) \in E(G), \pi^{-1}(v_j) < \pi^{-1}(v^p)\}$, and $\pi^{-1}(w) > \pi^{-1}(v^p)$, removing the edge $\{v, w\}$ would not change the value of $f(v)$. This edges has no contribution in defining $f(u)$ for any other vertex u. Hence it is F-redundant.

For the converse, suppose that $e = \{v, w\}$ is not one of the three type. Assume w is processed before v in π. Then v is not a root, and w appears before v^p. Then removing the edge e will change the value of $f(v)$. Hence it is not F-redundant. □

Let $R_1(G; F)$, $R_2(G; F)$, and $R_3(G; F)$ denote the sets of F-redundant edges of types 1, 2, and 3, respectively. Among them, $R_3(G; F)$ is the most interesting one, as $R_1(G; F)$ and $R_2(G; F)$ are a consequence of the requirement that $f(r) = \infty$ for any root r. Let $R(G; F)$ be the union of these three sets. Clearly the F-redundant edges are mutually independent, and can be removed one by one without changing the corresponding G-multiparking function. Hence

Theorem 14. Let H be a subgraph of G with $V(H) = V(G)$. Then $\Psi_{\gamma,G}(F) = \Psi_{\gamma,H}(F)$ if and only if $G - R(G; F) \subseteq H \subseteq G$.

2. A Classification of the Edges of G

The notion of F-redundancy allows us to classify the edges of a graph in terms of the algorithm A. Roughly speaking, the edges of any graph can be thought of as either lowering $val(v)$ for some v to 0, being in the forest, or being F-redundant. Explicitly, we have

Proposition 15. Let f be a G-multiparking function and let $F = \Phi(f)$. Then

$$|E(G)| = \left(\sum_{v:f(v)\neq\infty} f(v) \right) + |E(F)| + |R(G;F)|.$$

Proof. For each non-root vertex v, the number of different values that $val_i(v)$ takes on during the execution of algorithm A is $f(v)+1+n_v$, where $n_v = -val_n(v)$. At the beginning, $val_0(v) = f(v)$. The value $val_i(v)$ then is lowered by one whenever there is a vertex w which is adjacent to v and processed before v^p. When v^p is being processed, $val_i(v) = -1$, and the edge $\{v^p, v\}$ contributes to the forest F. Afterward, the value of $val_i(v)$ decreases by 1 for each F-redundant edge $\{u, v\}$ with $\pi^{-1}(u) < \pi^{-1}(v)$. Summing over all non-root vertices gives

$$\sum_{v:f(v)\neq\infty} deg_{<_\pi}(v) \quad = \quad \sum_{v:f(v)\neq\infty} f(v) + |E(F)| + \sum_{v:f(v)\neq\infty} n_v,$$

where $deg_{<_\pi}(v) = |\{\{w,v\} \in E(G) | \pi^{-1}(w) < \pi^{-1}(v)\}|$.

The edges that lower $val(v)$ below -1 are exactly the F-redundant edges of type (3) in Prop. 13, hence $\sum_{v:f(v)\neq\infty} n_v = |R_3(G;F)|$. On the other hand, $\sum_{v:f(v)\neq\infty} deg_{<_\pi}(v)$ is exactly $|E(G)| - |R_1(G;F)| - |R_2(G;F)|$. The claim follows from the fact that the sets $R_1(G;F), R_2(G;F)$, and $R_3(G;F)$ are mutually exclusive. $\square$

One notes that for roots of f and $F = \Phi(f)$, $|R_1(G;F)| + |R_2(G;F)|$ is exactly $\sum_{root\ v} deg_{<_\pi}(v)$, where π is the processing order in algorithm A. But it is not necessary to run the full algorithm A to compute $|R_1(G;F)| + |R_2(G;F)|$. Instead, we can apply the burning algorithm in a greedy way to find an ordering $\pi' = v'_1 v'_2 \cdots v'_n$ on $V(G)$: Let $v'_1 = 1$. After determining $v'_1, \ldots, v'_{i-1}$, if $V_i = V(G) - \{v_1, \ldots, v'_{i-1}\}$ has a well-behaved vertex, let v'_i be one of them; otherwise, let v'_i be the minimal vertex of

V_i, (which has to be a root.)

π' may not be the same as π, but they have the following properties:

1. Let $r_1 < r_2 < \cdots < r_k$ be the roots of f. Then $r_1, r_2, \ldots, r_k$ appear in the same positions in both π and π'.

2. The set of vertices lying between r_i and r_{i+1} are the same in π and π'. In fact, they are the vertices of the tree T_i with root r_i in $F = \Phi(f)$.

It follows that for any root vertex v, $deg_{<_\pi}(v) = deg_{<_{\pi'}}(v)$. The value of $deg_{<_\pi}(v)$ (v root) can be characterized by a global description: Let $\mathcal{U}_v$ be the collection of subsets U of $V(G)$ such that $v = \min(U)$, and U does not have a well-behaved vertex. $\mathcal{U}_v$ is nonempty for a root v since $U = \{v\}$ is such a set. Then

$$deg_{<_\pi}(v) = \min_{U \in \mathcal{U}_v} \mathbb{O}_U(v).$$

We call $deg_{<_\pi}(v)$ the *record* of the root v, and denote it by $\mathrm{rec}(v)$. Then

$$|R_1(G;F)| + |R_2(G;F)| = \sum_{root\ v} deg_{<_{\pi'}}(v) = \sum_{root\ v} \mathrm{rec}(v)$$

is the *total root records*. Let $\mathrm{Rec}(f) = |R_1(G;F)| + |R_2(G;F)|$. It is the number of F-redundant edges adjacent to a root. By the above greedy burning algorithm, the total root records $\mathrm{Rec}(f)$ can be computed in linear time.

3. A New Expression for the Tutte Polynomial

In this subsection we relate G-multiparking functions to the Tutte polynomial $T_G(x,y)$ of G. We follow the presentation of [13] for the definition of Tutte polynomial and its basic properties. Although the theory works for general graphs with multiedges, we assume G is a simple connected graph to simplify the discussion. There is no loss of generality by assuming connectedness, since for a disconnected graph, $T_G(x, y)$ is just the product of the Tutte polynomials of the components of G. We restrict ourselves to connected graphs to avoid any possible confusion when we consider their spanning forests. The modification when G has multiple edges is explained at the end of (CITE).

Suppose we are given G and a total ordering of its edges. Consider a spanning tree T of G. An edge $e \in G - T$ is *externally active* if it is the largest edge in the unique cycle contained in $T \cup e$. We let

$$\mathcal{EA}(T) = \text{set of externally active edges in } T$$

and $ea(T) = |\mathcal{EA}(T)|$. An edge $e \in T$ is *internally active* if it is the largest edge in the unique cocycle contained in $(G - T) \cup e$. We let

$$\mathcal{IA}(T) = \text{set of internally active edges in } T$$

and $ia(T) = |\mathcal{IA}(T)|$. Tutte [29] then defined his polynomial as

$$T_G(x, y) = \sum_{T \subset G} x^{ia(T)} y^{ea(T)}, \tag{2.1}$$

where the sum is over all spanning trees T of G. Tutte showed that T_G is well-defined, i.e., independent of the total ordering of the edges of G. Henceforth, we will not assume that the edges of G are ordered.

Let H be a (spanning) subgraph of G. Denote by $c(H)$ the number of components of H. Define two invariants associated with H as

$$\sigma(H) = c(H) - 1, \qquad \sigma^*(H) = |E(H)| - |V(G)| + c(H). \tag{2.2}$$

The following identity is well-known, for example, see [1].

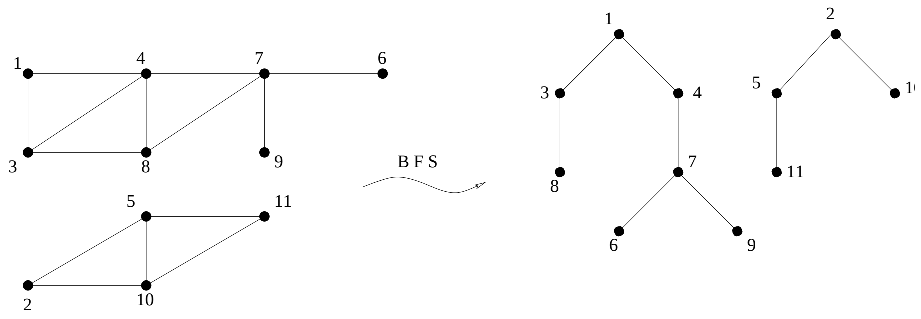
Theorem 16.

$$T_G(1+x, 1+y) = \sum_{H \subseteq G} x^{\sigma(H)} y^{\sigma^*(H)}, \tag{2.3}$$

where the sum is over all spanning subgraphs H of G.

Recall that the *breadth-first search* (BFS) is an algorithm that gives a spanning forest in the graph H. Assume $V(G) = [n]$. We will use our favorite description to express the BFS as a queue Q that starts at the least vertex 1. This description was first introduced in [26] to develop an exact formula for the number of labeled connected graphs on $[n]$ with a fixed number of edges, and was used by the second author in [31] to reveal the connection between the classical parking functions (resp. k-parking functions) and the complete graph (resp. multicolored graphs).

Given a subgraph H of G with $V(H) = V(G) = [n]$, we construct a queue Q. At time 0, Q contains only the vertex 1. At each stage we take the vertex x at the head of the queue, remove x from the queue, and add all unvisited neighbors $u_1, \dots, u_{t_x}$ of x to the queue, in numerical order. We will call this operation "processing x". If the queue becomes empty, add the least unvisited vertex to Q. The output F is the forest whose edge set consists of all edges of the form $\{x, u_i\}$ for $i = 1, \dots t_x$. We will denote this output as $F = BFS(H)$. Figure 12 shows the spanning forest found by BFS for a graph G.



t	0	1	2	3	4	5	6	7	8	9	10	11
Q	(1)	(3,4)	(4,8)	(8,7)	(7)	(6,9)	(9)	(2)	(5,10)	(10,11)	(11)	$\emptyset$

Fig. 12. Spanning forest found by BFS.

For a spanning forest F of G, let us say that an edge $e \in G - F$ is *BFS-externally active* if $BFS(F \cup e) = F$. A crucial observation is made by Spencer [26]: An edge $\{v, w\}$ can be added to F without changing the spanning forest under the BFS if and only if the two vertices v and w have been present in the queue at the same time. In our example of Figure 12, edges $\{3,4\}, \{4,8\}, \{7,8\}, \{6,9\}, \{5,10\}$ and $\{10,11\}$ could be added back to F. We write $\mathcal{E}(F)$ for the set of BFS-externally active edges.

Proposition 17 (Spencer). If H is any subgraph and F is any spanning forest of G then $BFS(H) = F$ if and only if $F \subseteq H \subseteq F \cup \mathcal{E}(F)$.

Now consider the Tutte polynomial. Note that if $BFS(H) = F$, then $c(H) = c(F)$. So $\sigma(H) = c(F) - 1$ and $\sigma^*(H) = |E(H)| - |E(F)| = |\mathcal{E}(F) \cap H|$. Hence if we fix a forest F and sum over the corresponding interval $[F, F \cup \mathcal{E}(F)]$, we have

$$\sum_{H:BFS(H)=F} x^{\sigma(H)} y^{\sigma^*(H)} = x^{c(F)-1} \sum_{A \subseteq \mathcal{E}(F)} y^{|A|} = x^{c(F)-1}(1+y)^{|\mathcal{E}(F)|}.$$

Summing over all forests F, we get

$$T_G(1+x, 1+y) = \sum_{H \subseteq G} x^{\sigma(H)} y^{\sigma^*(H)} = \sum_{F \subseteq G} x^{c(F)-1}(1+y)^{|\mathcal{E}(F)|}.$$

Or, equivalently,

$$T_G(1+x,y) = \sum_{F\subseteq G} x^{c(F)-1} y^{|\mathcal{E}(F)|}. \tag{2.4}$$

To evaluate $\mathcal{E}(F)$, note that when applying BFS to a graph H, the queue Q only depends on the spanning forest $F = BFS(H)$. Given a forest F, the processing order in Q is a total order $<_Q = <_Q (F)$ on the vertices of F satisfying the following condition: Let $T_1, T_2, \ldots, T_k$ be the tree components of F with minimal elements $r_1 = 1 < r_2 < \cdots < r_k$. Then (1) If v is a vertex in tree T_i, w is a vertex in tree T_j and $i < j$, then $v <_Q w$. (2) Among vertices of each tree T_i, r_i is minimal in the order $<_Q$. (3) For two non-root vertices v, w in the same tree, $v <_Q w$ if $v^p <_Q w^p$. In the case $v^p = w^p$, $v <_Q w$ whenever $v < w$.

Comparing with the examples in section (CITE), we note that $<_Q$ is exactly the order $<_{bf,q}$ described in Example 5 of (CITE), as *breadth-first order with a queue.* Fix the choice function $\gamma = \gamma_{bf,q}$, the one associated to $<_{bf,q}$ and consider the maps $\Phi_{\gamma,G}$ and $\Psi_{\gamma,G}$. Given F, the condition that two vertices v, w have been present at the queue Q at the same time when applying BFS to F is equivalent to $v^p <_{bf,q} w <_{bf,q} v$ or $w^p <_{bf,q} v <_{bf,q} w$. That is, an edge is BFS-externally active if and only if it is an F-redundant edge of type 3, as defined in §4.1. It follows that $\mathcal{E}(F) = R_3(G;F)$.

Therefore by Prop. 15,

$$|\mathcal{E}(F)| = |R_3(G;F)| = |E(G)| - |E(F)| - \left(\sum_{v:f(v)=-1} f(v)\right) - \mathrm{Rec}(f),$$

where $f = \Psi_{\gamma,G}(F)$ is the corresponding G-multiparking function. Note that $|E(F)| = n - c(F)$, and $c(F) = r(f)$, where $r(f)$ is the number of roots of f. Therefore

Theorem 18.

$$T_G(1+x,y) = y^{|E(G)|-n} \sum_f x^{r(f)-1} y^{r(f)-\mathrm{Rec}(f)-\left(\sum_{v:f(v)\neq\infty} f(v)\right)},$$

where the sum is over all G-multiparking functions.

For a G-multiparking function f, where G is a graph on n vertices, we call the statistics $|E(G)| - n + r(f) - \mathrm{Rec}(f) - \sum_{v:f(v)\neq\infty} f(v)$ the *reversed sum* of f, denote by $rsum(f)$. The name comes from the corresponding notation for classical parking functions, see, for example, [19]. Theorem 18 expresses Tutte polynomial in terms of generating functions of $r(f)$ and $rsum(f)$. In [13] Gessel and Sagan gave a similar expression, in terms of $\mathcal{E}_{DFS}(F)$, the set of *greatest-neighbor externally active* edges of F, which is defined by applying the greatest-neighbor depth-first search on subgraphs of G. Combining the result of [13] (Formula 5), we have

$$xT_G(1+x,y) = \sum_{F\subseteq G} x^{c(F)} y^{|\mathcal{E}_{DFS}(F)|} = \sum_{F\subseteq G} x^{c(F)} y^{|\mathcal{E}(F)|} = \sum_{f\in\mathcal{MP}_G} x^{r(f)} y^{rsum(f)}. \quad (2.5)$$

That is, the three pairs of statistics, $(c(F), |\mathcal{E}_{DFS}(F)|)$ and $(c(F), |\mathcal{E}(F)|)$ for spanning forests, and $(r(f), rsum(f))$ for G-multiparking functions, are equally distributed.

Remark. Alternatively, one can prove Theorem 18 by conducting *neighbors-first search* (NFS), a tree traversal defined in [13, §6], and using $\gamma = \gamma_{df}$, the choice function associated with the depth-first search order, (c.f. Example 3, §3). Here the NFS is another algorithm that builds a spanning forest F given an input graph H. The following description is taken from [13].

NFS1 Let $F = \emptyset$.

NFS2 Let v be the least unmarked vertex in V and mark v.

NFS3 Search v by marking all neighbors of v that have not been marked and adding to F all edges from v to these vertices.

NFS4 Recursively search all the vertices marked in NFS3 in increasing order, stopping when every vertex that has been marked has also been searched.

NFS5 If there are unmarked vertices, then return to NFS2. Otherwise, stop.

The NFS searches vertices of H in a depth-first manner but marks children in a locally breadth-first manner. Figure 13 shows the result of NFS, when applies to the graph on the left of Figure 12.

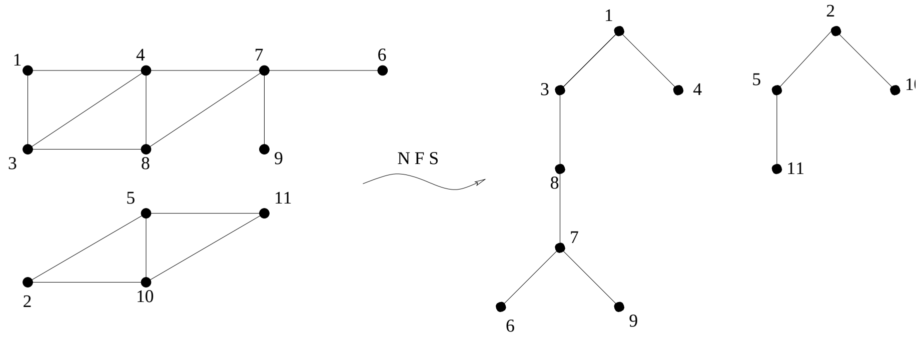


Fig. 13. Spanning forest found by NFS.

Similarly, one defines $\mathcal{E}_{NFS}(F)$, the set of edges *externally active with respect to NFS*, to be those edges $e \in G - F$ such that $NFS(F \cup e) = F$. Then Prop. 17 and Eq. (2.4) hold again when we replace BFS with NFS, and $\mathcal{E}(F)$ with $\mathcal{E}_{NFS}(F)$.

Now let $\gamma = \gamma_{df}$ and use the bijections $\phi_{\gamma_{df},G}$ and $\Psi_{\gamma_{df},G}$, one notices again that an edge is externally active with respect to NFS if and only if it is F-redundant of type 3. And hence we get another proof of Theorem 18.

An interesting specialization of Theorem 18 is to consider $T_G(1, y)$, the restriction to spanning trees of G and G-parking functions. For a G-parking function f, or equivalently a G-multiparking function with exactly one root (which is vertex 1), $r(f) = 1$ and $\mathrm{Rec}(f) = 0$. Hence $rsum(f) = |E(G)| - n + 1 - \sum_{v \neq \infty} f(v)$. Thus we obtain

$$T_G(1, y) = \sum_{f \,:\, G\text{-parking functions}} y^{rsum(f)}.$$

An equivalent form of this result, in the language of sand-pile models, was first proved by López [21] using a recursive characterization of Tutte polynomial. A bijective proof was given by Cori and Le Borgne in [6] by constructing a one-to-one correspondence between trees with external activity i (in Tutte's sense) to recurrent configurations of level i, which is equivalent to G-parking functions with reversed sum i. Our treatment here provides a new bijective proof.

In [13] it is shown that, restricted to simple graphs, the greatest-neighbor externally active edges of F are in one-to-one correspondence with certain inversions of F. For a simple graph G, view each tree T of F as rooted at its smallest vertex. An edge $\{u, v\}$ is greatest-neighbor externally active if and only if v is a descendant of u, and $w > v$ where w is the child of u on the unique $u - v$ path in F, (that is, $u = w^p$). Call such a pair $\{w, v\}$ a G-inversion. And denoted by $Ginv(F)$ the number of G-inversions of the forest F. Then we have the following corollary.

Corollary 19. Let $\mathcal{F}_k(G)$ be the set of spanning forests of G with exactly k tree components. And $\mathcal{MP}_k(G)$ be the set of G-multiparking functions with k roots. Then

$$\sum_{F \in \mathcal{F}_k(G)} y^{Ginv(F)} = \sum_{f \in \mathcal{MP}_k(G)} y^{rsum(f)}.$$

In particular, when G is the complete graph K_{n+1} and $k = 1$, we have the well-known result on the equal-distribution of inversions over labeled trees, and the reversed sum over all classical parking functions of length n, (for example, see [16, 28])

$$\sum_{T \text{ on } [n+1]} y^{inv(T)} = \sum_{\alpha \in P_n} y^{\binom{n}{2} - \sum_{i=1}^{n} \alpha_i},$$

where P_n is the set of all (classical) parking functions of length n.

CHAPTER III

MULTIPARKING FUNCTIONS WITHOUT MINIMALITY

A. Definitions

In this section, we will use a slightly different definition of a G-multiparking functionin order to establish a connection to a generalized *chip firing game* and, later, a generalization of a *descending traversal*.

Throughout this chapter, we let a *G-multiparking function* be a function $f : V(G) \to \mathbb{N} \cup \{\infty\}$ such that for any $U \subseteq V(G)$ there exists $i \in U$ with either **(A)** $f(i) = \infty$, or **(B)** $0 \leq f(i) < \mathbb{O}_U(i)$. As before, we will refer to those vertices i with $f(i) = \infty$ as *roots* and those with $0 \leq f(i) < \mathbb{O}_U(i)$ as being *well-behaved in U*. Let $\mathcal{MP} = \mathcal{MP}_{R,G}$ denote the set of G-multiparking functions with root set R. The important difference between this definition and the one in Chapter II is that the minimal vertex in each component of G is required to be a root; here there is no such restriction. Note, however, that R cannot be empty; $V(G)$ cannot have a well-behaved vertex, so it must have a root. Proposition 10 still holds, however, and thus we retain much of the theory and intuition about G-multiparking functions that we had in the previous section.

B. Dirichlet Configurations

Now we introduce a generalization of a structure that appears in the literature in a variety of contexts and, with minor variations, is known as a critical configuration, a sandpile model, and a chip-firing game. Let R be a set of vertices containing at least one vertex from each component of G. A *configuration* μ on G (with *root set* R) is an integer-valued function on the vertex set for which $\mu(i) = -\infty$ if $i \in R$ and $0 \leq \mu(i) < \infty$ otherwise. A vertex i is said to be *ready* (in μ) if $\mu(i) \geq \deg(i)$. μ is *stable* if $0 \leq \mu(i) < \deg(i)$ for every $i \notin R$. An *avalanche* is a finite sequence $\alpha = (\mu_1, \mu_2, \ldots \mu_t)$ of configurations on G, where for each $1 \leq s < t$ there exists a vertex $i_s \in V(G)$ which is ready in μ_s and

$$\mu_{s+1}(i) = \begin{cases} \mu_s(i) - \deg(i) & \text{if } i = i_s \\ \mu_s(i) + e(i, i_s) & \text{if } i \neq i_s \end{cases}$$

where $e(i, i_s)$ is the number of edges between i and i_s. If we think of μ_s as keeping track of how many "chips" are stored at each vertex on the graph, then we transform μ_s into μ_{s+1} by sending a chip down each edge adjacent to i_s. This process is often called *firing* a vertex (hence the "chip-firing" terminology), and so one usually thinks of an avalanche as a sequence of vertex firings. Note that only vertices that are ready can be fired, and that the same vertex may be fired several times in succession if it has a large enough number of chips. We say that α *begins* at μ_1, *ends* at μ_t, and *connects* these two configurations. We use the convention that if, in any avalanche, μ_1 is stable then every vertex in R is fired in some arbitrary but fixed order and that this is the only situation in which roots are fired. Note that if a chip is sent to a root it disappears from the system; it follows from the connectedness of each component of G that, given any configuration, there is an avalanche leading to a stable configuration. μ is *recurrent* if there is an avalanche that begins and ends at μ. μ is *Dirichlet* if it is both stable and recurrent. Let $\mathcal{DC} = \mathcal{DC}_{R,G}$ denote the set of Dirichlet configurations on G with root set R.

Dirichlet configurations are usually called *critical configurations* when G is connected, and this case has been studied extensively (see, for example, [2]). Aspects of

Dirichlet configurations were first examined in [4], such as bounds on the number of vertex firings necessary to reach a stable configuration.

Example 5. The following example illustrates a critical configuration for Γ. Every vertex is labelled "v_i/n", where v_i is the vertex label and n is the number of chips at that vertex at that configuration. The vertex about to be fired in each configuration is circled.

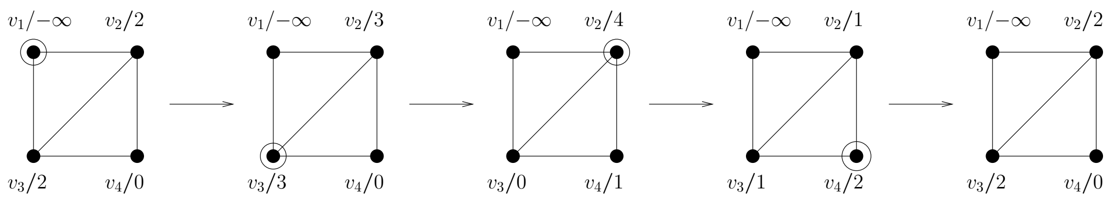


Fig. 14. A sequence of firings proving recurrency.

Note that in the second configuration in the above avalanche, we could have fired either v_2 or v_3. If we had fired v_2 instead of v_3, we would still have ended the avalanche on the configuration we started with.

It is not difficult to see that Dirichlet configurations exist on every graph (for instance, the configuration with $R = V(G)$) and that for every configuration there is an avalanche ending on a stable configuration. This is essentially because every component contains a root and therefore the total number of chips on the graph is nonincreasing after the root firings (if the first configuration is stable) in an avalanche. See Lemma 1 of [4] for a detailed proof.

The following is a characterization of recurrent configurations. Let χ be the configuration

$$\chi(i) = \begin{cases} 0 & \text{if } i \in R \\ \sum_{r \in R} e(i,r) & \text{if } i \notin R \end{cases}$$

Proposition 20. The configuration μ is Dirichlet $\Longleftrightarrow$ it is stable and there is an avalanche connecting $(\mu + \chi)$ to μ, where $(\mu + \chi)(v) = \mu(v) + \chi(v)$.

Proof. ($\Longleftarrow$) The trivial avalanche (consisting of firing all the roots only) connects μ to $(\mu + \chi)$, and thus concatenating this avalanche with the avalanche connecting $(\mu + \chi)$ to μ shows that μ is recurrent. Since it is stable, μ is Dirichlet.

($\Longrightarrow$) Given a Dirichlet μ, it is stable and recurrent. Thus there is an avalanche $(\mu, \omega_1, \omega_2, \ldots, \omega_l, \mu)$. But since μ is stable, the roots are the only vertices that can be fired first. Thus, $\omega_k = (\mu + \chi)$ where k is the number of roots. Thus, $(\omega_k, \omega_{k+1}, \ldots, \omega_l, \mu)$ is the necessary avalanche. $\square$

Cori and Rossin [8] have a similar proof for the case when the graph is connected. The set of critical configurations of G is closely related to the set of G-parking functions; the most famous connection is that both sets are in bijection to the spanning trees of G. Here, however, we provide a bijection between G-multiparking functions and Dirichlet configurations on G that does not go through the set of spanning trees. To simplify the presentation, we will assume G has no multiple edges.

C. A Bijection Between Dirichlet Configurations and G-Multiparking Functions

Theorem 21. Fix a root set R and let $\mathcal{MP} = \mathcal{MP}_{\mathcal{R},\mathcal{G}}$ and $\mathcal{DC} = \mathcal{DC}_{\mathcal{R},\mathcal{G}}$. Define $\Omega : \mathcal{MP} \to \mathcal{DC}$ by $\Omega(f) = \Omega_f$ where

$$\Omega_f(i) = \begin{cases} -\infty & \text{if } i \in R. \\ \deg(i) - 1 - f(i) & \text{if } i \notin R. \end{cases}$$

Then Ω is a bijection, whose inverse $\Omega^{-1} : \mathcal{DC} \to \mathcal{MP}$, is given by $\Omega^{-1}(\mu) = \Omega_\mu^{-1}$ where

$$\Omega_\mu^{-1}(i) = \begin{cases} \infty & \text{if } i \in R. \\ \deg(i) - 1 - \mu(i) & \text{if } i \notin R. \end{cases}$$

Proof. Let f be any G-multiparking function. First we show that Ω_f is a Dirichlet configuration. This is trivial if $R = V(G)$, so assume $R \subset V(G)$. As $\Omega_f(i) < \deg(i)$ for every vertex i, Ω_f is stable. By proposition 20, finding an avalanche connecting $(\Omega_f + \chi)$ to Ω_f is enough to show that Ω_f is recurrent.

Note that $(\Omega_f + \chi)(i) = \deg(i) - 1 - f(i) + \chi(i)$ for every nonroot vertex i. Therefore, a vertex i in the configuration $(\Omega_f + \chi)$ is ready if and only if $\chi(i) > f(i)$. Since f is a G-multiparking function, the set of all non-root vertices must have a well-behaved vertex, say j, and this implies $\chi(j) > f(j)$. Hence, $(\Omega_f + \chi)$ is not stable.

Let $i_1, i_2, \ldots, i_n$ be a burning sequence for f (in the sense of Lemma 10), with $i_1, i_2, \ldots, i_k$ as the roots of f. We have just shown that there is a vertex that can be labelled i_{k+1}. It is enough to show that if the vertices $i_{k+1}, \ldots, i_{l-1}$ can be fired, then i_l can be fired. Notice that for any $U \subseteq V(G)$, $\deg(i_l) = \mathbb{O}_U(i_l) + \mathbb{I}_U(i_l)$. So if $U = \{i_l, i_{l+1}, \ldots, i_n\}$, then firing i_{k+1} through i_{l-1} sends exactly $\mathbb{O}_U(i_l) - \chi(i_l)$ chips to i_l. So, i_l will have at least $\deg(i_l) - 1 - f(i_l) + \mathbb{O}_U(i_l)$ chips. Since f is a G-multiparking function, $f(i_l) < \mathbb{O}_U(i_l)$, so $\deg(i_l) - 1 - f(i_l) + \mathbb{O}_U(i_l) = \deg(i_l) - 1 - (f(i_l) - \mathbb{O}_U(i_l)) \geq \deg(i_l)$, and thus i_l will be ready. Hence, every non-root vertex in an avalanche beginning with $(\Omega_f + \chi)$ must be fired, and the throwing-out sequence specified is also a sequence in which the vertices can be fired. (Note that although there may be several throwing-out sequences for f, they all yield the same final configuration.) Note that μ is a Dirichlet configuration if and only if a firing sequence exists, and this argument can be reversed to obtain a burning sequence, proving that this correspondence is surjective.

Finally, we must show that this sequence of firings beginning at $(\Omega_f + \chi)$ ends at Ω_f. If i is any vertex, it loses $\deg(i)$ chips when fired. As its neighbors are fired, i recovers exactly $\deg(i) - \chi(i)$ chips, since the roots are not fired. Thus, at the end of this avalanche, i has exactly $\deg(i) - 1 - f(i) + \chi(i) - \deg(i) + (\deg(i) - \chi(i)) = \deg(i) - 1 - f(i)$ chips, meaning that we end on the configuration Ω_f.

Finally, it is obvious that Ω^{-1} is the inverse of Ω. $\square$

This result strengthens earlier work by Biggs (see Lemma 3(ii) in [3]). This simple bijection also provides information on the natural poset orders on the sets of G-multiparking functions and Dirichlet configurations with a given root set. If f is a G-multiparking function, it is immediate from the definition that any vertex function which is less than or equal to f on each vertex is also a G-multiparking function. This determines a simple poset order on the G-multiparking functions. Analogously, if μ is a Dirichlet configuration then any other configuration which is stable and greater than or equal to μ on every vertex is also Dirichlet. Hence there is also a simple poset order on the Dirichlet configurations and the Hasse diagrams of these two posets are identical, except that one is upside-down.

Corollary 22. If f and g are G-multiparking functions, then $f \leq g$ (in the G-multiparking function poset order described above) if and only if $\Omega_f \geq \Omega_g$ (in the Dirichlet configuration poset order described above).

Theorem 21 also suggests a burning-type algorithm for verifying that a configuration is Dirichlet for a given graph.

Corollary 23. A configuration μ on G is Dirichlet $\Longleftrightarrow$ there exists a permutation $\pi \in S_n$ such that for every vertex i, either $\pi(i)$ is a root or $\deg(\pi(i)) > \mu(\pi(i)) \geq \mathbb{I}_{U_i}(\pi(i))$, where $U_i := V(G) - \{\pi(1), \ldots, \pi(i-1)\}$.

Proof. By theorem 21, μ is critical if and only if $f := \Omega^{-1}(\mu)$ is a G-multiparking function, and this is true if and only if there is a permutation $\pi \in S_n$ such that $0 \leq f(\pi(i)) < \mathbb{O}_{U_i}(\pi(i))$ for every nonroot vertex i. But this is true if and only if

$$\begin{aligned} & 0 \leq \deg(\pi(i)) - 1 - \mu(\pi(i)) < \mathbb{O}_{U_i}(\pi(i)) \\ \iff\ & \deg(\pi(i)) - 1 \geq \mu(\pi(i)) > \deg(\pi(i)) - 1 - \mathbb{O}_{U_i}(\pi(i)) \\ \iff\ & \deg(\pi(i)) > \mu(\pi(i)) \geq \mathbb{I}_{U_i}(\pi(i)) \end{aligned}$$

This ends the proof. $\square$

We will hereafter refer to the permutations in proposition 23 as *Dirichlet certificates for μ*. This proposition also helps us identify the avalanches connecting a Dirichlet configuration to itself.

Proposition 24. Let μ be a Dirichlet configuration and let $\pi \in S_n$. π is a Dirichlet certificate for $\mu \iff$ the avalanche determined by the firing sequence $\pi(1), \pi(2), \ldots,$ $\pi(n)$ connects μ to itself.

Proof. ($\Longleftarrow$) Let $\alpha = (\mu = \mu_1, \mu_2, \ldots, \mu_n, \mu_1)$ be the avalanche determined by π and suppose $\pi(i)$ is a nonroot. We must show that $\deg(\pi(i)) > \mu_1(\pi(i)) \geq \mathbb{I}_{U_i}(\pi(i))$ for every such $i \leq n$. By assumption, μ is a Dirichlet configuration, so μ is stable, and thus $\deg(\pi(i)) > \mu_i(\pi(i))$ for every i.

The structure of α is that the vertices $\pi(1), \ldots, \pi(i-1)$ are fired, and after these firings we arrive at μ_i. Then $\mu_i(\pi(i)) - \mu_1(\pi(i)) = \mathbb{O}_{U_i}(\pi(i))$, since $U_i = V(G) - \{\pi(1), \ldots, \pi(i-1)\}$. Also, $\pi(i)$ is ready in μ_i and therefore $\mu_i(\pi(i)) \geq \deg(\pi(i))$.

Thus

$$\begin{aligned}
\mathbb{O}_{U_i}(\pi(i)) &= \mu_i(\pi(i)) - \mu_1(\pi(i)) \\
&\geq \deg(\pi(i)) - \mu_1(\pi(i)) \\
&= \mathbb{O}_{U_i}(\pi(i)) + \mathbb{I}_{U_i}(\pi(i)) - \mu_1(\pi(i))
\end{aligned}$$

Thus we have $\mu_1(\pi(i)) \geq \mathbb{I}_{U_i}(\pi(i))$, proving that π is a Dirichlet certificate for μ_1.

($\Longrightarrow$) If π is a Dirichlet certificate, then $\deg(\pi(i)) > \mu_1(\pi(i)) \geq \mathbb{I}_{U_i}(\pi(i))$ for every nonroot $\pi(i)$. Since $\mu = \mu_1$ is Dirichlet, it is stable, and thus only the roots can be fired. Suppose $\pi(1), \ldots \pi(i-1)$ have been fired in that order. Assuming $\pi(i)$ is not a root, $\mu_i(\pi(i)) = \mu_1(\pi(i)) + \mathbb{O}_{U_i}(\pi(i)) \geq \mathbb{I}_{U_i}(\pi(i)) + \mathbb{O}_{U_i}(\pi(i)) = \deg(\pi(i))$, and so $\pi(i)$ is ready in μ_i. Thus, $\pi(1), \ldots, \pi(n)$ defines an avalanche.

It is clear that this avalanche connects μ_1 to itself, since we begin at that configuration and every vertex is fired exactly once, meaning that the net change in chips at each vertex i is $\sum_{j \neq i} e(i,j) - \deg(i) = 0$. □

D. Descending Traversals

Let G be as above, but connected and with a total ordering $<_E$ on the edge set $E(G)$ and $V(G) = [n]$. Let $m = n + \#E(G)$. Let $\Sigma = \Sigma(G) = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ be a sequence of the edges and vertices of G in which each edge and vertex appears exactly once. Let $\Sigma^{\leq i} := (\sigma_1, \sigma_2, \ldots, \sigma_i)$ and $\Sigma^{\geq i} := (\sigma_i, \sigma_{i+1}, \ldots, \sigma_m)$. (Similarly, $\Sigma^{<i} := (\sigma_1, \sigma_2, \ldots, \sigma_{i-1})$ and $\Sigma^{>i} := (\sigma_{i+1}, \sigma_{i+2}, \ldots, \sigma_m)$.) We define Σ to be a *descending traversal on G* if it satisfies three conditions:

1. σ_1 is a vertex,

2. σ_i $(i \neq 1)$ a vertex $\Rightarrow$ σ_{i-1} is an edge adjacent to σ_i,

3. σ_i an edge $\Rightarrow$ it is adjacent to a vertex σ_k with $k < i$ and σ_i is maximal with respect to $<_E$ among all edges in $\Sigma^{\geq i}$ that are adjacent to some vertex in $\Sigma^{<i}$.

This definition is due to Cori and LeBorgne [6]. They provided explicit bijections from the descending traversals to the spanning trees and from the descending traversals to the critical configurations, and hence a bijection between these other two objects.

Now assume G is the same as above, except not necessarily connected, and $R \subseteq V(G)$. Let Σ^* be a list of some vertices and edges of G (Σ^* contains no repetitions). Let $\mathbb{E}(\Sigma^*)$ be the set of edges not in Σ^* which are adjacent to a vertex in Σ^*. We let Π_i be the set of ordered pairs (Σ^*, W) where $W \subseteq \mathbb{E}(\Sigma^*)$ and where not both of $W = \emptyset$ and $R \subseteq \Sigma^*$ is true.

We will now re-define another concept from Chapter II. A *choice function* on G is any function ζ from Π_i to $E(G) \cup R$ such that

$$\zeta(\Sigma^*, W) \begin{cases} \in W & \text{if } W \neq \emptyset \\ \in R - \Sigma^* & \text{if } W = \emptyset \text{ and } R \nsubseteq \Sigma^* \end{cases}$$

Fix a choice function ζ and let $\Sigma = (\sigma_i)_{i=1}^m$ be a sequence containing each edge and vertex of G exactly once. We call Σ a *descending R-traversal on G*, where $R = \{\sigma_{s_1}, \sigma_{s_2}, \ldots, \sigma_{s_k}\}$, such that each subsequence $S_i = (\sigma_{s_i}, \sigma_{s_i+1}, \ldots, \sigma_{s_{i+1}-1})$ of Σ satisfies:

1. $\sigma_{s_i} = \zeta(\Sigma^{<s_i}, \emptyset)$, where σ_{s_i} is a root,

2. $\sigma_j \in S_i$, $j > s_i$, a vertex $\Rightarrow \sigma_{j-1}$ is an edge adjacent to σ_j,

3. $\sigma_j \in S_i$ an edge $\Rightarrow \sigma_j$ is adjacent to a vertex σ_k with $k < j$ and $\sigma_j = \zeta(\Sigma^{\leq j-1}, \mathbb{E}(\Sigma^{\leq j-1}))$.

Let $\mathcal{DT} = \mathcal{DT}_{R,G,\zeta}$ denote the set of descending R-traversals on G. Note that the first condition and the requirement that Σ can be partitioned into subsequences S_i is not very restrictive. To check that a subsequence is a descending R-traversal it is generally only necessary to confirm that the last two conditions hold.

If one defines $R := \{v_1\}$ and ζ to be the function that picks the largest-index edge available, then the descending R-traversals of G are, in fact, just the descending traversals of G.

Example 6. We illustrate some descending R-traversals of Γ, for different R. In all these examples, let $\zeta(\Sigma^*, W)$ be the largest-index edge in W if $W \neq \emptyset$ and the smallest vertex in R otherwise.

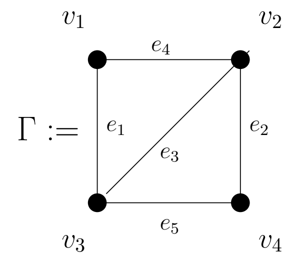


Fig. 15. A graph with total orders on the edges and vertices.

1. Let $R = \{v_1\}$. Then, $(v_1, e_4, v_2, e_3, e_2, v_4, e_5, v_3, e_1)$ and $(v_1, e_4, e_1, v_3, e_5, e_3, v_2, e_2, v_4)$ are descending R-traversals of Γ.
2. Let $R = \{v_2, v_3\}$. Then, $(v_2, e_4, e_3, e_2, v_3, e_5, e_1, v_1, v_4)$ and $(v_2, e_4, e_3, e_2, v_4, e_5, v_3, e_1, v_1)$ are descending R-traversals of Γ.
3. Let $R = \{v_1, v_2, v_4\}$. Then, $(v_1, e_4, e_1, v_2, e_3, v_3, e_5, e_2, v_4)$ is a descending R-traversal of Γ.

Now suppose Σ is a descending R-traversal, $R = \{\sigma_{s_1}, \sigma_{s_2}, \ldots, \sigma_{s_k}\}$, and ζ is the choice function. With this as input, we define a function f_Σ on $V(G)$ in the following way:

Algorithm A

1. If $v = \sigma_{s_i}$ for some i, then set $f_\Sigma(v) = \infty$.

2. Otherwise, set $f_\Sigma(v) = j - 1$, where j is the number of edges adjacent to v that precede v in Σ.

Note that if $v \notin R$, then by condition (2) of the definition of a descending R-traversal, it is preceded by an edge adjacent to it. Thus, $f_\Sigma(v) = j - 1 \geq 0$ and so $f_\Sigma : V(G) \to \mathbb{N} \cup \{\infty\}$.

Proposition 25. $f_\Sigma \in \mathcal{MP}$ for any $\Sigma \in \mathcal{DT}$.

Proof. Let $f = f_\Sigma$ and let $\sigma_{v_1}, \sigma_{v_2}, \ldots, \sigma_{v_n}$ be the vertex subsequence of Σ. We will show that this is a burning sequence for f, proving by Lemma 10 that f is a G-multiparking function. (It is clear that f has k roots.)

First, note that $f(\sigma_{v_1}) = \infty$. Let U_i be the set of vertices in $\Sigma^{\leq i}$. Now suppose $\sigma_{v_1}, \sigma_{v_2}, \ldots, \sigma_{v_{i-1}}$ are all either roots or well-behaved in $U_1, U_2, \ldots U_{i-1}$, respectively. Suppose σ_{v_i} is not a root. If $f(\sigma_{v_i}) = j - 1$, then there are exactly j edges adjacent to σ_{v_i} and preceding it in Σ. Each of these edges is preceded in Σ by a vertex adjacent to it (note part 3 of the definition of a descending R-traversal). These vertices are among $\{\sigma_{v_1}, \sigma_{v_2}, \ldots, \sigma_{v_{i-1}}\} = U_i$, and thus $0 \leq f(\sigma_{v_i}) = j - 1 < j = \mathbb{O}_{U_i}(\sigma_{v_i})$. Lemma 10 implies that $f \in \mathcal{MP}$. $\square$

Example 7. Let $\zeta(\Sigma^*, W)$ be the largest-index edge if $W \neq \emptyset$ and the lowest-index vertex in $R - \Sigma^*$ otherwise. Let $R = \{v_1, v_4\}$. Figure 16 is a table listing some descending R-traversals of Γ on the left-hand side and the corresponding (under algorithm A) Γ-multiparking functions on the right-hand side. (The list of descending R-traversals is not exhaustive.)

$$\begin{array}{lcl}
(v_1, e_4, e_1, v_4, e_5, v_3, e_3, v_2, e_2) & \to & (\infty, 1, 1, \infty) \\
(v_1, e_4, e_1, v_4, e_5, v_3, e_3, e_2, v_2) & \to & (\infty, 2, 1, \infty) \\
(v_1, e_4, e_1, v_4, e_5, e_2, v_2, e_3, v_3) & \to & (\infty, 1, 2, \infty) \\
\left.\begin{array}{l}(v_1, e_4, v_2, e_3, e_2, v_4, e_5, e_1, v_3) \\ (v_1, e_4, v_2, e_3, e_2, e_1, v_4, e_5, v_3)\end{array}\right\} & & (\infty, 0, 2, \infty) \\
\left.\begin{array}{l}(v_1, e_4, e_1, v_3, e_5, v_4, e_3, v_2, e_2) \\ (v_1, e_4, e_1, v_3, e_5, e_3, v_2, e_2, v_4)\end{array}\right\} & & (\infty, 1, 0, \infty) \\
\left.\begin{array}{l}(v_1, e_4, e_1, v_3, e_5, v_4, e_3, e_2, v_2) \\ (v_1, e_4, e_1, v_3, e_5, e_3, v_4, e_2, v_2)\end{array}\right\} & & (\infty, 2, 0, \infty) \\
\left.\begin{array}{l}(v_1, e_4, v_2, e_3, e_2, v_4, e_5, v_3, e_1) \\ (v_1, e_4, v_2, e_3, e_2, e_1, v_3, e_5, v_4)\end{array}\right\} & & (\infty, 0, 1, \infty) \\
\left.\begin{array}{l}(v_1, e_4, v_2, e_3, v_3, e_5, e_2, e_1, v_4) \\ (v_1, e_4, v_2, e_3, v_3, e_5, e_2, v_4, e_1)\end{array}\right\} & & (\infty, 0, 0, \infty)
\end{array}$$

Fig. 16. Some examples of algorithm A.

Lemma 1 of [6] states that if $(\sigma_i)_{i=1}^m$ and $(\tau_i)_{i=1}^m$ are descending traversals and k is the minimal index at which they differ, then one of σ_k and τ_k is an edge and the other is a vertex. The example above shows that this is not necessarily true for descending R-traversals; $(v_1, e_4, e_1, v_4, e_5, e_2, v_2, e_3, v_3)$ and $(v_1, e_4, e_1, v_3, e_5, v_4, e_3, e_2, v_2)$ do not observe this property.

Let $\Psi = \Psi_{R,G,\zeta} : \mathcal{DT} \to \mathcal{MP}$ be defined by $\Psi(\Sigma) = f_\Sigma$. The above example also illustrates that Ψ, as defined, is not generally injective. We will now define, for each graph G, root set R, and choice function ζ, a partition of $\mathcal{DT}$ over which Ψ will turn out to be injective.

Let f be any function from $V(G)$ to $\mathbb{N} \cup \{\infty\}$ such that $f(i) = \infty$ if and only if $i \in R$. We can consider $\Psi^{-1}(f)$, the (possibly empty) set of all descending R-traversals that are mapped to f. It is then clear that $\mathcal{R} = \mathcal{R}_{R,G,\zeta} := \{\Psi^{-1}(f) \mid \Psi^{-1}(f) \neq \emptyset\}$ is a partition of the set of descending R-traversals, where $\Psi^{-1}(f) = \{\Sigma \in \mathcal{DT} \mid \Psi(\Sigma) = f\}$. It is also clear that Ψ is constant over each $\Psi^{-1}(f)$ in $\mathcal{R}$, and that Ψ is injective when viewed as a function with $\mathcal{R}$ as its domain. Throughout the rest of this paper, we will view Ψ as a function from $\mathcal{R}$ to $\mathcal{MP}$.

Now we define an algorithm that will convert a a multiparking function to a descending R-traversal.

Algorithm B

- **Step 1: initial condition.** If $i = 1$ then $\Sigma^{\leq i} := (\zeta(\emptyset, \emptyset))$.

- **Step 2: insert the next entry.** Suppose $i > 1$. If there exists a vertex $v \notin \Sigma^{\leq i-1}$ such that $\Sigma^{\leq i-1}$ contains exactly $f(v) + 1$ edges adjacent to v, then $\Sigma^{\leq i} :=< \Sigma^{\leq i-1}, v >$. If no such vertex exists, then $\Sigma^{\leq i} :=$ $< \Sigma^{\leq i-1}, \zeta(\Sigma^{\leq i-1}, \mathbb{E}(\Sigma^{\leq i-1})) >$. Repeat this step until $i = m$.

Example 8. Recall the conditions in Example 7. Figure 17 is a table listing all the Γ-multiparking functions on the right-hand side and the corresponding (under algorithm B) descending R-traversals of Γ on the right-hand side.

$$\begin{array}{ccc}
(\infty,1,1,\infty) & \rightarrow & (v_1,e_4,e_1,v_4,e_5,v_3,e_3,v_2,e_2) \\
(\infty,2,1,\infty) & \rightarrow & (v_1,e_4,e_1,v_4,e_5,v_3,e_3,e_2,v_2) \\
(\infty,1,2,\infty) & \rightarrow & (v_1,e_4,e_1,v_4,e_5,e_2,v_2,e_3,v_3) \\
(\infty,0,2,\infty) & \rightarrow & (v_1,e_4,v_2,e_3,e_2,e_1,v_4,e_5,v_3) \\
(\infty,1,0,\infty) & \rightarrow & (v_1,e_4,e_1,v_3,e_5,e_3,v_2,e_2,v_4) \\
(\infty,2,0,\infty) & \rightarrow & (v_1,e_4,e_1,v_3,e_5,e_3,v_4,e_2,v_2) \\
(\infty,0,1,\infty) & \rightarrow & (v_1,e_4,v_2,e_3,e_2,e_1,v_3,e_5,v_4) \\
(\infty,0,0,\infty) & \rightarrow & (v_1,e_4,v_2,e_3,v_3,e_5,e_2,e_1,v_4)
\end{array}$$

Fig. 17. Some examples of algorithm B.

Proposition 26. $\Sigma^{\leq m} \in \mathcal{DT}$ for any $f \in \mathcal{MP}$.

Proof. We must first show that algorithm B can, in fact, always reach $\Sigma^{\leq m}$ if it acts on some $f \in \mathcal{MP}$. Clearly $\Sigma^{\leq 1}$ can be reached, so suppose $\Sigma^{\leq i} = (\sigma_j)_{j=1}^{i}$ can be reached for some $1 \leq i < m$. There are two cases in which Algorithm B might fail to reach $\Sigma^{\leq m}$.

First, suppose that there are two vertices v and w, neither in $\Sigma^{\leq i}$, such that there are exactly $f(v)+1$ and $f(w)+1$ edges in $\Sigma^{\leq i}$ that are adjacent to them respectively. (We may also assume, without loss of generality, that i is the minimum index at which there is more than one vertex ready to be appended to $\Sigma^{\leq i}$.) Note that the edge $\{v, w\}$, if it exists, is not in $\Sigma^{\leq i}$; no such edge could be in $\mathbb{E}(\Sigma^{\leq j})$ for any $j \leq i$ since neither v nor w is in $\Sigma^{\leq j}$. Therefore, in the sequence $\Sigma^{\leq i-1}, \Sigma^{\leq i-2}, \ldots, \Sigma^{\leq 1}$ there must be a $\Sigma^{\leq j}$ which contains exactly $f(v)+1$ edges adjacent to v but fewer than $f(w)+1$ edges adjacent to w. Thus, v should have been added earlier and i does not exist.

We must also show that there is no index i for which $R \subseteq \Sigma^{\leq i}$ and $\mathbb{E}(\Sigma^{\leq i}) = \emptyset$. Let i be an index for which there is no vertex $v \notin \Sigma^{\leq i}$ adjacent to exactly $f(v)+1$ edges in $\Sigma^{\leq i}$. Assume $R \subseteq \Sigma^{\leq i}$. Clearly, if $\Sigma^{\leq i}$ contains $V(G)$, then $\mathbb{E}(\Sigma^{\leq i})$ cannot be empty unless $i = m$. So let U be the set of vertices not in $\Sigma^{\leq i}$. Since $f \in \mathcal{MP}$ and there is no root in U, this set must have a well-behaved vertex. That is, there is a vertex $v \in U$ such that $0 \leq f(v) < \mathbb{O}_U(v)$. In particular, $0 < \mathbb{O}_U(v) \leq \#\mathbb{E}(\Sigma^{\leq i})$.

It is clear, from the construction of Algorithm B, that $\Sigma^{\leq m}$ satisfies the last two conditions in the definition of a descending R-traversal. □

Let $\Phi = \Phi_{R,G,\zeta} : \mathcal{MP} \to \mathcal{DT}$ be defined by $\Phi(f) = \Sigma^{\leq m}$.

Proposition 27. Φ is injective.

Proof. Let f and g be different functions in $\mathcal{MP}$, and $\Phi(f) = \Sigma_f$ and $\Phi(g) = \Sigma_g$. Since f and g are different, there is a vertex v at which $f(v) < g(v)$ (v is not a root, since f and g have the same root set). There is an index i at which v appears in Σ_f. This means $\Sigma_f^{\leq i} =< \Sigma_f^{\leq i-1}, v >$, but then either $\Sigma_f^{\leq i-1} \neq \Sigma_g^{\leq i-1}$ or $\Sigma_g^{\leq i} \neq< \Sigma_g^{\leq i-1}, v >$ and so $\Phi(f) \neq \Phi(g)$. □

Proposition 28. $\Psi(\Phi(f)) = f$ for any $f \in \mathcal{MP}$.

Proof. It is enough to show that $\Phi(f) \in \Psi^{-1}(f)$ for any $f \in \mathcal{MP}$. Suppose $\Phi(f) = (\sigma_i)_{i=1}^{m}$. Note that $\sigma_i \in R$ if and only if $\sigma_i = \zeta(\Sigma^{\leq i-1}, \emptyset)$ (where $\Sigma^{\leq i-1} = \emptyset$ if $i = 1$) and this is true if and only if $f(\sigma_i) = \infty$. Therefore $\Psi(\Phi(f))|_{\sigma_i} = \infty = f(\sigma_i)$. If $f(\sigma_i) = a$ for some vertex σ_i then $\Sigma^{\leq i-1}$ contains exactly $a+1$ edges adjacent to σ_i. Therefore, $\Psi(\Phi(f))|_{\sigma_i} = a = f(\sigma_i)$. So, Ψ maps $\Phi(f)$ to f and thus $\Phi(f) \in \Psi^{-1}(f)$. $\square$

CHAPTER IV

A DOUBLE-PARKING FUNCTION

A. Introduction

Here, we will consider a new generalization. Throughout this paper, let $X = (x_0, x_1, \ldots, x_q) \in \mathbb{N}^{q+1}$ and $Y = (y_0, y_1, \ldots, y_p) \in \mathbb{N}^{p+1}$, where $0 \leq x_i \leq x_{i+1} \leq x_q = x$ and $0 \leq y_j \leq y_{j+1} \leq y_p = y$. Let G be the complete bipartite graph $K_{p,q}$, where the bipartite vertex sets are $P = \{v_1, v_2, \ldots, v_p\}$ and $Q = \{v_{p+1}, v_{p+2}, \ldots, v_{p+q}\}$. This situation is illustrated in Figure 18 below…

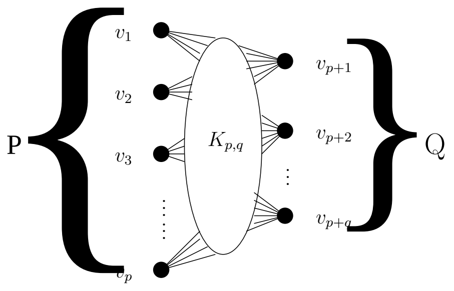


Fig. 18. The basic setting.

If $v \in U$ recall that $\mathbb{O}_U(v)$ denotes the *outdegree* of vertex v in U; that is, $\mathbb{O}_U(v) := \#\{w \notin U | \{w, v\} \in E(G)\}$. An $(\mathbf{X}, \mathbf{Y})$–**parking function** is a function $\tau : V(G) \to \mathbb{N}$, such that for every proper subset $U \subsetneq V(G)$, there exists a $v \in U$ satisfying

$$0 \leq \tau(v) < \begin{cases} x_{\mathbb{O}_U(v)} & \text{if } v \in Q \\ & \qquad \textit{or} \\ y_{\mathbb{O}_U(v)} & \text{if } v \in P \end{cases}$$

A vertex v that satisfies one of these two properties is said to be *well-behaved* (in U). Let $PF_{p,q}(\mathbf{X}, \mathbf{Y})$ be the total number of $(\mathbf{X}, \mathbf{Y})$–parking functions. We will often use the notation $\tau = (\tau_1, \tau_2, \ldots, \tau_p, \tau_{p+1}, \ldots, \tau_{p+q})$, where $\tau_i = \tau(v_i)$. (Our notation for sequences will vary somewhat; if u is a sequence, we will sometimes denote its i^{th} term by u_i and sometimes by $u(i)$.)

The indexing scheme defining the $(\mathbf{X}, \mathbf{Y})$–parking function will seem awkward at first sight. However, it becomes more natural when one compares it to that for a G-parking function as defined in chapter II. Recall that if G is an undirected graph with vertex set $V(G) = \{v_0, v_1, \ldots, v_n\}$, one can define a *$G$-parking function* to be a function $f : V(G) \to \mathbb{N} \cup \infty$ if (1) $f(v) = \infty \Leftrightarrow v = v_0$ and (2) for every vertex subset $U \subseteq V(G) - \{v_0\}$, there exists a $v \in U$ such that $0 \leq f(v) < \mathbb{O}_U(v)$. (This definition is equivalent to the one given in chapter I. In the setting of G-parking functions, the outdegree is itself a local bound on the value of f at a given vertex; in the setting of $(\mathbf{X}, \mathbf{Y})$–parking functions, the outdegree only provides the index for a vector, which determines the bound.

There is also a similarity between $(\mathbf{X}, \mathbf{Y})$–parking functions and x-parking functions. Recall that if $x = (x_1, x_2, \ldots, x_n) \in \mathbb{N}^n$ is a vector, then $(p_1, p_2, \ldots, p_n)$ is an *x-parking function* if, for every index i, $p_i \leq x_1 + \ldots + x_i$. We stress, however, that the $(\mathbf{X}, \mathbf{Y})$–parking function is not a generalization of either of these two generalized parking functions. This is clear for G-parking functions, since $(\mathbf{X}, \mathbf{Y})$–parking functions are never ∞-valued.

In section B, we will show how the $(\mathbf{X}, \mathbf{Y})$–parking function is related to the (p, q)–parking function and exhibits some properties similar to other generalized parking functions. In section C, we will show how the operator-theoretic machinery from a series of papers by Kung and Yan (see [18], [19], and [20]) provides important facts about the number of $(\mathbf{X}, \mathbf{Y})$–parking functions. In section D, we will show how $(\mathbf{X}, \mathbf{Y})$–parking functions and (p, q)–parking functions are related to lattice paths.

B. The Decompositional Formula

For parking functions generalized to graphs, such as the G-parking and G-multiparking functions, there is a *burning algorithm* to determine, in $O(\#V(G))$ number of operations, whether a vertex function is a generalized parking function. A burning algorithm can be thought of as any procedure which establishes an ascending chain $A_1 \subsetneq A_2 \subsetneq \ldots \subsetneq A_{\#V(G)} = V(G)$ of vertex subsets, each of which contains a well-behaved (in the appropriate sense) vertex. In order for the existence of such a chain to prove that *every* vertex subset has a well-behaved vertex, an analog of the following lemma must hold true.

There is a burning algorithm (see chapter I, section 4) in this situation. It is easy to see that lemma 6 holds here, and there is an analogous version of proposition 10.

Proposition 29. A vertex function τ is an $(\mathbf{X}, \mathbf{Y})$–parking function if and only if there exists a permutation $\pi \in S_{p+q}$ so that $v_{\pi(i)}$ is a root or well-behaved in $W_i := \{v_{\pi(i)}, v_{\pi(i+1)}, \ldots, v_{\pi(p+q)}\}$, for each $1 \le i \le p+q$.

Proof. If τ is an $(\mathbf{X}, \mathbf{Y})$–parking function, then it is clear that the latter condition holds. Conversely, first note that every proper vertex subset $U \subsetneq V(G)$ is a subset of at least one of the sets $V(G) - \{v_1\}, V(G) - \{v_2\}, \ldots, V(G) - \{v_{p+q}\}$. If the latter condition in the proposition holds, then there is some minimal i for which $U \subseteq W_i$. Lemma 6 now implies U has a well-behaved vertex and thus τ is an $(\mathbf{X}, \mathbf{Y})$–parking function. □

Definition 2. A permutation π of the sort described in Proposition 29 is called a *certificate* for τ.

It is important to note that there may exist multiple certificates for an $(\mathbf{X}, \mathbf{Y})$–

parking function. This fact is easily observed if either P or Q contains two vertices with the same τ-value. There is a concrete example of this behavior below.

Example 1. We will refer to the below example throughout what follows. The graph illustrated in Figure 19

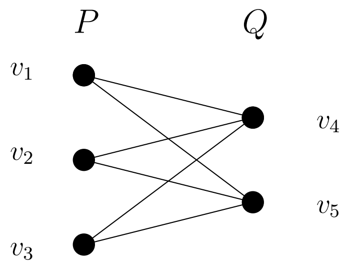


Fig. 19. An example of an $(\boldsymbol{X},\boldsymbol{Y})$-parking function.

with the function $\tau = (2,4,0,2,2)$ is an $(\mathbf{X},\mathbf{Y})$–parking function for $\mathbf{X} = (1,3,4,4)$ and $\mathbf{Y} = (1,4,5)$. One can see this by applying Proposition 29 with the certificate (24153). (Note that this certificate is not unique; (25143) is also a certificate.)

The given definition of an $(\mathbf{X},\mathbf{Y})$–parking function is difficult to work with, and is more naturally thought of as a function whose various statistics are other kinds of parking functions (see Lemma 31 below). With the below notation, we can think of $(\mathbf{X},\mathbf{Y})$–parking functions as "(q,p)-sequences" for $|P| = p$ and $|Q| = q$.

Definition 3. An (x,y)*–sequence* is a sequence $(a_1,\ldots,a_p,a_{p+1},\ldots,a_{p+q})$ such that $0 \le a_i \le x$ if $i \le p$, and $0 \le a_i \le y$ if $i > p$.

Definition 4. Let π be a permutation in S_{p+q}. We define its associated (p,q)-sequence as $\widehat{\pi} = (\widehat{\pi}_1,\ldots,\widehat{\pi}_{p+q})$, where

$$\widehat{\pi}_i = \begin{cases} |\{k|\pi_k < \pi_i, k > p\}| & \text{if } i \le p; \\ |\{k|\pi_k < \pi_i, k \le p\}| & \text{if } i > p. \end{cases}$$

A (p, q)*–parking function* is a (p, q)-sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{p+q})$ for which there exists a permutation $\pi \in S_{p+q}$ such that $\sigma_i \leq \widehat{\pi}_i$ for all i. We will denote this by $\sigma \leq \widehat{\pi}$.

Example 2. Let $p = 3$ and $q = 2$. Consider the $(3, 2)$-sequence $(0, 1, 0, 0, 1)$. If $\pi = (13425)$, then $\widehat{\pi} = (01113)$ and $(0, 1, 0, 0, 1) \leq (0, 1, 1, 1, 3)$ and thus $(0, 1, 0, 0, 1)$ is a $(3, 2)$-parking function. (Note that other permutations can verify that this is a $(3, 2)$-parking function; $\pi = (25314)$ also works.)

Definition 4 is due to Cori and Poulalhon [7]. They proved that there are $(p + q + 1)(p + 1)^{q-1}(q + 1)^{p-1}$ (p, q)-parking functions, and that a (p, q)-sequence (u, v) is an (x, y)-parking function if and only if the concatenation of u and v is a parking function. This latter fact has an analogous result here, given in the next lemma.

Lemma 30. An (x, y)–sequence $(a_1, \ldots, a_{p+q})$ is an $(\mathbf{X}, \mathbf{Y})$–parking function if and only if there exists a permution π with associated sequence $\widehat{\pi}$, such that $a_i < x_{\widehat{\pi}_i}$ if $i \leq p$, and $a_i < y_{\widehat{\pi}_i}$ if $i > p$.

Proof. Suppose $(a_1, \ldots, a_{p+q})$ is an $(\mathbf{X}, \mathbf{Y})$–parking function and let π be a certificate for it. Let

$$U_i := \begin{cases} \{v_j | \pi_j < \pi_i, j > p\} & \text{if } i \leq p \\ \{v_j | \pi_j < \pi_i, j \leq p\} & \text{if } i > p \end{cases}$$

Then it is clear that $\widehat{\pi}_i = |U_i| = \mathbb{O}_{U_i}(v_i)$ and therefore $a_i < x_{\mathbb{O}_{U_i}(v_i)} = x_{\widehat{\pi}_i}$ (or $y_{\widehat{\pi}_i}$). Conversely, one can construct the U_i's and therefore a certificate π for $(a_1, \ldots, a_{p+q})$ proving that it is an $(\mathbf{X}, \mathbf{Y})$–parking function. $\square$

The remainder of this section will establish a recursive formula for the number of $(\mathbf{X}, \mathbf{Y})$–parking functions. To do this, we first show how $(\mathbf{X}, \mathbf{Y})$–parking functions can be described as various statistics of other parking functions. Then, we will prove that there is a decomposition of each half of an (x, y)–sequence into two sub-sequences, one of which "looks like" an $(\mathbf{X}, \mathbf{Y})$–parking function and the other is just a bounded sequence. This decomposition is the key to proving the recursive formula.

Definition 5. Given a sequence $\mu = (u_1, \ldots, u_n)$, let $\rho_\mu(i)$ be the location of u_i in the order statistics of μ. If $\nu = (\nu_1, \nu_2)$ is an (x, y)–sequence, then we define the *order sequence* $\operatorname{ord}(\nu)$ to be $(\operatorname{ord}(\nu_1), \operatorname{ord}(\nu_2))$, where $\operatorname{ord}(\mu) = (\mu_1 + \rho_\mu(1), \mu_2 + \rho_\mu(2), \ldots, \mu_l + \rho_\mu(l))$.

Definition 6. The *rank sequence* of an $(\mathbf{X}, \mathbf{Y})$–parking function τ is the sequence $\widetilde{\tau} = (\widetilde{\tau}_1, \ldots, \widetilde{\tau}_{p+q})$ with $x_{\widetilde{\tau}_i - 1} \le \tau_i < x_{\widetilde{\tau}_i}$ if $i \le p$, and $y_{\widetilde{\tau}_i - 1} \le \tau_i < y_{\widetilde{\tau}_i}$ if $i > p$, with the additional convention that $x_{-1} = y_{-1} = 0$.

Example 3. Consider the $(\mathbf{X}, \mathbf{Y})$–parking function τ from Example 1. The order sequence for τ (when viewed as a $(3, 2)$-sequence) is $\operatorname{ord}(\tau) = (1, 3, 7, 3, 4)$. The rank sequence for τ is $\widetilde{\tau} = (1, 2, 0, 1, 1)$.

Lemma 31. The following are equivalent ...

1. τ is an $(\mathbf{X}, \mathbf{Y})$–parking function.
2. There exists a permutation π such that $\widetilde{\tau}_i \le \widehat{\pi}_i$ for all i.
3. $\widetilde{\tau}$ is a (p, q)–parking function.
4. $\operatorname{ord}(\widetilde{\tau})$ is a regular parking function.

Proof. Suppose a permutation π as described in (2) exists. $\widetilde{\tau}_i$ is the smallest integer such that $\tau_i < x_{\widetilde{\tau}_i}$. By (2) we have $\tau_i < x_{\widetilde{\tau}_i} \le x_{\widehat{\pi}_i}$. If

$$U_i := \begin{cases} V(G) - \{v_j | \pi_j < \pi_i, j > p\} & \text{if } i \le p \\ V(G) - \{v_j | \pi_j < \pi_i, j \le p\} & \text{if } i > p \end{cases}$$

then $\widehat{\pi}_i = \mathbb{O}_{U_i}(v_i)$ (since G is a complete bipartite graph). So $\tau_i < x_{\widetilde{\tau}_i} \leq x_{\widehat{\pi}_i} = x_{\mathbb{O}_{U_i}(v_i)}$ and thus τ is an $(\mathbf{X}, \mathbf{Y})$–parking function. Thus (2) implies (1). Now suppose τ is an $(\mathbf{X}, \mathbf{Y})$–parking function and let π be a certificate for τ. Note that $\widehat{\pi}_i = |V(G) - U_i|$, where U_i are the sets defined above. By the burning algorithm, $\tau_i < x_{\widehat{\pi}_i}$. Since $\widetilde{\tau}_i$ is the least index j for which $\tau_i < x_j$, we have $x_{\widetilde{\tau}_i} \leq x_{\widehat{\pi}_i}$. As $\mathbf{X}$ is a nondecreasing sequence, we have $\widetilde{\tau}_i \leq \widehat{\pi}_i$. Thus (1) implies (2).

Finally, (2) $\Leftrightarrow$ (3) is a restatement of the definition of a (p, q)–parking function and (3) $\Leftrightarrow$ (4) is proven in [7], Proposition 3. $\square$

We are primarily interested in $\text{ord}(\widetilde{\tau})$, and hereafter we will replace this cumbersome notation with $\overrightarrow{\tau}$.

Example 4. Let τ be the same as in Example 1. It is easy to see that $\widetilde{\tau}$ is a (p, q)-parking function, since $(1, 2, 0, 1, 1)$ is a parking function of length 5. (This is another way of verifying that a function τ is an $(\mathbf{X}, \mathbf{Y})$–parking function; we do not need to use Proposition 29.)

The following lemma gives us a decomposition of an $(\mathbf{X}, \mathbf{Y})$–parking function into Here, we adopt the notation that if $S = (a_1, \ldots, a_n)$ is a sequence, then the subsequence of $(a_1, \ldots, a_i)$ will be denoted $S|_{\leq i}$.

Lemma 32. For any (x, y)–sequence $\tau = (\tau_1, \ldots, \tau_p, \tau_{p+1}, \ldots, \tau_{p+q})$ be an (x, y)–sequence, there exist integers i and j, such that $(\tau_1, \ldots, \tau_p)$ can be decomposed into two parts $(\tau_{r_1}, \ldots, \tau_{r_i})$ and $(\tau_{r_{i+1}}, \ldots, \tau_{r_p})$, while $(\tau_{p+1}, \ldots, \tau_{p+q})$ into $(\tau_{r_{p+1}}, \ldots, \tau_{r_{p+j}})$ and $(\tau_{r_{p+j+1}}, \ldots, \tau_{r_{p+q}})$. Additionally, $(\tau_{r_1}, \ldots, \tau_{r_i}, \tau_{r_{p+1}}, \ldots, \tau_{r_{p+j}})$ is an $(\mathbf{X}|_{\leq \mathbf{j}}, \mathbf{Y}|_{\leq \mathbf{i}})$–parking function, while $(\tau_{r_{i+1}}, \ldots, \tau_{r_p})$ is in the discrete interval $[x_j, x_q)$, and $(\tau_{r_{p+j+1}}, \ldots, \tau_{r_{p+q}})$ in $[y_i, y_p)$. Furthermore, this decomposition provides a bijection between all (x, y)–sequences and sequence quartets of the type above.

Proof. For any (x, y)–sequence $\tau = (u, v)$, with u and v of lengths p and q respectively, consider its rank sequence $\widetilde{\tau} = (\widetilde{u}, \widetilde{v})$, and correspondingly the order sequence $\overrightarrow{\tau} = (\overrightarrow{u}, \overrightarrow{v})$. Let σ be the order statistic of the latter sequence. Let k be the first occurance in σ such that $\sigma_k > k$. Notice those first $k-1$ numbers in the sequence form a parking function of length $k-1$. Assume there are i numbers in $(\sigma_1, \ldots, \sigma_{k-1})$ that orginate from u, and j numbers from v. Then those $i + j$ numbers form an $(\mathbf{X}|_{\leq \mathbf{j}}, \mathbf{Y}|_{\leq \mathbf{i}})$–parking function by Lemma 31.

Supposely $\sigma_k \leq \overrightarrow{u}_l$, since $i+j = k-1 < \sigma_k - 1 = \overrightarrow{u}_l - 1 = \widetilde{u}_l + \rho_{\widetilde{u}}(l) - 1 = \widetilde{u}_l + i$, so $\widetilde{u}_l > j$, thus $u_l \geq x_j$. Similarly, if $\sigma_k \leq \overrightarrow{v}_l$, $v_l \geq y_i$. Hence all the remaining numbers in τ, which are *not* in those forming the $(\mathbf{X}|_{\leq \mathbf{j}}, \mathbf{Y}|_{\leq \mathbf{i}})$–parking function above, must be greater than either x_j or y_i respectively. This proves the existence of the decomposition.

Conversely, any sequence quartet of this type can be assembled into an (x, y)–sequence, which gives the inverse mapping of the decomposition. Therefore, both mappings are bijections. $\square$

This lemma implies the following decomposition for enumerating (x, y)–sequences.

$$x^p y^q \;=\; \sum_{i=0}^{p} \sum_{j=0}^{q} \binom{p}{i} (x - x_j)^{p-i} \binom{q}{j} (y - y_i)^{q-j} PF_{j,i}(x_0, \ldots, x_j; y_0, \ldots, y_i).$$

In the next section, we will show how this identity yields other combinatorial identities.

C. A Theory of Bivariate Goncarov Polynomials

In the previous section, we observed enumerative properties of $(\mathbf{X}, \mathbf{Y})$–parking functions related to other kinds of parking functions. In this section, we will show how a modified version of the operator-theoretic approach of Kung and Yan (see [18], [19],

[20]) also yields enumerative properties of $(\mathbf{X}, \mathbf{Y})$–parking functions, but this time related to generalized "Goncarov polynomials." [1]

Definition 7. Let $\mathbb{K}[x, y]$ be a bivariate polynomial ring, where $\mathbb{K}$ is any field of characteristic 0. For $p(x, y) \in \mathbb{K}[x, y]$, define $D_x p(x, y)$ and $D_y p(x, y)$ to be differentiation of p with respect to x and y, respectively. For each pair of non-negative integers r and s we define an infinite-dimensional array of integers $(b^{r,s}_{i,j})^{\infty}_{i,j=0}$ and define the operators

$$\Phi_{r,s} := \sum_{i,j=0}^{\infty} b^{r,s}_{i,j} D_x^{r+i} D_y^{s+j}.$$

A *(bivariate) Goncarov polynomial* is a polynomial $g_{m,n}(x, y) = \sum_{i=0}^{m} \sum_{j=0}^{n} a_{i,j} x^i y^j \in \mathbb{K}[x, y]$, where for all r and s,

$$\Phi_{r,s} \left(g_{m,n}(x, y) \right)\big|_{x,y=0} = m!n!\delta_{rm}\delta_{sn}. \tag{4.1}$$

For each r and s, Figure 20 shows the system of equations resulting from this definition.

$$\begin{array}{rcl}
b^{0,0}_{0,0}a_{0,0} + b^{0,0}_{0,1}a_{0,1} + b^{0,0}_{1,0}a_{1,0} + 2!b^{0,0}_{2,0}a_{2,0} + 2!b^{0,0}_{0,2}a_{0,2}) + b^{0,0}_{1,1}a_{1,1} + \ldots + m!n!b^{0,0}_{m,n}a_{m,n} & = & 0 \\
b^{1,0}_{0,0}a_{1,0} + b^{1,0}_{0,1}a_{1,1} + 2!b^{1,0}_{1,0}a_{2,0} + 3!b^{1,0}_{2,0}a_{3,0} + 2!b^{1,0}_{0,2}a_{1,2} + 2!b^{1,0}_{1,1}a_{2,1} + \ldots + m!n!b^{1,0}_{m-1,n}a_{m,n} & = & 0 \\
b^{0,1}_{0,0}a_{0,1} + 2!b^{0,1}_{0,1}a_{0,2} + b^{0,1}_{1,0}a_{1,1} + 3!b^{1,0}_{2,0}a_{3,0} + 2!b^{0,1}_{1,1}a_{1,2} + 3!b^{0,1}_{0,2}a_{0,3} + \ldots + m!n!b^{0,1}_{m,n-1}a_{m,n} & = & 0 \\
 & \vdots & \\
m!n!b^{m,n}_{0,0}a_{m,n} & = & m!n!
\end{array}$$

Fig. 20. The system of $(r + 1)(s + 1)$ equations that condition 4.1 induces.

[1]The correct, untransliterated spelling is "Gončarov", but we will suppress the accent here.

1. Goncarov Polynomials and Matrices

Throughout this section, we will consider the special case when

$$\Phi_{rs} := \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \frac{a_s^i b_r^j}{i!j!} D_x^{r+i} D_y^{s+j} \tag{4.2}$$

It is important to note that this sum is not generally a product of two sums, because of the exponents of a_s and b_r.

This system of equations can be restated in terms of matrices. Let $\bar{\mathbf{0}}$ be the $(n+1)\times(n+1)$ matrix. For each $0 \le i \le j \le m$, define the $(n+1)\times(n+1)$ matrices $B[i,j]$, $B_p[0]$, and $B[i,j;p]$ by

$$(l,k)^{\text{th}} \text{ entry of } B[i,j] := \begin{cases} (i+j)!(l+k-2)!b_{j,k-l}^{i,l-1} & \text{if } 0 \le k-l \\ 0 & \text{otherwise} \end{cases}$$

$$(l,k)^{\text{th}} \text{ entry of } B_p[i,j] := \begin{cases} (i+j)!(l+k-2)!b_{j,k-l}^{i,l-1} & \text{if } 0 \le k-l \text{ and } l \ne n+1 \\ \frac{x^p y^{k-1}}{p!(k-1)!} & \text{if } l = n+1 \\ 0 & \text{otherwise} \end{cases}$$

$$(l,k)^{\text{th}} \text{ entry of } B_p[0] := \begin{cases} \frac{x^p y^{k-1}}{p!(k-1)!} & \text{if } l = n+1 \\ 0 & \text{otherwise} \end{cases}$$

Figure 21 below illustrates more explicitly what these matrices look like.

$$B[i,j] := (i+j)! \begin{pmatrix} b^{i,0}_{j,0} & 1!b^{i,0}_{j,1} & \dots & \dots & n!b^{i,0}_{j,n} \\ 0 & 1!b^{i,1}_{j,0} & 2!b^{i,1}_{j,1} & \dots & n!b^{i,1}_{j,n-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & (n-1)!b^{i,n-1}_{j,0} & n!b^{i,n-1}_{j,1} \\ 0 & 0 & \dots & 0 & n!b^{i,n}_{j,0} \end{pmatrix}$$

$$B_p[i,j] := \begin{pmatrix} (i+j)!b^{i,0}_{j,0} & (i+j)!1!b^{i,0}_{j,1} & \dots & \dots & (i+j)!n!b^{i,0}_{j,n} \\ 0 & (i+j)!b^{i,1}_{j,0} & (i+j)!1!b^{i,1}_{j,1} & \dots & (i+j)!n!b^{i,1}_{j,n-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & (i+j)!(n-1)!b^{i,n-1}_{j,0} & (i+j)!n!b^{i,n-1}_{j,1} \\ \frac{x^p}{p!} & \frac{x^p y}{p!1!} & \dots & \frac{x^p y^{n-1}}{p!(n-1)!} & \frac{x^p y^n}{p!n!} \end{pmatrix}$$

$$B_p[0] := \begin{pmatrix} 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \dots & \dots & \vdots \\ 0 & 0 & \dots & \dots & 0 \\ \frac{x^p}{p!} & \frac{x^p y}{p!1!} & \dots & \frac{x^p y^{n-1}}{p!(n-1)!} & \frac{x^p y^n}{p!n!} \end{pmatrix}$$

Fig. 21. The matrix representations of the defining goncarov equations.

Then Definition 7 is equavalent to $BA = M$, where B, A, and M are the matrices depicted in Figure 22.

$$B := \begin{pmatrix} B[0,0] & B[0,1] & B[0,2] & \dots & \dots & B[0,m] \\ \mathbf{\bar{0}} & B[1,0] & B[1,1] & \dots & \dots & B[1,m-1] \\ \mathbf{\bar{0}} & \mathbf{\bar{0}} & B[2,0] & \dots & \dots & B[2,m-2] \\ \mathbf{\bar{0}} & \mathbf{\bar{0}} & \mathbf{\bar{0}} & \ddots & \dots & B[3,m-3] \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{\bar{0}} & \mathbf{\bar{0}} & \mathbf{\bar{0}} & \dots & \mathbf{\bar{0}} & B[m,0] \end{pmatrix} ;\ A := \begin{pmatrix} a_{0,0} \\ a_{0,1} \\ \vdots \\ a_{0,n} \\ a_{1,0} \\ a_{1,1} \\ \vdots \\ a_{m,n} \end{pmatrix} ;\ M := \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ m!n! \end{pmatrix}$$

Fig. 22. The matrix equation for the defining goncarov equations.

Using Cramer's rule and Laplace's expansion to solve and regroup the results, we arrive at a determinental formula, as depicted in Figure 23.

2. Recursions and Relations

Theorem 33. The following recursive identities hold.

1. For any polynomial $p(x,y)$ of order (m,n),

$$p(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} \frac{\Phi_{i,j}(p(x,y))|_{x,y=0}}{i!j!} g_{i,j}(x,y), \tag{4.3}$$

$$g_{m,n}(x,y) = \frac{m!n!}{\prod_{i=0}^{m}\prod_{j=0}^{n} b_{00}^{ij}} \begin{vmatrix} B[0,0] & B[0,1] & B[0,2] & \dots & \dots & B[0,m] \\ \bar{\mathbf{0}} & B[1,0] & B[1,1] & \dots & \dots & B[1,m-1] \\ \bar{\mathbf{0}} & \bar{\mathbf{0}} & B[2,0] & \dots & \dots & B[2,m-2] \\ \bar{\mathbf{0}} & \bar{\mathbf{0}} & \bar{\mathbf{0}} & \ddots & \dots & B[3,m-3] \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \bar{\mathbf{0}} & \bar{\mathbf{0}} & \bar{\mathbf{0}} & \dots & B[m-1,0] & B[m-1,1] \\ B_0[0] & B_1[0] & B_2[0] & \dots & B_{m-1}[0] & B_m[m,0] \end{vmatrix}$$

Fig. 23. The determinental formula.

2. The *linear recursion* becomes

$$x^m y^n = \sum_{i=0}^{m}\sum_{j=0}^{n} \frac{m!n!}{i!j!} b_{m-i,n-j}^{i,j} g_{i,j}(x,y). \tag{4.4}$$

3. the *Appell relation* is

$$e^{(x+y)t} = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \frac{p_{i,j}(t) g_{i,j}(x,y)}{i!j!}, \tag{4.5}$$

where $p_{i,j}(t) = t^{i+j} \sum_{m,n=0}^{\infty} b_{m,n}^{i,j} t^{m+n}$.

Now we can consider the special case when

$$\phi_{r,s} = \sum_{i,j=0}^{\infty} \frac{a_s^i b_r^j}{i!j!} D_x^{r+i} D_y^{s+j}.$$

The determental formula now becomes

$$g_{m,n}(x,y) = m!n! \begin{vmatrix} \frac{a_0^0 b_0^0}{0!0!} & \frac{a_0^0 b_0^1}{0!1!} & \cdots & \frac{a_0^0 b_0^n}{0!n!} & \frac{a_0^1 b_0^0}{1!0!} & \cdots & \frac{a_0^m b_0^n}{m!n!} \\ 0 & \frac{a_1^0 b_0^0}{0!0!} & \cdots & \frac{a_1^0 b_0^{n-1}}{0!(n-1)!} & 0 & \cdots & \frac{a_1^m b_0^{n-1}}{m!(n-1)!} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{a_n^0 b_0^0}{0!0!} & 0 & \cdots & \frac{a_n^m b_0^0}{m!0!} \\ 0 & 0 & \cdots & 0 & \frac{a_0^0 b_1^0}{0!0!} & \cdots & \frac{a_0^{m-1} b_1^n}{(m-1)!n!} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & y & \cdots & \frac{y^n}{n!} & x & \cdots & \frac{x^m y^n}{m!n!} \end{vmatrix}$$

3. The Shift Formulas

Goncarov polynomials exhibit periodic behavior that is both computationally convenient and reveals a close relationship between them and the number of $(\mathbf{X}, \mathbf{Y})$–parking functions, functions we defined in section A above. If $a \in \mathbb{K}$ and $p(x, y) \in \mathbb{K}[x, y]$, define the *shift operators* $E_x(a)p(x, y) := p(x+a, y)$ and $E_y(a)p(x, y) := p(x, y+a)$. The shift operators can be represented as differential operators, as the next proposition shows.

Proposition 34. For any $a \in \mathbb{K}$, $E_x(a) = \sum_{l=0}^{\infty} \frac{a^l}{l!} D_x^l$ and $E_y(a) = \sum_{l=0}^{\infty} \frac{a^l}{l!} D_y^l$.

Proof. Observe how the operator $\frac{a^l}{l!} D_x^l$ acts on a monomial $c_{ij} x^i y^j$ of $p(x, y)$:

$$\frac{a^l}{l!} D_x^l [c_{ij} x^i y^j] = \begin{cases} 0 & \text{if } i < l \\ c_{ij} \frac{a^l}{l!} \frac{i!}{(i-l)!} x^{i-l} y^j & \text{if } i \geq l \end{cases} = \begin{cases} 0 & \text{if } i < l \\ c_{ij} \binom{i}{l} a^l x^{i-l} y^j & \text{if } i \geq l \end{cases}$$

In $p(x + a, y)$, this monomial becomes $c_{ij}(x + a)^i y^j =$
$\sum_{l=0}^{i} c_{ij} \binom{i}{l} a^l x^{i-l} y^j = \sum_{l=0}^{i} \frac{a^l}{l!} D_x^l [c_{ij} x^i y^j]$. So, $p(x + a, y) = \sum_i \sum_j c_{ij}(x + a)^i y^j = \sum_i \sum_j \sum_l D_x^l [c_{ij} x^i y^j] = \sum_l D_x^l \sum_i \sum_j c_{ij} x^i y^j = \sum_l D_x^l p(x, y)$. A similar argument holds for the y variable. $\square$

With Proposition 34 and the definition of $\phi_{r,s}$, the following is easily observed.

Corollary 35. $\phi_{r,s} p(x, y) = D_x^r D_y^s E_x(a_s) E_y(b_r) p(x, y) = D_x^r D_y^s p(x + a_s, y + b_r)$.

We pause here to note that $g_{m,n}(x, y)$ can be expressed as
$g_{m,n}(x; a_0, \ldots, a_n; y; b_0, \ldots, b_m)$, where the a_i and the b_i are coefficients given in the $\Phi_{r,s}$ operator in Definition 4.2. We will adopt this representation for the rest of this chapter. Note that $g_{m,n}(x; a_0, \ldots, a_n; y; b_0, \ldots, b_m)$ has the initial conditions

$$g_{m,n}(a_0; a_0, \ldots, a_n; b_0; b_0, \ldots, b_m) = \delta_{0n} \delta_{0m} \tag{4.6}$$

The first two equations can be obtained by checking the uniqueness of the Goncarov polynomials, whilest the last by setting $r = s = 0$ and $p(x, y) = g_{m,n}(x, y)$ in Corollary 35 and applying the definition of Goncarov polynomials 4.1.

Proposition 36. The polynomial $g_{m,n}(x; a_0, \dots, a_n; y; b_0, \dots, b_m)$ exhibits the following properties ...

1. $g_{m,0}(x; a_0; y; b_0, \dots, b_m) = (x - a_0)^m$ and $g_{0,n}(x; a_0, \dots, a_n; y; b_0) = (y - b_0)^n$
2. $D_x g_{m,n}(x; a_0, \dots, a_n; y; b_0, \dots, b_m) = m g_{m-1,n}(x; a_0, \dots, a_n; y; b_1, \dots, b_m)$ and $D_y g_{m,n}(x; a_0, \dots, a_n; y; b_0, \dots, b_m) = n g_{m,n-1}(x; a_1, \dots, a_n; y; b_0, \dots, b_m)$ with initial conditions $g_{m,n}(a_0; a_0, \dots, a_n; b_0; b_0, \dots, b_m) = \delta_{0n}\delta_{0m}$
3. For any real ζ and ν, $g_{m,n}(\zeta x; \zeta a_0, \dots, \zeta a_n; \nu y; \nu b_0, \dots, \nu b_m) =$ $\zeta^m \nu^n g_{m,n}(x; a_0, \dots, a_n; y; b_0, \dots, b_m)$
4. For any real ζ and ν, $g_{m,n}(x + \zeta; a_0 + \zeta, \dots, a_n + \zeta; y + \nu; b_0 + \nu, \dots, b_m + \nu) =$ $g_{m,n}(x; a_0, \dots, a_n; y; b_0, \dots, b_m)$
5. $g_{m,n}(x; a_0, \dots, a_n; y; b_0, \dots, b_m) = m \int_{a_0}^{x} g_{m-1,n}(x; a_1, \dots, a_n; y; b_0, \dots, b_m) +$ $n \int_{b_0}^{y} g_{m,n-1}(x; a_0, \dots, a_n; y; b_1, \dots, b_m) -$ $mn \int_{b_0}^{y} \int_{a_0}^{x} g_{m-1,n-1}(x; a_1, \dots, a_n; y; b_1, \dots, b_m)$

Proofs of these facts are straightforward: formulae 3 and 4 can be proven by induction.

Example 5.

$$\begin{aligned}
g_{0,0}(x;a_0;y;b_0) &= 1\\
g_{1,0}(x;a_0;y;b_0,b_1) &= x-a_0\\
g_{0,1}(x;a_0,a_1;y;b_0) &= y-b_0\\
g_{1,1}(x;a_0,a_1;y;b_0,b_1) &= b_0a_1+a_0b_1-a_0b_0-b_1x-a_1y+xy\\
g_{2,1}(x;a_0,a_1;y;b_0,b_1,b_2) &= 2a_0^2b_1-b_0a_1^2-2a_0b_1a_1-a_0^2b_2+2a_0b_0a_1-a_0^2b_0+2xb_1a_1\\
&\quad +2xa_0b_2-2xa_0b_1-b_2x^2+a_1^2y-2a_1xy+x^2y
\end{aligned}$$

In particular, the linear recursion becomes

$$x^my^n = \sum_{i=0}^{m}\sum_{j=0}^{n}\binom{m}{i}\binom{n}{j}a_j^{m-i}b_i^{n-j}g_{i,j}(x;a_0,\dots,a_j;y;b_0,\dots,b_i). \tag{4.7}$$

and the Appell relation

$$e^{(x+y)t} = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\frac{t^{i+j}e^{(a_j+b_i)t}}{i!j!}g_{i,j}(x;a_0,\dots,a_j;y;b_0,\dots,b_i). \tag{4.8}$$

Example 6.

$$\begin{aligned}
&g_{m,n}(x;s,s+c,\dots,s+nc;y;t,t+d,\dots,t+md)\\
=\ &(x-s-nc)^{m-1}(y-t-md)^{n-1}((x-s-nc)(y-t-md)-mdnc)
\end{aligned}$$

Immediately from this, we have

$$\begin{aligned}
g_{m,n}(x;s,s,\dots,s;y;t,t,\dots,t) &= (x-s)^m(y-t)^n\\
g_{m,n}(x;0,c,\dots,nc;y;0,d,\dots,md) &= (x-nc)^{m-1}(y-md)^{n-1}(xy-mdx-ncy)
\end{aligned}$$

Furthermore, by comparing equation 4.7 with equation 4.1, we can find the relationship between the two, which is,

Theorem 37. $PF_{n,m}(x;x_0,\dots,x_n;y;y_0,\dots,y_m) = g_{m,n}(x;x-x_0,\dots,x-x_n;y;y-y_0,\dots,y-y_m) = g_{m,n}(0;-x_0,\dots,-x_n;0;-y_0,\dots,-y_m) = (-1)^{m+n}g_{m,n}(0;x_0,\dots,x_n;0;y_0,\dots,y_m)$

In particular, when X and Y are linear progressions, we get, by eqn 4.9,

Corollary 38. $PF_{n,m}(s,s+c,\dots,s+nc;t,t+d,\dots,t+md) = (-1)^{m+n}g_{m,n}(0;s,s+c,\dots,s+nc;0;t,t+d,\dots,t+md) = (-1)^{m+n+1}(-s-nc)^{m-1}(-t-md)^{n-1}(-st-msd-nct) = (s+nc)^{m-1}(t+md)^{n-1}(st+msd+nct)$

Furthermore, equation 4.1 demonstrates that $(s+nc)^{t+md}(t+md)^{s+nc}$ is equal to

$$\sum_{i=0}^{m}\sum_{j=0}^{n}\binom{m}{i}(nc-jc)^{m-i}\binom{n}{j}.$$

In particular, when $s=c=t=d=1$, we have $x_i=y_i=i$, and we are enumerating the regular (p,q)–parking functions, which gives, by eqn 4.9,

$$PF_{p,q}(1,\dots,q+1;1,\dots,p+1) \quad = \quad (p+q+1)(p+1)^{q-1}(q+1)^{p-1},$$

which agrees with the result in [7]. And eqn 4.1 becomes

$$(q+1)^p(p+1)^q = \sum_{i=0}^{p}\sum_{j=0}^{q}\binom{p}{i}(q-j)^{p-i}\binom{q}{j}(p-i)^{q-j}(i+j+1)(i+1)^{j-1}(j+1)^{i-1}$$

Since now the number of $(\mathbf{X},\mathbf{Y})$–parking function is related to the Goncarov polynomials, any formula that is satisfied by the latter is also true to the former, i.e.,

Corollary 39. $PF_{n,m}(su_1,\dots,su_n;tv_1,\dots,tv_m) = s^nt^mPF_{n,m}(u_1,\dots,u_n;v_1,\dots,v_m)$

Corollary 40. Let D be the $(m+1)(n+1)\times(m+1)(n+1)$ matrix, whose $(i_1(n+1)+j_1, i_2(n+1)+j_2)$–th entry, with $0\le i_1,i_2\le m$ and $0\le j_1,j_2\le n$, is:

$$\begin{cases} \frac{a_{j_1}^{i_2-i_1}b_{i_1}^{j_2-j_1}}{(i_2-i_1)!(j_2-j_1)!} & \text{if } i_2\ge i_1 \text{ and } j_2\ge j_1, \\ 0 & \text{otherwise.}\end{cases}$$

Then $PF_{n,m}(a_0, \ldots, a_n; b_0, \ldots, b_m) = m!n!|D|$.

Example 7. $g_{m,0}(x; a_0; y; b_0, \ldots, b_m) = (x - a_0)^m$ indicates that $PF_{0,m}(a_0; b_0, \ldots, b_m) = a_0^m$, which says that, in the $(\mathbf{X}, \mathbf{Y})$–parking functions, there is no choice for the entry in the position of x, while the m entries in the position of y can vary from 1 to a. $PF_{1,1}(0; a_0, a_1; 0; b_0, b_1) = g_{1,1}(0; a_0, a_1; 0; b_0, b_1) = b_0a_1 + a_0b_1 - a_0b_0$ shows that the $(\mathbf{X}, \mathbf{Y})$–parking functions are the pairs (a, b), where $a \le a_1$ and $b \le b_1$ except those cases when both $a > a_0$ and $b > b_0$.

In both cases, the number derived from the Goncarov polynomials agree with the ones from the definition.

D. A New Perspective of (p, q)-Parking Functions

In this section we present two new interpretations of (p, q)–parking functions. Let $P^\top := (a_0, \ldots, a_{q-1})$ be a lattice path from $(0, 0)$ to (p, q) such that the right-most point on the i^{th} horizontal line is $(a_i, i), i < q$; let $Q^\perp := (b_0, \ldots, b_{p-1})$ be a lattice path from $(0, 0)$ to (p, q) such that the top-most point on the j^{th} vertical line is $(j, b_j), j < p$.

Lemma 41. The associated increasing (p, q)–sequence (σ, τ) of each $\pi \in S_{p+q}$ defines two paths from $(0, 0)$ to (p, q), and $\sigma^\top$ and $\tau^\perp$ are identical. Conversely, any path from $(0, 0)$ to (p, q) can be written as $\sigma^\top$ and $\tau^\perp$ (for some sequences σ and τ) and (σ, τ) is an associated increasing (p, q)–sequence of some permutation in S_{p+q}.

Proof. Let $\pi \in S_{p+q}$, and (σ, τ) be its associated increasing (p, q)–sequence. We can assume, without the lost of generality, that $\pi = (\phi_0, \ldots, \phi_{p-1}, \psi_0, \ldots, \psi_{q-1})$, where ϕ and ψ are increasing sequences, and (σ, τ) still is its associated increasing (p, q)–sequence.

By the definition of σ, (i, j) is on $\sigma^\top$ iff $\sigma_{j-1} \le i \le \sigma_j$. This assures us that there are at least i entries in ψ that are less than ϕ_j, and at most i entries that are less than ϕ_{j-1}. Since the sequences start with index 0, this means $\psi_{i-1} < \phi_j$ and $\psi_i > \phi_{j-1}$, i.e., $\tau_{i-1} \le j \le \tau_i$. Using the definition of τ, we know (i, j) is on $\tau^\perp$ too. Since both paths have $p + q$ points, they are identical.

On the other hand, for every pair of sequences (σ, τ), if $\sigma^\top = \tau^\perp$, we can use the algorithm that follows to construct a permutation:

1. Let an integer n be 1; ϕ, ψ be two empty sequences; and we walk the path starting from $(0, 0)$;
2. At the current lattice point, if the line segment after it in the path is moving up, attach n to the end of ϕ, otherwise, attach it to the end of ψ;
3. Increase the value of n by 1, and move to the next point on the path;
4. Return to step 2, until we reach the point (p, q);
5. (ϕ, ψ) is the desired permutation.

In the algorithm, when each lattice point is visited and a number inserted, the number of vertical (horizontal) segments is the number of integers showed up in ϕ (ψ), and thus the values of τ and σ. $\square$

The permutation found in the proof above is unique when both ϕ and ψ are increasing. It is clear from the algorithm that $\phi_i = \sigma_i + i + 1$ and $\psi_j = \tau_j + j + 1$.

Corollary 42. The number of increasing (p, q)–parking function is the number of pairs of non-crossing paths from $(0, 0)$ to (p, q).

Proof. Let (σ, τ) be an increasing (p, q)–parking function, π be one of its certificates, and P be the unique path defined in the proof above. By the definition of (p, q)–parking function, we can see that $\sigma^\top$ is above P while $\tau^\perp$ is below.

On the other hand, if we have a pair of non-crossing paths P_1, P_2 from $(0,0)$ to (p,q), where P_1 is above P_2, let σ, τ be two sequences such that $\sigma^\top = P_1$ and $\tau^\perp = P_2$, and let π be permutation defined in the algorithm above based on P_2. It is then clear that (σ, τ) is an increasing (p,q)–parking function with certificate π. $\square$

Example 8. Applying the algorithm in Lemma 41 to the lattice path illustrated below in Figure 24 gives the permutation $(2,5,1,3,4)$.

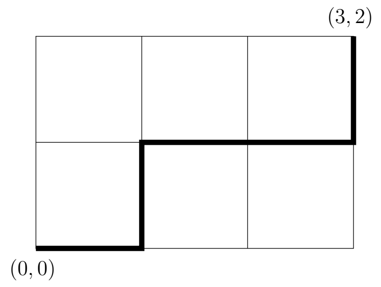


Fig. 24. A lattice path from $(0,0)$ to $(3,2)$.

Similarly, an $(\mathbf{X}, \mathbf{Y})$–parking function can be viewed as pairs of paths too. We can draw dotted lines within an $x-1$ by $y-1$ rectangle through the lattice points. Redraw the vertical lines with x–coordinates $\{x_0 - 1, \ldots, x_n - 1 = x - 1\}$, and the horizontal lines with y–coordinates $\{y_0 - 1, \ldots, y_m - 1 = y - 1\}$ as solid lines. There is a one-to-one mapping that maps an $(\mathbf{X}, \mathbf{Y})$–parking function to a pair of non-crossing lattice paths within the rectangle such that there is a path along the solid lines that seperates the two, which we call the *firewall.* The exact proof is omitted.

In the case when $X_i = Y_i = i + 1$, all those lines are solid, and thus we get the regular (p,q)–parking functions. If $X_0 = Y_0 = 0$, we cannot build a firewall at all, and therefore the total number of such $(\mathbf{X}, \mathbf{Y})$–parking function is 0, which agree with the initial condition eq. 4.6 and Theorem 37.

We can also derive an interesting identity for the *Narayama numbers* $N(j,k) = \frac{1}{j+k+1}\binom{j+k+1}{j}\binom{j+k+1}{k}$.

Proposition 43. For nonnegative integers p and q we have

$$\binom{p+q}{p}^2 = \sum_{j=0}^{p}\sum_{k=0}^{q}\binom{p-j+q-k}{p-j-1}\binom{p-j+q-k}{q-k-1}N(j,k)$$

Proof. Let $L((0,0),(p,q))$ denote the rectangular lattice whose diagonally opposite corners are $(0,0)$ and (p,q). Let P be the set of all pairs of paths in $L((0,0),(p,q))$ beginning at $(0,0)$ and ending at (p,q) which move either in northward (from (i,j) to $(i,j+1)$) or eastward (from (i,j) to $(i+1,j)$) steps. It is elementary that $|P| = \binom{p+q}{p}^2$. We can also enumerate $|P|$ in the following way. Given any pair of paths in L, let (j,k) be the point nearest the origin at which the two paths cross (i.e. one path follows $(j,k-1)$ to (j,k) to $(j,k+1)$ and the other follows $(j-1,k)$ to (j,k) to $(j+1,k)$). The two paths are then noncrossing in $L((0,0),(j,k))$ and are cross and intersect arbitrarily in $L((j,k),(p,q))$. The possibilities in $L((0,0),(j,k))$ are enumerated by $N(j,k)$. The possibilities in $L((j,k),(p,q))$ are enumerated by $\binom{p-j+q-k}{p-j-1}\binom{p-j+q-k}{q-k-1}$ (the $p-j-1$ and $q-k-1$ are because one step of both paths in $L((j,k),(p,q))$ is already determined). $\square$

CHAPTER V

CONCLUSION

Chapter I was an overview of the current state of the theory of generalized parking functions. Chapter II has served to strengthen the relationship between generalized parking functions and various matroid structures, most particularly spanning forests. Chapter III placed the new generalization in the context of chip-firing games. Finally, Chapter IV emphasizes the importance of (p, q)-parking functions in even more generalized settings.

Much work remains to be done to more fully understand choice functions. It is clear that if ζ is a choice function, then algorithm A of Chapter II is a "ζ-first" search. However basic questions about these functions, such as how many exist for a given graph, seem difficult. Answering such questions would likely provide important insights into the connectivity of a graph.

Another interesting question unaddressed in this dissertation is a whether a group structure exists on the set of multiparking functions of a given graph. Biggs [2] has demonstrated that there is an elegant group structure on the critical configurations of a graph. Since G-parking functions are in bijection to critical configurations, this group structure exists on them as well. However, it is not so easy to describe the critical group in terms of G-parking functions. Whether this group structure can be extended to multiparking functions (of either the kind described in Chapter II or Chapter III) remains to be investigated.

REFERENCES

[1] N. Biggs, *Algebraic Graph Theory.* 2^{nd} ed (Cambridge University Press, London, 1993).

[2] N. Biggs, Chip firing and the critical group of a graph, CDAM Research Report Series, 1995.

[3] N. Biggs, The tutte polynomial as a growth function, J. Alg. Combin. 10 (1999) 115-133.

[4] F. Chung and R. Ellis, A chip-firing game and dirichlet eigenvalues, Discrete Math. 257 (2002) 341-355.

[5] D. Chebikin and P. Pylyavskyy, A family of bijections between G-parking functions and spanning trees, J. of Combin. Theory Ser. A 110 (2005) 31-41.

[6] R. Cori and Y. Le Borgne, The sand-pile model and tutte polynomials, Adv. in Appl. Math. 30 (2003) 44-52.

[7] R. Cori and D. Poulalhon, Enumeration of (p,q)-parking functions, Discrete Math. 256 (2002) 609-623.

[8] R. Cori and D. Rossin, On the sandpile group of dual graphs, Euro. J. Comb., 21 (2000) 447-459.

[9] D. Dhar, Self-organized critical state of the sandpile automaton models, Phys. Rev. Lett., 64 (1990) no. 14, 1613-1616.

[10] D. Foata and J. Riordan, Mappings of acyclic and parking functions, Aequationes Math. 10 (1974) 10-22.

[11] A. Gabrielov, Abelian avalanches and tutte polynomials, Physica A 195 (1993) 253-274.

[12] I. Gessel, A noncommutative generalization and q-analog of the lagrange inversion formula, Transactions of the American Mathematical Society 257 (1980) 455-482.

[13] I. Gessel and B. Sagan, The tutte polynomial of a graph, depth-first search, and simplicial complex partitions, Electron. J. Combin. 3 (no. 2) R9.

[14] I. Gessel and D. Wang, Depth-first search as a combinatorial correspondence, J. Combin. Theory Ser. A, 26 (1979) 308-313.

[15] A. Konheim and B. Weiss, An occupancy discipline and applications, SIAM J. Appl. Math. 14 (1966) 1266-1274.

[16] G. Kreweras, Une famille de polynômes ayant plusieurs propriétés énumeratives, Period. Math. Hungar. 11 (1980) 309-320.

[17] J. Kung, X. Sun and C.H. Yan, Goncarov-type polynomials and applications in combinatorics, preprint (accessed 25 March 2007). http://www.math.tamu.edu/∼cyan/Files/DGP.pdf

[18] J. Kung and C.H. Yan, Gončarov polynomials and parking functions, J. Combin. Theory Ser. A, 102 (2003) 16-37.

[19] J. Kung and C.H. Yan, Exact formula for moments of sums of classical parking functions, Adv. in Appl. Math. 31 (2003) 215-241.

[20] J. Kung and C.H. Yan, Expected sums of moments of general parking functions, Ann. Comb. 7 (2003) 16-37.

[21] C.M. López, Chip firing and tutte polynomials, Ann. Comb. 3 (1997) 253-259.

[22] C.L. Mallows and J. Riordan, The inversion enumerator for labelled trees, Bull. Amer. Math. Soc., 74 (1968), 92-94.

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

[24] A. Postnikov and B. Shapiro, Trees, parking functions, syzygies, and deformations of monomial ideals, Trans. Amer. Math. Soc. 356 (2004) 3109-3142.

[25] R. Speicher, *Combinatorics of Free Probability Theory* (accessed 1 March 2007), http://www.mast.queensu.ca/∼speicher/papers/lectures-IHP.pdf

[26] J. Spencer, Enumerating graphs and brownian motion, Comm. Pure Appl. Math., vol. L (1997) 291-294.

[27] R. Stanley, Parking functions and noncrossing partitions, Electron. J. Combin., 4 R20 (1997).

[28] R. Stanley, Hyperplane arrangements, parking functions, and tree inversions, in: B. Sagan and R. Stanley, eds., *Mathematical Essays in Honor of Gian-Carlo Rota* (Birkhäuser, Boston, 1998) 359-375.

[29] W. Tutte, A contribution to the theory of chromatic polynomials, Canad. J. Math., 6 (1953) 80-91.

[30] C.H. Yan, Generalized tree inversions and k-parking functions, J. Combin. Theory Ser. A, 79 (1997) 268-280.

[31] C.H. Yan, Generalized parking functions, tree inversions, and multicolored graphs, Adv. in Appl. Math., 27 (2001) 641-670.

VITA

Dimitrije Nenad Kostić received his Bachelor of Arts degree in mathematics from Lawrence University in May of 2001. He entered the graduate program in mathematics at Texas A&M University later in 2001 and received a Master of Science degree in December of 2003. His research interests include algebraic graph theory, algebraic combinatorics, and combinatorial number theory. Chapters II and III of this dissertation have been accepted for publication in Advances in Applied Mathematics and the Annals of Combinatorics, respectively, and Chapter IV has been submitted. Mr. Kostić can be contacted at the Department of Mathematics, Texas A&M University, College Station, TX 77843 and at dkostic@math.tamu.edu until August of 2007. After that date, he can be reached at the Department of Mathematics and Computer Science, Drew University, Madison, NJ 09234 and at dkostic@drew.edu.