Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090601

Combinatorial Analysis of Diagonal, Box, and Greater-Than Polynomials as Packing Functions

Jose Torres-Jimenez[1,2,*], *Nelson Rangel-Valdez*[3], *Raghu N. Kacker*[2] *and James F. Lawrence*[4]

[1] CINVESTAV-Tamaulipas, Information Technology Laboratory, Km. 5.5 Carretera Victoria-Soto La Marina, 87130 Victoria Tamps., Mexico
[2] National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
[3] Universidad Politécnica de Victoria, Av. Nuevas Tecnologías 5902, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., México
[4] George Mason University, Fairfax, VA 22030, USA

Received: 2 Feb. 2015, Revised: 2 Apr. 2015, Accepted: 3 Apr. 2015
Published online: 1 Nov. 2015

Abstract: A packing function is a bijection between a subset $V \subseteq N^m$ and N, where N denotes the set of non negative integers N. Packing functions have several applications, e.g. in partitioning schemes and in text compression. Two categories of packing functions are Diagonal Polynomials and Box Polynomials. The bijections for diagonal ad box polynomials have mostly been studied for small values of m. In addition to presenting bijections for box and diagonal polynomials for any value of m, we present a bijection using what we call Greater-Than Polynomial between restricted $m-$dimensional vectors over N^m and N. We give details of two interesting applications of packing functions: (a) the application of greater-than polynomials for the manipulation of Covering Arrays that are used in combinatorial interaction testing; and (b) the relationship between grater-than and diagonal polynomials with a special case of Diophantine equations. A comparison of the bijections for box, diagonal and greater-than polynomials are presented and we conclude that the bijection for box polynomials is efficient because its direct and inverse methods have orders of $O(n^2 \cdot m)$ and $O(n^3 \cdot m)$ (measured in terms of bit operations, where n is the number of bits of an integer involved in the methods)

Keywords: Diagonal Polynomials, Box Polynomials, Greater-Than Polynomials

1 Introduction

Denoting by N the set of non negative integers, let $f : V \to N$, where $V \subseteq N^m$ be a function. Then f is a *Packing Function* (PF) whenever f is a bijection. The function f is a *Storing Function* (SF) if it is an injection. PFs and SFs have been used to address several combinatorial problems having applications in computer science. Two examples of these problems are the compression of data [11] and the generation of extensible arrays [25, 3].

An efficient PF would be one that, when implemented in a computer system, performs the bijection using minimal resources (i.e. requiring less computer time and space). The most studied types of PFs are the *Diagonal Polynomial* (DP) and the *Box Polynomial* (BP). The research done in this area involves the study of direct methods (those that map a vector to an integer) and inverse methods (the ones that transform an integer into the corresponding vector); most of the work found in the literature involves vectors of small number of dimensions (commonly two or three dimensions [25, 3, 26]). The work is rather theoretical and seldom involves a computational complexity analysis.

In addition to presenting bijections for DP and BP for any number of dimensions, we present a new PF derived from the combinatorial numeration system reported in [10]. The new PF is called *Greater-Than Polynomial* (GTP) due to the fact that the vectors satisfy the property that a vector element is strictly greater than the previous element. After presenting algorithms for computing the direct and inverse methods for DP, BP, and GTP, we analyze them in terms of their computational complexity. The analysis compares the number of bit operations required for each PF to map an integer number to a vector of N^m and vice versa. Additional contributions of the

* Corresponding author e-mail: jtj@cinvestav.mx

paper are as follows: (a) the application of GTP for the manipulation of Covering Arrays (CAs), that are used in combinatorial interaction testing; and (b) an interesting bijection between GTP and DP used to generate random solutions of Linear Diophantine Equations with Unit coefficients (LDEU).

The remainder of the document is organized as follows: Section 2 presents some applications of PFs. Sections 3, 4, and 5 present the description of GTP, DP, and BP; for each kind of polynomial, a formal definition, some related work, and their inverse and direct methods are presented. Section 3 additionally shows the application of GTP for manipulating Covering Arrays (CAs). Section 4 also presents the application of GTP and DP to generate random solutions of a LDEU. Section 6 summarizes the computational complexity of the algorithms proposed in this paper. Finally, Section 7 presents some conclusions.

2 Applications and Related Work

This section describes several applications of packing functions.

2.1 Applications of Packing Functions

There are many ways of representing an integer uniquely [16, 10]. Some representations rely on vectors and are useful in tasks related to digital information processing, representation of a problem solution in meta heuristics, and memory organization schemes. Of particular interest are the integer vectorial representations, these representations include the polynomials DP, GTP, and BP, which can uniquely represent each non negative integer as a vector over N^m. A bijection of N^m to N can be used to organize elements of an $m-$dimensional array into a $1-$dimensional array. The task is simple: just use a bijection like DP, GTP, or BP to transform an element of the $m-$dimensional array into an integer value that represents the position in the $1-$dimensional array. The work presented by Rosenberg [24] shows an addressing scheme of data graphs. A data graph is a representation of data structures. The data graphs are mapped into an address space using an approach that transforms vectors into integers. The work mainly discusses the addressing of binary trees.

Rosenberg [25] and Brodnik et al. [3] present strategies for the management of extensible arrays, i.e. arrays where rows and/or columns can be appended dynamically. Usually, arrays are stored by rows or columns, for example, after the first row (or column) has been stored in memory, the second row (or column) is stored next. This organization scheme makes it more expensive to add new rows or columns to the current array, because adding a row (or a column) just requires the addition of it at the end of the array, but the addition of a column (or a row) requires a reorganization of the stored array. The works presented in [25, 3] deal with the problem of allocating storage for extensible arrays. In these approaches DP and BP can be used as possible organization schemes. Similar organization schemes based on DP and BP have been used in [21, 12] as storage schemes for parallel array access and as a matrix storage format for symmetric and triangular matrices.

Another application of packing functions based on DP is found in [13]; there, DP support the consecutive retrieval organization of files and their queries, i.e., a scheme for the storage of the file on a linear storage medium such that for each permissible query, there exists a block of consecutive locations with exactly those records that are pertinent to the query. Fraenkel proposed in [11] a scheme for combinatorial compression and partitioning for large dictionaries. More recently, DPs have been used in [26] as a mechanism for accountability in web computing projects. Finally, the works done by Charlier, et al. [6] and Lecomte et al. [15], study arithmetic properties between integer values and some numeration systems. They present abstract numeration systems which can be used to describe DPs and GTPs.

The remaining sections of this paper describe and compare the direct and inverse methods for GTP, DP, and BP; also we discuss two applications that can make use of the GTP and DP packing functions.

3 Greater-Than Polynomials (GTP)

In [10, 16] is presented a numeration system that associates a vector $V = (v_1,...,v_m)$ with a natural number α where the entries satisfy $v_i < v_{i+1}$, $v_1 \geq 0$. The values v_i are used in binomial coefficients and summed up as in $\alpha = \sum_{i=1}^{m} \binom{v_i}{i}$ to uniquely define an integer $\alpha \in N$. This numeration system will be called Greater-Than Polynomials (GTP), because the expansion of the binomial coefficients yields a polynomial in $(v_1,...,v_m)$ and the components of the vector V are monotonically increasing.

The direct and inverse methods for GTP to perform the bijection are presented in the next subsections. The analysis of the computational cost of using such methods is also presented. In particular the GTP inverse method proposed in this paper is compared with the method reported in [11]. **Table 1** shows examples of vectors that belong to GTP. Each cell in this matrix is a vector $V = (v_1, v_2)$ (the order is (row, column)). The value appearing in each cell is the integer corresponding to the vector defined by that cell. Note that only the cells where the column is greater than the row exist.

Table 1: Enumeration of the vectors defined by a Matrix of size 5×5 using GTP.

	0	1	2	3	4
0		0	1	3	6
1			2	4	7
2				5	8
3					9
4					

3.1 Direct Method for GTP

The algorithm to transform a vector V, where $0 \le v_1 < v_2 < \ldots < v_{m-1} < v_m$, to an integer α can be easily derived from its definition [16, 10, 29]. The pseudocode is presented in **Algorithm 1**.

Algorithm 1 DirectGTP(V)

```
1: $\alpha := 0$
2: for $i := 1$ to $m$ do
3:     $\alpha := \alpha + Binomial(v_i, i)$
4: end for
5: return $\alpha$
```

The method presented in **Algorithm 1** requires the calculation of m binomial coefficients. The computational cost of computing all the binomial coefficients is proportional to m^2 multiplications and divisions (see **Algorithm 2**). Given that one multiplication requires $O(n^2)$ bit operations, where $n = \lceil \log_2(\alpha) \rceil$ (i.e. the bits of the integer α) the direct method for GTP has a temporal complexity proportional to $O(n^2 \cdot m^2)$. The spatial complexity is $O(m)$.

Algorithm 2 Binomial(k,r)

```
1: if $k < r$ then
2:     return 0
3: end if
4: $b := 1$
5: for $i := 1$ to $r$ do
6:     $b := (b * (k - i + 1))/i$
7: end for
8: return $b$
```

3.2 Inverse Method for GTP

GTP has been used by Charlier, et al. [6] to enumerate different words that can be derived from a given language $\mathscr{B}_l$. There, an algorithm to perform the inverse of GTP is described; this algorithm was first presented by Fraenkel [11]. The decoding process described for this algorithm computes the combinatorial representation of an integer α with respect to a value m (the size of the vector) by iteratively computing the values v_i as:

$\binom{v_m}{m} \le \alpha < \binom{v_m+1}{m}$;

$\binom{v_{m-1}}{m-1} \le \alpha - \binom{v_m}{m} < \binom{v_{m-1}+1}{m-1}$;

$\binom{v_{m-2}}{m-2} \le \alpha - \binom{v_m}{m} - \binom{v_{m-1}}{m-1} < \binom{v_{m-2}+1}{m-2}$;

and so on.

To compute v_i in $\binom{v_i}{i}$, for $i = 1, \ldots, m$, the algorithm proposed by Fraenkel [11] calculates in each of its iterations the value $\lceil \frac{\alpha}{e} \sqrt[i]{\sqrt{2\pi i}\alpha} \rceil$ as an approximation of v_i. Then, in a maximum of m steps, each involving a multiplication and a division, it computes the exact value of v_i. The number α is updated to $\alpha - \binom{v_i}{i}$ in order to compute the next value v_{m-1}. The process is repeated until the value v_1 is obtained. The number of bit operations performed by this algorithm is $O(n^3 \cdot m^2)$, taking into account that the m^{th} root is computed in $O(n^3 \cdot m)$ bit operations.

In this paper we propose an alternative method to the one presented in [11]. Our approach uses for initial solution the integer root $a_i = \sqrt[i]{i! \cdot \alpha}$. The calculation of a_i avoids the use of the constant e and the division operation. Also, the value v_i is at most $a_i + m$. An additional advantage of our approach is that it can be implemented using only additions, multiplications, and divisions using the integer root algorithm presented in [30]. The precision issues from using floating point operations are also eliminated in this approach. **Algorithm 3** is the pseudocode of the inverse of **Algorithm 1**. This pseudocode details each of the elements (with the only exception of the integer root operation) required for the computation of V, where $m = |V|$, from an integer α.

The pseudocode of **Algorithm 3** is characterized by a main loop and three main blocks inside it. These blocks involve calculation of: (a) the initial value a_i in line 3; (b) the binomial coefficient in line 5; and (c) the value v_i in the loop starting in line 6 (in this loop **Algorithm 4** is used, it requires one multiplication and one division to compute $\binom{v_i+1}{i}$).

Assuming that the value a_i is calculated by computing the i^{th} integer root of $i! \cdot \alpha$ using the algorithm FηRA [30], the number of bit operations is bounded by $O(n^3 \cdot m)$ (m is the maximum value of i). The binomial coefficient $\binom{a_i}{i}$ requires $O(n^2 \cdot m)$ bit operations, as described previously in **Algorithm 2**. Finally, the loop in line 6 is repeated at most m times and each time it performs a constant number of multiplications and divisions. The value v_i can be found in at most $O(n^2 \cdot m)$ bit operations. Given that the cost $O(n^3 \cdot m)$ of computing $\lfloor \sqrt[i]{i! \cdot \alpha} \rfloor$ dominates all the others in the main loop of **Algorithm 3**, the computational complexity is bounded by $O(n^3 \cdot m^2)$.

Algorithm 3 InverseGTP (α, m, V)

```
1:  $f := m!$
2:  **for** $i := m$ **downto** 1 **do**
3:      $a_i := \sqrt[i]{f * \alpha}$
4:      $v_i := a_i$
5:      $\Delta := Binomial(v_i, i)$
6:      **while** $NextBinomial(\Delta, v_i, i) \leq \alpha$ **do**
7:          $\Delta := NextBinomial(\Delta, v_i, i)$
8:          $v_i := v_i + 1$
9:      **end while**
10:     $\alpha := \alpha - \Delta$
11:     $f := f/i$
12: **end for**
13: **return** $V$
```

Algorithm 4 NextBinomial(p, k, r) Input $p = \binom{k}{r}$

```
1: **if** $k + 1 < r$ **then**
2:     **return** 0
3: **end if**
4: **if** $k + 1 = r$ **then**
5:     **return** 1
6: **end if**
7: $p := p * (k+1)/(k+1-r)$ { 2 operations for $\binom{k+1}{r}$ }
8: **return** $b$ { Output: the value $\binom{k+1}{r}$ }
```

3.3 GTP and Covering Arrays

Empirical studies in software testing have shown that combinatorial interaction testing is a useful approach to guarantee the functionality of software components [14, 2]. The mathematical object that supports combinatorial interaction testing is the Covering Array (CA). A *CA*, denoted by $CA(N;t,k,v)$, is an $N \times k$ array, where every entry of the array takes values from a set of symbols of size v, such that every $N \times t$ sub-array contains all the possible v^t t-tuples at least once. The test cases are represented by the rows, the parameters are represented by the columns, the parameter values are taken from the set $\{0, 1 \ldots, v-1\}$ which is called alphabet, and t is the strength or combinatorial interaction degree between parameters covered by the CA. **Figure 1** shows an example of a $CA(9;2,4,3)$.

0	1	2	2	0	1	1	2	0
0	1	2	1	2	0	2	0	1
0	1	2	0	1	2	0	1	2
0	0	0	1	1	1	2	2	2

Fig. 1: Transposed matrix of a $CA(9;2,4,3)$.

The manipulation of CAs requires: (a) the computation of all the possibles subsets of t columns taken from k columns (this operation is required to verify that all the possible v^t t-tuples appear in the CA at least once [1]); and (b) the generation of a random subset of t columns from k columns (this operation is required in meta heuristic algorithms to construct CAs [31]). The generation of all the possible subsets of t columns taken from k columns can be done by calling the inverse method for GTP with the numbers in the range $0,..,\binom{k}{t} - 1$. Another option is presented in **Algorithm 5** that builds all the vectors in a GTP. In this algorithm the vector V stores a GTP vector of size t (in which the maximum valid value of v_t is $k-1$). In **Algorithm 5** ω defines the starting index for copying the previous element when it occurs that the last element v_t has reached the maximum value. The order of **Algorithm 5** is $O(\binom{k}{t})$.

Algorithm 5 AllSubsets(k, t, V)

```
1:  **for** $i := 1$ **to** $t$ **do**
2:      $V_i := i - 1$
3:  **end for**
4:  **if** $k = t$ **then**
5:      $\omega := 1$
6:  **else**
7:      $\omega := t$
8:  **end if**
9:  **while** $V_t \neq k$ **or** $\omega \neq 1$ **do**
10:     {use V to test coverage of CA}
11:     $V_t := V_t + 1$
12:     **if** $V_t = k$ **and** $\omega \neq 1$ **then**
13:         $V_{\omega-1} := V_{\omega-1} + 1$
14:         **for** $i := \omega$ **to** $t$ **do**
15:             $V_i := V_{i-1} + 1$
16:         **end for**
17:         **if** $V_{\omega-1} = k - t + \omega - 2$ **then**
18:             $\omega := \omega - 1$
19:         **else**
20:             $\omega := t$
21:         **end if**
22:     **end if**
23: **end while**
```

The generation of a random subset of columns can be computed using the inverse method for GTPs, first we compute a random number in the range $0,\ldots,\binom{k}{t} - 1$ and then use the inverse method to construct the corresponding vector (a random subset of t columns taken from k columns).

The following section presents DP, a widely studied PF. The methods for the bijection of this PF are described and the use of GTP and DP to generate random solutions of an LDEU is presented.

4 Diagonal Polynomials (DP)

The idea of *Diagonal Polynomials* (DPs) appears in the 1821 writings of Cauchy about double summations. Later, Cantor [4] uses DP to show the equivalence between

Table 2: Enumeration of the vectors defined by a Matrix of size 5×5 using DP.

	0	1	2	3	4
0	0	1	3	6	10
1	2	4	7	11	
2	5	8	12		
3	9	13			
4	14				

vectors of two dimensions and N. The *diagonal argument* was formally given in Cantor's inspiring paper [5], in trying to solve the *Continuum Problem*. The diagonal argument can be taken a step further through the definition of an *equivalence class* over the vector $W = (w_1, w_2)$. Its equivalence class is defined by $(w_1 + w_2)$, so that vectors are categorized in diagonals, see **Table 2** (the order is (row, column)). The direct mapping of the vector $W = (w_1, w_2)$ to $\alpha \in N$ is: $\alpha = w_1 + \binom{1+w_1+w_2}{2}$. The inverse method (mapping $W = (w_1, w_2)$ to $\alpha \in N$), can be performed in two steps: (1) calculation of the equivalence class d through the expression $d = \lfloor \frac{\sqrt{1+8\alpha}-1}{2} \rfloor$; and (2) calculation of the values of w_1 and w_2 using $w_1 = \alpha - \frac{(d)(d+1)}{2}$ and $w_2 = d - w_1$. The simple method proposed by Cantor to map $2-$dimensional vectors into N can be generalized to higher dimensions. In the remainder of this section we discuss the direct and inverse DP methods.

4.1 Direct Method for DP

A wide variety of direct methods for computing the integer value of an $m-$dimensional vector are given in the literature [19, 20, 17, 18, 23, 22, 32, 9, 8]. Most of them are named as packing polynomials (or packing functions) because they are bijections between a vector in N^m and N. Some of these methods result from a natural generalization of Cantor's polynomials [28, 7]. For instance

$$\alpha = \sum_{i=1}^{m} \binom{i-1+\sum_{j=1}^{i} w_j}{i}$$

maps an $m-$dimensional vector $W = (w_1, \ldots, w_m)$ to $\alpha \in N$. **Algorithm 6** presents the pseudocode to compute the integer number α corresponding to the vector W. The algorithm is simple. It adds the values of the vector and in each iteration it computes the corresponding binomial coefficient. The space complexity of the **Algorithm 6** is $O(m)$, where m is the number of dimensions of W. The temporal complexity is bounded by $O(n^2 \cdot m^2)$; it is so since the cost to compute each binomial coefficient is $O(n^2 \cdot m)$ bit operations.

Algorithm 6 DirectDP(W)

```
1: $s := 0, \alpha := 0, m := |W|$
2: for $i := 1$ to $m$ do
3:     $s := s + w_i$
4:     $\alpha := \alpha + Binomial(i-1+s, i)$
5: end for
6: return $\alpha$
```

4.2 Inverse Method for DP

The bijection for DP has been widely studied [28, 7]. However, to the best of our knowledge, while the direct methods are commonly given, the inverse methods are not. This subsection presents a bijection between DP and GTP. Let W denote a DP vector and V a GTP vector, then $v_i = i - 1 + \sum_{j=1}^{i} w_j$ transforms W into V; and the transformation from a GTP vector V into a DP vector W is done by $w_i = v_i - v_{i-1} - 1$ (with $w_1 = v_1$), given that only additions and subtractions are used, the bijections can be computed in $O(n \cdot m)$ bit operations and they use $O(m)$ space (here n is the number of bits required for the maximum number being manipulated in the bijection). Hence, the inverse method for DP is that proposed for GTP using the bijection between DP and GTP; therefore, the computational complexity of the inverse method for DP is $O(n^3 \cdot m^2)$ bit operations.

4.3 GTP, DP, and LDEU

As a bonus, GTP and DP can be used in conjunction with a random number generator for generating random solutions for an LDEU. It has been shown [27] that the number of possible ways to arrange a set of l indistinguishable objects into r urns is equivalent to the problem of finding the number of distinct non negative integer valued vectors $(x_1, x_2, \ldots, x_r)$ such that the LDEU $v_i = i - 1 + \sum_{j=1}^{i} w_j$ holds. The number of non negative solutions for the previous LDEU is $\binom{l+r-1}{r-1}$, the solutions can be obtained by mapping each value between 0 and $\binom{l+r-1}{r-1} - 1$ to its corresponding $(r-1)-$dimensional vector V according to the definition of GTP. Then, the resulting $(r-1)-$dimensional vector is mapped to its corresponding vector W in DP. The entries of the vector W in DP correspond to $r-1$ variables in LDEU. The entry for the remaining variable is $l - \sum_{i=1}^{r-1} w_i$.

Table 3 shows an example of the bijection between GTP, DP, and solutions for a particular LDEU. The LDEU has $l = 4$ and $r = 3$. The GTP and DP vectors are of 2 dimensions. Column 1 shows the non negative integers corresponding to each different solution. Columns 2 and 3 show the GTP and DP vectors associated to that number, respectively. Column 4 presents the solution for the LDEU associated to that number (and derived from the DP vector).

Table 3: Solutions of the LDEU $x_1 + x_2 + x_3 = 4$ obtained through GTP and DP.

	GTP	DP	
No	W	V	LDEU Solution
0	$(0,1)$	$(0,0)$	$0+0+4$
1	$(0,2)$	$(0,1)$	$0+1+3$
2	$(1,2)$	$(1,0)$	$1+0+3$
3	$(0,3)$	$(0,2)$	$0+2+2$
4	$(1,3)$	$(1,1)$	$1+1+2$
5	$(2,3)$	$(2,0)$	$2+0+2$
6	$(0,4)$	$(0,3)$	$0+3+1$
7	$(1,4)$	$(1,2)$	$1+2+1$
8	$(2,4)$	$(2,1)$	$2+1+1$
9	$(3,4)$	$(3,0)$	$3+0+1$
10	$(0,5)$	$(0,4)$	$0+4+0$
11	$(1,5)$	$(1,3)$	$1+3+0$
12	$(2,5)$	$(2,2)$	$2+2+0$
13	$(3,5)$	$(3,1)$	$3+1+0$
14	$(4,5)$	$(4,0)$	$4+0+0$

5 Box Polynomials (BP)

This section presents another packing function, the *Box Polynomial* (or BP). This is the most efficient of the three packing functions discussed in this paper, according to the number of bit operations required to perform the bijection. The enumeration of vectors of a given dimension m can be done using box shells. This kind of enumeration is called box enumeration and it consists in enumerating the vectors according to the maximum value of components and the index for the first occurrence of the maximum value. **Table 4** shows an example of how vectors of two dimensions could be enumerated (the order is (row, column)). The box enumeration can also be described by BP.

Table 4: Enumeration of the vectors defined by a Matrix of size 5×5 using BP.

	0	1	2	3	4
0	0	3	7	13	21
1	1	2	8	14	22
2	4	5	6	15	23
3	9	10	11	12	24
4	16	17	18	19	20

While in the DP enumeration the class d is defined by the summation of the values of a vector, in the BP enumeration the class d is defined by two values: the maximum value of the vector components and the index of the first occurrence of the maximum value. An advantage of using BP as a PF over DP and GTP is that in BP the transformation between the vector space and the natural numbers requires powers of integers instead of binomial coefficients. BP enumeration schemes have been reported in the literature [28]; however, to the best of our knowledge, the algorithms for that bijection address only small values of m. The remainder of this section presents the direct and inverse methods for BP; also an analysis of the computational complexity is presented for each method.

5.1 Direct Method for BP

The method to convert a BP vector to a natural number requires the computation of two contributions: a contribution from the class to which the vector belongs and a contribution for the specific vector. The class is defined by the maximum value of the vector and the index of the first occurrence of the maximum value in the vector. Two vectors belong to the same class if they have the same schemata. A schemata is defined by the first occurrence of the maximum value in the vector (and in this way the class is defined by two values: the maximum value denoted by β and the index for the first occurrence of this value in the vector denoted by the Greek symbol ι). For instance a vector B with the schemata: $(b_1, b_2, b_3,\ 3,\ b_5, b_6, b_7, b_8)$ is characterized by the pair $\{\beta, \iota\} = \{3, 4\}$ (given that the maximum value is 3 and its first occurrence is in position 4). Clearly $b_1, \ldots, b_{\iota-1} < \beta$; and $b_{\iota+1}, \ldots, b_m \leq \beta$.

The contribution for a class, given the values of m, β, and ι is computed by:

$$\beta^m + \sum_{j=1}^{\iota-1} \beta^{j-1}(\beta+1)^{m-j}$$

and using the identity:

$$\sum_{j=1}^{\iota-1} \beta^{j-1}(\beta+1)^{m-j} \equiv (\beta+1)^m - \beta^{\iota-1}(\beta+1)^{m-\iota+1}$$

we obtain a simplified form:

$$\beta^m + (\beta+1)^m - \beta^{\iota-1}(\beta+1)^{m-\iota+1}$$

The contribution for a specific vector is equivalent to a number with $m-1$ digits in which the first $\iota - 1$ digits are of base β and $m - \iota$ digits are of base $\beta + 1$. Given that this transformation is well known we do not explain it in more detail.

Table 5 illustrates the use of the direct method of BP (**Algorithm 7**) to enumerate the first 27 vectors B with cardinality $m = 3$. For each vector, it is identified its β and ι values.

Algorithm 7 presents the pseudocode for the direct method of BP. It takes as input a vector B and gives as output an integer α. The values of β and ι are computed in the for loop starting in line 2. In the for loop starting in line 8 ϕ_1, ϕ_2, and ϕ_3 store each of the terms of $\beta^m + (\beta+1)^m - \beta^{\iota-1}(\beta+1)^{m-\iota+1}$ respectively. The contribution of the vector is computed in the while loop starting in line 21 and stored in α. In line 30 the final value is computed as the contribution of the specific vector plus the contribution of the class (computed as $\phi_1 + \phi_2 - \phi_3$).

Table 5: The first 27 vectors of dimension 3 and their corresponding integer values. For each vector the values of β and ι are given.

Num.	β	ι	Vector	Num.	β	ι	Vector
0	0	1	$(0,0,0)$	14	2	1	$(2,2,0)$
1	1	1	$(1,0,0)$	15	2	1	$(2,2,1)$
2	1	1	$(1,0,1)$	16	2	1	$(2,2,2)$
3	1	1	$(1,1,0)$	17	2	2	$(0,2,0)$
4	1	1	$(1,1,1)$	18	2	2	$(0,2,1)$
5	1	2	$(0,1,0)$	19	2	2	$(0,2,2)$
6	1	2	$(0,1,1)$	20	2	2	$(1,2,0)$
7	1	3	$(0,0,1)$	21	2	2	$(1,2,1)$
8	2	1	$(2,0,0)$	22	2	2	$(1,2,2)$
9	2	1	$(2,0,1)$	23	2	3	$(0,0,2)$
10	2	1	$(2,0,2)$	24	2	3	$(0,1,2)$
11	2	1	$(2,1,0)$	25	2	3	$(1,0,2)$
12	2	1	$(2,1,1)$	26	2	3	$(1,1,2)$
13	2	1	$(2,1,2)$				

The complexity of **Algorithm 7** is $O(n^2 \cdot m)$ (remember that n is the number of bits of the integer α and a multiplication requires $O(n^2)$ bit operations). The space complexity for this algorithm is again $O(m)$.

Algorithm 7 DirectBP(B, m)

```
1:  $\beta := b_1, \iota := 1$
2:  for $i := 2$ to $m$ do
3:      if $b_i > b_\iota$ then
4:          $\beta := b_i, \iota := i$
5:      end if
6:  end for
7:  $\phi_1 := 1, \phi_2 := 1, \phi_3 := 1$
8:  for $i := 1$ to $m$ do
9:      $\phi_1 := \phi_1 * \beta, \phi_2 := \phi_2 * (\beta + 1)$
10:     if $i < \iota$ then
11:         $\phi_3 := \phi_3 * \beta$
12:     else
13:         $\phi_3 := \phi_3 * (\beta + 1)$
14:     end if
15: end for
16: if $\iota = 1$ then
17:     $\alpha := v_2, i := 3$
18: else
19:     $\alpha := v_1, i := 2$
20: end if
21: while $i < m$ do
22:     if $i < \iota$ then
23:         $\alpha := \alpha * \beta + b_i$
24:     end if
25:     if $i > \iota$ then
26:         $\alpha := \alpha * (\beta + 1) + b_i$
27:     end if
28:     $i := i + 1$
29: end while
30: $\alpha := \alpha + \phi_1 + \phi_2 - \phi_3$
31: return $\alpha$
```

5.2 Inverse Method for BP

Once we have defined the direct method for BP, determining an inverse function is simple. First, we identify the values for β and ι After that, the values of $m-1$ components of the vector are computed as the mapping from a natural number to a number with $m-1$ digits, where the first $\iota-1$ are of base β and $m-\iota$ are of base $\beta+1$. The missing digit is inserted in the ι position and has the value β.

Algorithm 8 is the pseudocode of the inverse method for BP. The value of β is determined in line 1. The for loop in line 3 computes $\phi_1 = \beta^m$ and $\phi_2 = (\beta+1)^m$. The while loop in line 7 determines the value of ι. The value of the element b_ι must be β. The remaining components of the vector B are computed as the transformation of the number $\alpha - (\phi_1 + \phi_2 - \phi_3)$ to a number with $m-1$ digits (first $\iota-1$ digits are of base β and $m-\iota$ digits are of base $\beta+1$). This is done in the for loop in line 11.

Algorithm 8 *InverseBP*(α, m, B)

```
1:  $\beta := \lfloor \sqrt[m]{\alpha} \rfloor$
2:  $\phi_1 := 1, \phi_2 := 1$
3:  for $i := 1$ to $m$ do
4:      $\phi_1 := \phi_1 * \beta, \phi_2 := \phi_2 * (\beta + 1)$
5:  end for
6:  $\phi_3 := \phi_2, \iota := 1$
7:  while $\phi_1 + \phi_2 - \phi_3 * \beta/(\beta + 1) < \alpha$ do
8:      $\iota := \iota + 1, \phi_3 := (\phi_3 * \beta)/(\beta + 1)$
9:  end while
10: $\alpha := \alpha - (\phi_1 + \phi_2 - \phi_3), b_\iota := \beta$
11: for $i := m$ downto 1 do
12:     if $i > \iota$ then
13:         $b_i := \alpha$ mod $(\beta + 1), \alpha := \lfloor \alpha/(\beta + 1) \rfloor$
14:     end if
15:     if $i < \iota$ then
16:         $b_i := \alpha$ mod $\beta, \alpha := \lfloor \alpha/\beta \rfloor$
17:     end if
18: end for
19: return $B$
```

The complexity of **Algorithm 8** is $O(n^3 \cdot m)$. The cost of computing $\beta \leftarrow \lfloor \sqrt[m]{\alpha} \rfloor$ is of order $O(n^3 \cdot m)$ bit operations using the algorithm FηRA [30]. The loop in line 3 is repeated $O(m)$ times. Each time it does 2 multiplications, 1 division, and 1 addition; then, this loop requires $O(n^2 \cdot m)$ bit operations. The loop in line 7 searches the value of ι. The loop is repeated at most $O(m)$ times, each time it does 1 division, 1 multiplication, and 3 additions; then, its computational complexity is $O(n^2 \cdot m)$. The loop in line 11 computes 2 divisions and 2 additions $O(m)$ times, this loop then needs $O(n^2 \cdot m)$ bit operations. In conclusion, given that the cost of computing the integer m^{th} root dominates the others, the overall computational complexity of the inverse method for BP is $O(n^3 \cdot m)$. Again, the spatial complexity is $O(m)$.

6 Computational Complexity for GTP, DP, and BP

Table 6 presents the analysis of the computational complexity of DP, GTP, and BP. Column 1 shows the type of the polynomial. Column 2 presents the number of bit operations required to transform a vector into an integer, where m is the size of the vector. Column 3 shows the number of bit operations used to map an integer into a vector of m dimensions. Each algorithm presented in this paper has an spatial complexity $O(m)$, i.e. linear in the number of dimensions of the vector. The direct and inverse methods proposed in [11] and the one proposed in this paper require $O(n^3 \cdot m^2)$ and $O(n^2 \cdot m)$ bit operations. However, the inverse for GTP proposed in this paper avoids the use of floating point operations. Finally, according to the analysis shown in **Table 6**, we can conclude that the direct and inverse method for BP are the most efficient; since they require $O(n^2 \cdot m)$ and $O(n^3 \cdot m)$ bit operations respectively.

Table 6: Temporal complexity of the direct and inverse methods of DP, GTP, and BP. Here, m is the number of dimensions of the associated vector V and n the number of bits of its corresponding integer α.

Polynomial	Direct	Inverse
DP	$O(n^2 \cdot m^2)$	$O(n^3 \cdot m^2)$
GTP	$O(n^2 \cdot m^2)$	$O(n^3 \cdot m^2)$
BP	$O(n^2 \cdot m)$	$O(n^3 \cdot m)$

7 Discussion

This paper studied from a combinatorial viewpoint three packing functions: DP, BP, and GTP. Basically, we have presented a pseudocode to perform the bijections between each of these polynomials (represented as vectors over N^m) and the non negative integers. Also, we have included a theoretical analysis about the performance of each method. The analysis shows that BP represents the most efficient approach for a PF. Its direct method requires $O(n^2 \cdot m)$ bit operations. The inverse method requires $O(n^3 \cdot m)$ bit operations. Here, n denotes the number of bits of the integer being involved. With respect to GTP and DP, methods for their bijections were proposed. These methods can map a vector of dimension m into an integer value and vice versa, for both the order of the required bit operations is $O(n^2 \cdot m^2)$ for the direct method and $O(n^3 \cdot m^2)$ for the inverse method. The inverse method for GTP proposed in this paper was compared to the method proposed by Fraenkel [11] which, to the best of our knowledge, is the only general inverse method previously reported for a PF. Both methods require $O(n^3 \cdot m^2)$ bit operations for the inverse method. However, our approach avoids the use of floating point operations and requires only a basic set of operations: subtraction, addition, multiplication, and division. We also have presented two interesting applications of PFs: (a) manipulation of Covering Arrays (CAs) using GTP; and (b) generation of random solutions of a Linear Diophantine Equation with Unit coefficients (LDEU) using GTP and DP. Finally, the development of the bijection between DP and GTP raises a question concerning the existence of an efficient bijection between DP and BP, or GTP and BP. If this bijection has linear complexity in the number of dimensions m, then the direct and inverse methods for BP would make the calculation of the packing functions DP and GTP more efficient.

Acknowledgments

This research was partially funded by the following projects: 51623-Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas, 238469 - CONACyT Métodos Exactos para Construir Covering Arrays Óptimos; 232987 - CONACyT Conjuntos de Prueba Óptimos para Métodos Combinatorios (Optimal Test Sets for Combinatorial Methods); and the project Análisis de la Dificultad del Problema de Minimización de Ancho de Banda en Grafos por medio de Pareja de Invariantes supported by Programa de Mejoramiento al Profesorado PROMEP/103.5/12/3620. This paper was finished when the first author was a guest researcher in the U.S. National Institute of Standards and Technology, Gaithersburg MD 20899 USA.

Disclaimer

Any mention of commercial products in this paper is for information only; it does not imply recommendation or endorsement by NIST.

References

[1] H. Avila-George, J. Torres-Jimenez, N. Rangel-Valdez, A. Carrión, and V. Hernández. Supercomputing and grid computing on the verification of covering arrays. *The Journal of Supercomputing*, 62(2):916–945, 2012.

[2] K. Bell. *Optimizing effectiveness and efficiency of software testing: A hybrid approach*. PhD thesis, North Carolina State University, 2006.

[3] A. Brodnik, S. Carlsson, E. Demaine, J. Munro, and R. Sedgewick. Resizable arrays in optimal time and space. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures*, WADS 1999, pages 37–48, London, UK, 1999. Springer-Verlag.

[4] G. Cantor. Ein beitrag zur mannigfaltigkeitslhre. *J. Reine Angew. Math.*, 84:242–258, 1877.
[5] G. Cantor. Über eine elementare frage der mannigfaltigkeitslhre. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, I:75–78, 1891.
[6] E. Charlier, M. Rigo, and W. Steiner. Abstract numeration systems on bounded languages and multiplication by a constant. *Electronic Journal of Comb. Number Theory*, 8:1–15, 2008.
[7] P. Chowla. On some polynomials which represent every natural number exactly once. *Norske Vid. Selsk. Forth. Trondheim*, 34(2):8–9, 1961.
[8] R. Cilia and J. Gutiérrez. Dominated, diagonal polynomials on ℓ_p spaces. *Archiv der Mathematik*, 84:421–431, 2005.
[9] H. Fetter, J. A. R., and L. Morales. The diagonal polynomials of dimension four. *Advances in Applied Mathematics*, 34(2):316 – 334, 2005.
[10] A. Fraenkel. Systems of numeration. *Amer. Math. Monthly*, 92(2):105–114, 1985.
[11] A. Fraenkel and M. Mor. Combinatorial compression and partitioning of large dictionaries: theory and experiments. *SIGIR Forum*, 17:205–219, 1983.
[12] J. Gunnels and F. Gustavson. A new array format for symmetric and triangular matrices. *Journal of the Association for Computing Machinery*, 19(2):309–340, 1972.
[13] U. Gupta. Bounds on storage for consecutive retrieval. *Journal of the Association for Computing Machinery*, 26(1):28–36, 1979.
[14] D. Kuhn, D. Wallace, and A. Gallo Jr. Software fault interactions and implications for software testing. *IEEE Transactions on Software Engineering*, 30(6):418–421, 2004.
[15] P. Lecomte and M. Rigo. *Abstract numeration systems*, chapter 3, pages 108–162. Combinatorics, Automata, and Number Theory. Encyclopedia of Mathematics and its Applications No. 135. Cambridge University Press 2010, 2010.
[16] D. Lehmer. *The machine tools of combinatorics*, chapter 1, pages 5–31. Wiley, New York, 1964.
[17] J. Lew. Polynomial enumeration of multidimensional lattices. *Theory of Computing Systems*, 12:253–270, 1978.
[18] J. Lew, L. Morales, and A. Sánchez-Flores. Diagonal polynomials for small dimensions. *Theory of Computing Systems*, 29:305–310, 1996.
[19] J. Lew and A. Rosenberg. Polynomial indexing of integer lattice-points i. general concepts and quadratic polynomials. *Journal of Number Theory*, 10(2):192 – 214, 1978.
[20] J. Lew and A. Rosenberg. Polynomial indexing of integer lattice-points ii. nonexistence results for higher-degree polynomials. *Journal of Number Theory*, 10(2):215 – 243, 1978.
[21] Z. Liu, X. Li, and J. You. On storage schemes for parallel array access. In *Proceedings of the 6th international conference on Supercomputing*, ICS 1992, pages 282–291, New York, NY, USA, 1992. ACM.
[22] L. Morales. Diagonal polynomials and diagonal orders on multidimensional lattices. *Theory of Computing Systems*, 30:367–382, 1997.
[23] L. Morales and J. Lew. An enlarged family of packing polynomials on multidimensional lattices. *Theory of Computing Systems*, 29:293–303, 1996.
[24] A. Rosenberg. Addressable data graphs. *Journal of the Association for Computing Machinery*, 19(2):309–340, 1972.
[25] A. Rosenberg. Allocating storage for extendible arrays. *Journal of the Association for Computing Machinery*, 21(4):652–670, 1974.
[26] A. L. Rosenberg. Efficient pairing functions - and why you should care. *Parallel and Distributed Processing Symposium, International*, 2:0144, 2002.
[27] S. Ross. *A First Course in Probability*. CRC Press, New York, 1976.
[28] C. Smorynski. *Logical number theory, I. An introduction*. Springer-Verlag, Berlin, 1991.
[29] D. Stanton and D. White. *Constructive Combinatorics*. Undergraduate Texts in Mathematics. Springer, 1986.
[30] J. Torres-Jimenez, N. Rangel-Valdez, and P. Quiz-Ramos. An algorithm to compute integer ηth roots using subtractions. *International Journal of Computer Mathematics. First published on: 23 March 2011 (iFirst)*, 88(8):1629–1643, 2011.
[31] J. Torres-Jimenez and E. Rodriguez-Tello. New bounds for binary covering arrays using simulated annealing. *Information Sciences*, 185(1):137–152, 2012.
[32] M. Vsemirnov. Two elementary proofs of the fueter-pólya theorem on pairing polynomials. *Algebra i Analiz*, 13(5):1–15, 2001.

Jose Torres-Jimenez is a teacher and researcher at CINVESTAV-Tamaulipas, México. He obtained his Ph.D. from ITESM Cuernavaca in México. He is an expert in combinatorial optimization. He has graduated more than 10 PhD professionals and more than 50 MSc professionals. He has dedicated more than a decade to build many of the best-known covering arrays (mathematical objects that are used to do software and hardware testing). He has many international collaborations in USA, Spain, Colombia, France and Austria. He is a level II member of the national system of researches in México.

Nelson Rangel-Valdez is an Academic Appointee from the program called Cátedras of the National Council of Science and Technology (CONACyT) in Mexico. He has been assigned to the Tecnológico Nacional de Mexico, Instituto Tecnologico de Ciudad Madero (TecNM / ITCM). He has done relevant work in the field of combinatorial optimization. In his position as faculty appointee in the Postgraduate Division of Studies at the TecNM / ITCM he serves as an expert on combinatorial issues, attend graduate and undergraduate courses, and collaborates in the development of the project called “Optimization of Complex Problem”. He has a B.S. in Computer Systems Engineering from the Technological Institute of Ciudad Madero, and a Ph. D. in Computer Science from the Information Technology Laboratory at the Center of Research and Advanced Studies of the National Polytechnic Institute (CINVESTAV) in Tamaulipas, Mexico. He has been Member of the National System of Researchers in Mexico since 2014 and has held positions at the Polytechnical University of Victoria

Raghu N. Kacker is a researcher in the Applied and Computational Mathematics Division of NIST. His current interests include software testing and evaluation of the uncertainty in outputs of computational models and physical measurements. He has a Ph.D. in statistics and is a Fellow of the American Statistical Association, and American Society for Quality.

James F. Lawrence is a professor of mathematics at George Mason University. He is a respected authority in the field of convex and combinatorial geometry. In his position as faculty appointee in the Applied and Computational Mathematics Division at NIST he serves as an expert on combinatorial issues. He has a B.S. in mathematics from Oklahoma State University and a Ph.D. in mathematics from the University of Washington. He has been a National Research Council Postdoctoral Fellowship at NIST and has held positions at the University of Texas at Austin, the University of Massachusetts at Boston, and the University of Kentucky.