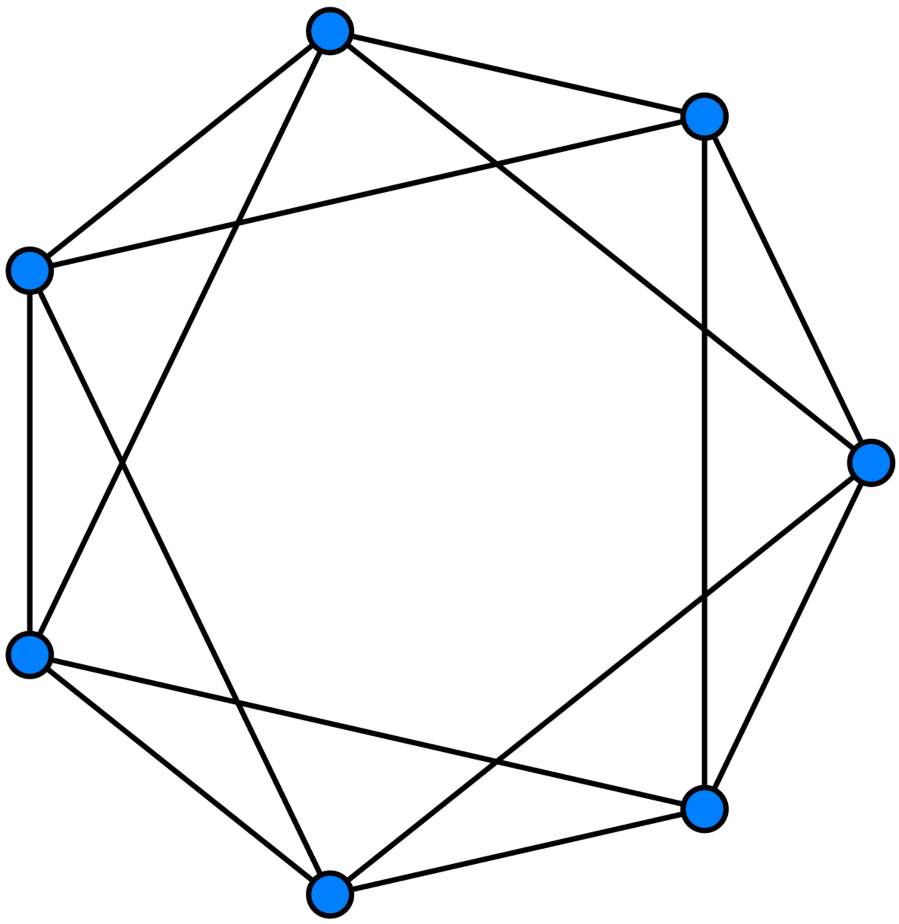
Bachelor thesis
The Shannon capacity on $C_{n,k}$



Timo den Dulk

(4954017)

supervised by: Dion Gijswijt

July 12, 2022

Abstract

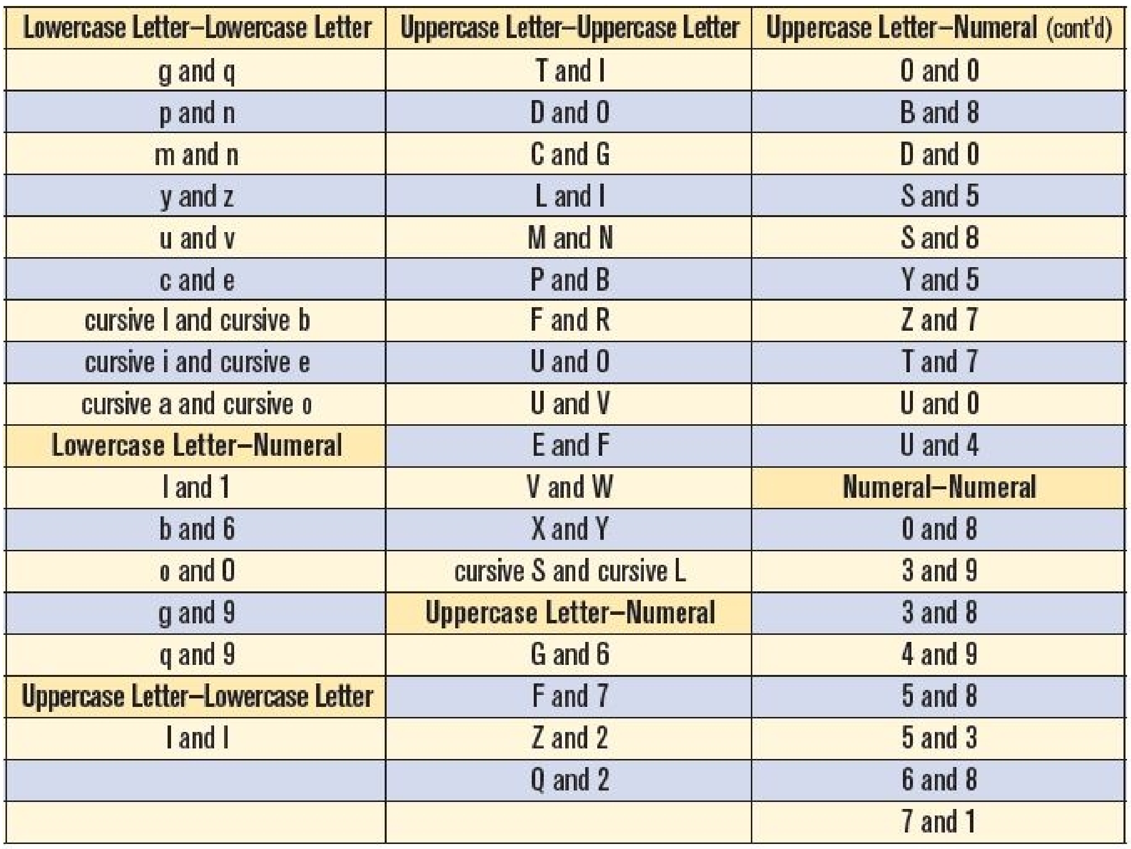
This thesis focuses on a problem formulated by Claude Shannon named the Shannon capacity. This problem is about information rate per time unit over a noisy channel. The noisy channel is here represented by a graph. We specifically focus on a class of circulant graphs that are denoted by $C_{n,k}$ with vertex set $\mathbb{Z}/n\mathbb{Z}$, where all vertices are connected with the $k-1$ vertices before and after it. We will discuss upper bounds that were found for the Shannon capacity and how $C_{n,k}$ behaves with these upper bounds. After that we will focus on multiple ways to calculate lower bounds for the Shannon capacity of $C_{n,k}$. For these three search methods will be used. These are exhaustive searching for optimal values, optimal ways to make packagings and solutions created by using a special form. As last the answers will be discussed by combining the upper and lower bounds for $C_{n,k}$. From this conclusions are drawn after which some possibilities will be given for further research.

Contents

Chapter 1

Introduction

Imagine someone left you a handwritten note with a password consisting of numbers and capital letters. But after using the password we get a message that we used the wrong password. What might have happened is that we interpreted certain characters wrongly. For example a 4 and 9 can be confused with each other if the upper part of the 9 is not a fully closed circle. Furthermore we might also confuse a 0 and O or 7 and 1. There are actually a lot more possible characters that can be confused as shown here in Table 1.1.



Lowercase Letter–Lowercase Letter	Uppercase Letter–Uppercase Letter	Uppercase Letter–Numeral (cont'd)
g and q	T and I	O and 0
p and n	D and O	B and 8
m and n	C and G	D and 0
y and z	L and I	S and 5
u and v	M and N	S and 8
c and e	P and B	Y and 5
cursive l and cursive b	F and R	Z and 7
cursive i and cursive e	U and O	T and 7
cursive a and cursive o	U and V	U and 0
Lowercase Letter–Numeral	E and F	U and 4
l and 1	V and W	**Numeral–Numeral**
b and 6	X and Y	0 and 8
o and 0	cursive S and cursive L	3 and 9
g and 9	**Uppercase Letter–Numeral**	3 and 8
q and 9	G and 6	4 and 9
Uppercase Letter–Lowercase Letter	F and 7	5 and 8
I and l	Z and 2	5 and 3
	Q and 2	6 and 8
		7 and 1

Table 1.1: A table with all possible misinterpretations between numbers and letters [7].

For simplification assume the password consisted of the characters {1,4,7,9,T}. Now we get 3 possible wrong interpretations. Firstly 4 and 9, secondly 1 and 7 and as last 7 and T. This problem is shown graphically in Figure 1.1.

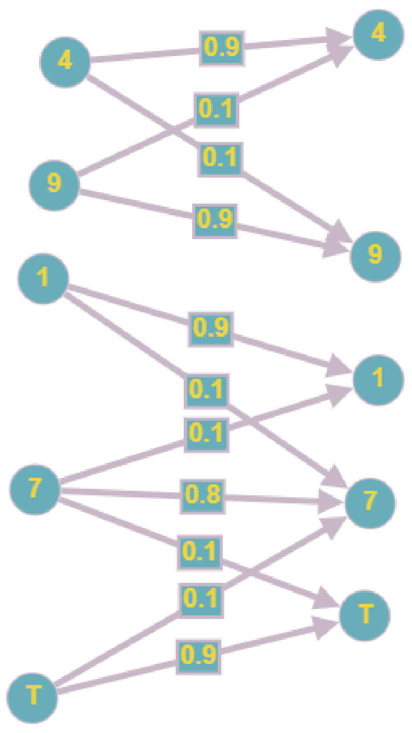


Figure 1.1: Possible misinterpretations with the chances that they happen.

The figure shows the assumed chances that we get a misinterpretation, but sometimes we want certainty that we got the correct password and thus zero possibility of getting the wrong one. To achieve this we can remove a set of code words from the total set of all possible code words. Assume there are code words of length 1, this gives a total of 5 code words namely {1,4,7,9,T}. Instead of this it is possible to take the set of 3 code words namely {1,4,T}. For this set we achieved zero possibility of using the wrong password since none of these code words can be confused with each other.

Now assume code words of length 2, this gives us a total set of 25 code words since there are 5 possibilities for position 1 and also 5 for position 2 in the code words. Now a possible set with code words that gives us certainty is given by.

$$\{(1,1),(1,4),(1,T),(4,1),(4,4),(4,T),(T,1),(T,4),(T,T)\}$$

This is a set of 9 code words simply created by using all combinations with elements of the zero error set of code words of length 1 {1,4,T}. For this specific problem with code words of length n it is possible to create a zero error set of size 3 to the power n created by the set {1,4,T} at n positions.

Instead of the visualization in Figure 1.1 we can also show our problem as a graph.

Definition 1.1. *(graph) A graph is a pair $G = (V, E)$. Where V is a set of elements in the graph and E is a set of unordered pairs of elements from V.*

Definition 1.2. *(independent set) Given a graph $G = (V, E)$. A set $S \subseteq V$ is independent in G if $\forall x, y \in S\ \{x, y\} \notin E$.*

Now it is possible to change the problem with code words of length 1 into a graph. This is done by making every possible symbol a vertex and if symbols can be confused with each other then an edge is added between those two vertices. Then the problem with {1,4,7,9,T} can be changed into the graph shown in Figure 1.2.

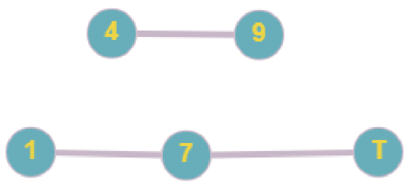


Figure 1.2: Graph G with the code words {1,4,7,9,T} and their edges.

In this graph every possible set with zero error for code words of length 1 is given by a independent set in the graph. From this follows that a biggest set with zero error for code words of length 1 is given by a biggest independent set in G. As said before {1,4,T} is such a set, but now by simple observation of the graph this is a biggest independent set. For the code words of length 2, the graph is given by the largest independent set in the strong product of G with itself.

Definition 1.3. *(Strong product of graphs) Given the graphs $G = (V, E), H = (W, F)$. the strong product of G and H is given as $G \boxtimes H = (V \times W, D)$. here $V \times W$ is the Cartesian product and*

$$\forall (u, w), (v, x) \in V \times W : \{(u, w)(v, x)\} \in D \iff \begin{cases} \{u, v\} \in E, \{w, x\} \in F \text{ or} \\ \{u, v\} \in E \text{ and } w = x \text{ or} \\ u = v \text{ and } \{w, x\} \in F \end{cases}$$

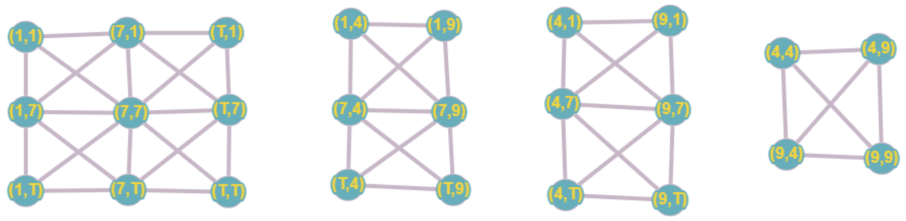


Figure 1.3: The strong product of G with itself.

After observation of $G \boxtimes G$ it can be found that we can take respectively at most 4,2,2 and 1 points from the 4 parts of the graph shown in figure 1.3. As said before there is a biggest independent set of G given by

$$\{(1,1),(1,4),(1,T),(4,1),(4,4),(4,T),(T,1),(T,4),(T,T)\}$$

This is thus actually a biggest independent set in $G \boxtimes G$.

1.1 The Shannon capacity of a graph

Based on this problem Claude Shannon defined the Shannon capacity[1]. For this he viewed the problem in another way. Namely the information rate per time unit over a noisy channel. The noisy channel mentioned is actually the graph we presented with its edges as noise. Shannon used symbols per time unit since writing a code word of length 2 costs the same amount of time as writing 2 code words of length 1.

Definition 1.4. *(Shannon capacity) Given $\alpha(H)$ as the size of a biggest independent set in H and G^n as the strong product of G with itself n times then the Shannon capacity is given as:*

$$\Theta(G) = \sup_{d \in \mathbb{N}} \sqrt[d]{\alpha(G^d)}.$$

Here the root follows from that we are interested in the information rate per single character instead of per n characters. It actually is so that we also have

$$\Theta(G) = \sup_{d \in \mathbb{N}} \sqrt[d]{\alpha(G^d)} = \lim_{d \to \infty} \sqrt[d]{\alpha(G^d)}.$$

This is proven by the following 2 lemmas.

Lemma 1.5. *For all graphs G and H, the following is true*

$$\alpha(G)\alpha(H) \leq \alpha(G \boxtimes H).$$

Proof. Let S be a largest independent set of vertices in G and let T be a largest independent set of vertices in H. Then by definition $|S| = \alpha(G)$ and $|T| = \alpha(H)$. Let $S \times T$ be the Cartesian product. It is simple to observe that $S \times T \subseteq V(G \boxtimes H)$ and $|S \times T| = \alpha(G)\alpha(H)$. Let (v_1, w_1) and (v_2, w_2) be two vertices in $S \times T$. Then these vertices are for sure not adjacent since v_1 and v_2 are not adjacent and w_1 and w_2 are not adjacent. Thus by definition of the strong product (v_1, w_1) and (v_2, w_2) are not adjacent in $G \boxtimes H$. From this it follows that $S \times T$ is an independent set in $G \boxtimes H$ of size $|S| \times |T| = \alpha(G)\alpha(H)$ so $\alpha(G)\alpha(H) \leq \alpha(G \boxtimes H)$. $\square$

This is used in the following lemma.

Lemma 1.6. *(Fekete's lemma) Let $f : \mathbb{N} \to \mathbb{N}$ be a function with $f(m+n) \geq f(m)f(n)$ for all $m, n \in \mathbb{N}$. Then $\lim_{n\to\infty} \sqrt[n]{f(n)}$ exist and*

$$\sup_{n\in\mathbb{N}} \sqrt[d]{f(n))} = \lim_{n\to\infty} \sqrt[d]{f(n)}.$$

Proof. Let $n = mq+r$ with $m, q.r \in \mathbb{N}$. Then by our assumptions $f(mq+r) \geq f(m)^q f(r)$. Now we get for fixed m and $r < m$

$$\begin{aligned}
\lim_{q\to\infty} \inf f(mq+r)^{\frac{1}{mq+r}} &\geq \lim_{q\to\infty} \inf(f(m)^q f(r))^{\frac{1}{mq+r}} \\
&= \lim_{q\to\infty} \inf(f(m)^{\frac{q}{mq+r}} f(r)^{\frac{1}{mq+r}}) \\
&= \lim_{q\to\infty} \inf((f(m)^{\frac{1}{m}})^{\frac{mq}{mq+r}} f(r)^{\frac{1}{mq+r}}) \qquad (1.1) \\
&= (f(m)^{\frac{1}{m}})^1 f(r)^0 \\
&= \sqrt[m]{f(m))}.
\end{aligned}$$

Thus for any m

$$\lim_{n\to\infty} \inf \sqrt[n]{f(n)} = \inf_{0\leq r<m} \lim_{q\to\infty} \inf f(mq+r)^{\frac{1}{mq+r}} \geq \sqrt[m]{f(m))}.$$

now by taking the supremum over m the lemma is proven. $\square$

Take for a graph G now $f(n) = a(G^n)$ then by Lemma 1.4 we got that $f(m+n) \geq f(m)f(n)$ and thus by Lemma 1.5 we got that the limit exist and is equal to the supremum.

Now it is thus shown that the Shannon capacity can always be found by letting the lengths of the code words go to infinity. But the problem for finding a biggest independent set grows exponentially with lengths. Thus instead of this we search for possible upper and lower bounds for the Shannon capacity of a graph. With this we can find a certain interval for the Shannon capacity on a graph and it might even give the exact answer when the upper and lower bound coincide.

1.2 The circulant graph $C_{n,k}$

Instead of the confusion between written text, there is a problem that is even more interesting. When we send a message over the internet, then this message changes into code words that your computer or telephone understands. For this problem the message changes into code words consisting of the n numbers created by working over $\mathbb{Z}/n\mathbb{Z}$. The problem is that there is a possibility that while sending this message these numbers change. So we might say that a number after sending can differ from its original value by a value ϵ with $\epsilon \in [0, a)$. Thus a number b can change into a value in the interval$(b-a, b+a)$ mod n.

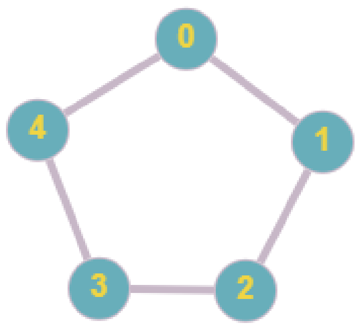
When translating this problem to a graph we get a special variant of the circulant graphs.

Definition 1.7. *(Circulant graph C_n^S) A circulant graph $C_n^S = (V, E)$ is a graph on $V = \mathbb{Z}/n\mathbb{Z}$ with $S \subseteq \mathbb{Z}/n\mathbb{Z}$ such that*

$$u, v \in E \iff u - v \in S \text{ or } v - u \in S$$

The special thing about the problem is that when $\epsilon \in [0, a)$ we got $S = \{1, \dots, a\}$. The notation that will be used for the special circulant graph will be $C_{n,k}$ which is equal to C_n^S where $S = \{1, \dots, k-1\}$. That $k \notin S$ is because of a special characteristic of $C_{n,k}$ that later will be shown.

As example $C_{5,2}$ and $C_{8,3}$ are shown in figure 1.4



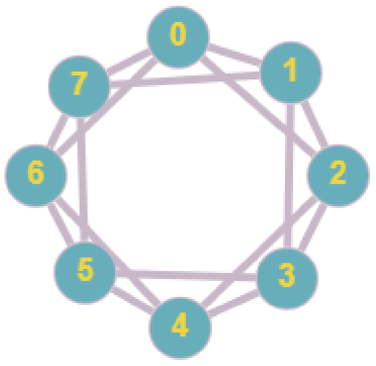


Figure 1.4: (left):$C_{5,2}$ (right): $C_{8,3}$

For many $C_{n,k}$ is the Shannon capacity not given by a biggest independent set of length 1.
For example $\alpha(C_{5,2}) = 2$ formed by{0,2}, but $\alpha(C_{5,2}^2) = 5$ formed by {(0,0),(1,2),(2,4),(3,1),(4,3)}.
The former example shows that the capacity can increase for higher lengths since $2 < \sqrt{5}$.

Chapter 2

Upper bounds on the shannon capacity of $C_{n,k}$

There are multiple known ways for finding upper bounds on the Shannon capacity of a graph. In this chapter well known upper bounds are given with their corresponding values for $C_{n,k}$.

Definition 2.1. *(Clique) Given a graph $G = (V, E)$. A* clique *in G is a set $C \subseteq V$ such that for all $v, w \in C$ with $v \neq w$ we got $\{v, w\} \in E$.*

Definition 2.2. *(Clique cover number) Given a graph $G = (V, E)$. The* clique cover number *of G noted as $\overline{\chi}(G)$ is the minimum numbers of cliques such that the union is V.*

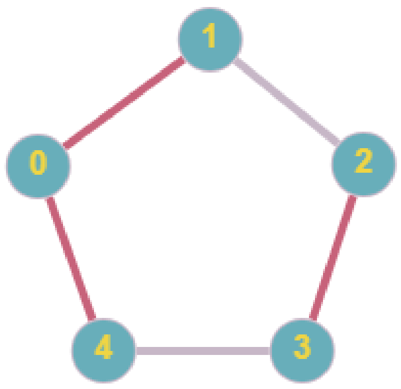


Figure 2.1: $C_{5,2}$ covered in the three cliques formed by the red edges.

For the graph $C_{n,k}$ a clique of maximum size contains k vertices. This clique is formed by the k consecutive vertices $\{a, a + 1, \ldots, a + k - 1\}$ mod n with $a \in \mathbb{Z}/n\mathbb{Z}$. When covering $C_{n,k}$ with cliques then at least $\lceil n/k \rceil$ cliques are needed, since a clique can not be bigger than a clique of maximum size. It is also possible to construct a clique cover of size $\lceil n/k \rceil$.

This is done by taking the first $\lfloor n/k \rfloor$ cliques containing $\{ak, ak+1, \dots, ak+k-1\}$ with $a \in 0, 1, \dots, \lfloor n/k \rfloor - 1$. If $\lfloor n/k \rfloor = \lceil n/k \rceil$ then all $v \in V(G)$ are in the cliques otherwise we can add the clique $\{\lfloor n/k \rfloor k, \lfloor n/k \rfloor k + 1, \dots, \lfloor n/k \rfloor k + k - 1\}$. Since $\lfloor n/k \rfloor k + k - 1 \geq n$, it follows that all $v \in V(G)$ are in the clique cover of size $\lceil n/k \rceil$.

When an independent set is constructed then it contains at most a single vertex from every clique, since every two vertices in a clique are connected by an edge. This means that $\alpha(C_{n,k}) \leq \lceil n/k \rceil$. It is also known that the strong product of two cliques is a clique and by using this a cover for the strong product can be created. A clique cover for $\alpha(C_{n,k}^d)$ can be constructed by taking all the cliques formed by the Cartesian products of d cliques of $C_{n,k}$, here a clique can be used multiple times. Then the number of cliques for this clique cover is $\lceil n/k \rceil^d$, since $\lceil n/k \rceil$ cliques can be chosen d times. Now it follows that $\alpha(C_{n,k}^d) \leq \lceil n/k \rceil^d$ for all d and thus $\Theta(C_{n,k}) \leq \lceil n/k \rceil$.

For $C_{n,k}$ the clique cover number is a very easy upper bound to calculate, for which is known that $\overline{\chi}(C_{n,k}) = \lceil n/k \rceil$.

2.1 Fractional clique cover number

Shannon [1] gave in his own article an upper bound for the Shannon capacity, that is now known as the fractional clique cover number.

Definition 2.3. *(Fractional clique cover number) Given a graph $G = (V, E)$ and the set C containing all cliques of G. The* fractional clique cover number *of G noted as $\alpha^*(G)$ is given by:*

$$\alpha^*(G) = \max_x \sum_{v \in V} x_v$$

where

$$x \in \mathbb{R}^V$$
$$\forall c \in C : \sum_{v \in c} x_v \leq 1$$
$$\forall v \in V : x_v \geq 0.$$

Assume $\alpha(G) = a$, then there exists an independent set $S \subseteq V$ of size a. Now take if $v \in S$ then $x_v = 1$ and else $x_v = 0$. All x_v are positive and every clique contains at most 1 element from S thus the sums over the cliques are either 1 or 0. So this x is a feasible solution and since we are searching for a maximum, this will result in a lower bound for $\alpha^*(G)$ of $1|S| = a$. Thus $\alpha^*(G) \geq \alpha(G)$.

What actually is wanted is an upper bound for the Shannon capacity and thus for $\sqrt[d]{\alpha(G^d)}$. This is achieved when $\alpha^*(G \boxtimes H) \leq \alpha^*(G)\alpha^*(H)$, since then

$$\sqrt[d]{\alpha(G^d)} \leq \sqrt[d]{\alpha^*(G^d)} \leq \sqrt[d]{(\alpha^*(G))^d} = \alpha^*(G).$$

By the duality theorem of linear programming we got for a graph $G = (V, E)$ and the set C containing all cliques of G:

$$\alpha^*(G) = \min_y \sum_{c \in C} y_c$$

where

$$y \in \mathbb{R}^C$$

$$\forall v \in V : \sum_{c \ni v} y_c \geq 1$$

$$\forall c \in C : y_c \geq 0.$$

This dual version is the motivation for the name 'fractional clique cover number'. Using the dual version consider that for G we achieve $\alpha^*(G)$ with $y \in \mathbb{R}^C$ and for H we achieve $\alpha^*(H)$ with $y' \in \mathbb{R}^{C'}$. Then take the vector:

$$Y = \{Y_{(i,j)} = y_i * y'_j : y_i \in y \text{ and } y'_j \in y'\}.$$

For this all $Y_{(i,j)}$ are non-negative and $\forall (v,w) \in V(G) \times V(H)$

$$\sum_{(c_1,c_2): v \in c_1, w \in c_2} Y_{c_1,c_2} = \sum_{c_1 \ni v} y_{c_1} \sum_{c_2 \ni w} y'_{c_2} \geq 1 * 1.$$

Thus this is a dual feasible solution for $G \boxtimes H$ and thus by the minimisation we got that this is an upper bound for $\alpha^*(G \boxtimes H)$ with

$$\alpha^*(G \boxtimes H) \leq \sum_{Y_{(c_1,c_2)} \in Y} Y_{c_1,c_2} = \sum_{y_{c_1} \in y} y_{c_1} \sum_{y'_{c_2} \in y'} y'_{c_2} = \alpha^*(G)\alpha^*(H).$$

Thus

$$\sqrt[d]{\alpha(G^d)} \leq \alpha^*(G) \text{ for all } d.$$

From this follows

$$\Theta(G) \leq \alpha^*(G).$$

Now for calculating $\alpha^*(C_{n,k})$, it is possible to create a lower bound with the maximisation problem and an upper bound with the minimisation problem.

For the maximisation take $x = \{1/k, 1/k, \ldots, 1/k\}$. As said before the cliques are of maximum size k thus all cliques have a value of at most 1. This gives the lower bound $\alpha^*(C_{n,k}) \geq n/k$.

For the minimisation take $y = \{y_0, y_1, \ldots, y_{|C|-1}\}$ where $y_i = 1/k$ if $|i| = k$ and 0 otherwise. Now the n cliques of maximum size k consisting of k consecutive vectors all have value $1/k$ and all other cliques have value zero. Every vertex is now contained in k cliques of value $1/k$ and thus always bigger or equal to 1. This gives the lower bound $\alpha^*(C_{n,k}) \leq n/k$. Thus $\alpha^*(C_{n,k}) = n/k$.

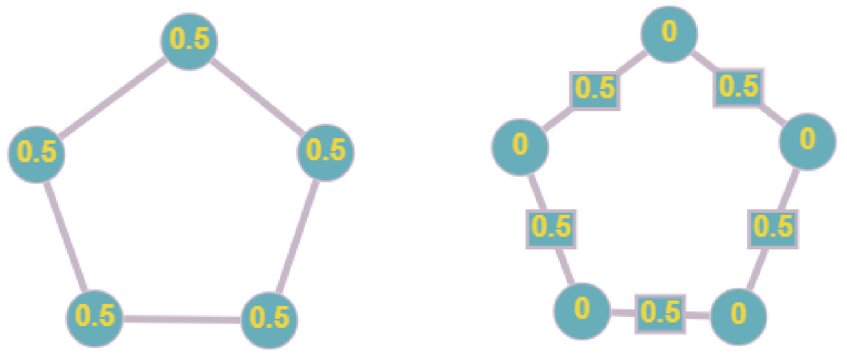


Figure 2.2: $\alpha^*(C_{5,2})$ with (left) maximum over vertices (right) minimum over cliques.

2.2 The Lovász theta function

Lovász [2] found an other upper bound for the Shannon capacity, that is known as the Lovász theta function.

Definition 2.4. *(Orthonormal representation of a graph) Given a graph $G = (V, E)$. An* orthonormal representation *of G is a map $f : V \to \mathbb{R}^n$ where $f(v)$ is a unit vector such that for $i \neq j$ if $\{i, j\} \notin E$ then $f(v_i)$ and $f(v_j)$ are orthogonal.*

Definition 2.5. *(Lovász theta function) given a graph $G = (V, E)$. Then the* Lovász theta function *$\vartheta(G)$ is given by:*

$$\vartheta(G) = \min_{U,c} \max_{u \in U} \frac{1}{\langle c, u\rangle^2}.$$

Where U is a orthonormal representation of G and c a unit vector.

When setting a orthonormal representation U fixed then the c which gives the lowest value is called the handle.

Lemma 2.6.

$$\alpha(G) \leq \vartheta(G).$$

Proof. Given a optimal orthonormal representation U of a graph $G = (V, E)$, with handle c. Let $S \subseteq V$ be a largest independent set in G. It is known that since c is a unit vector it follows that $\langle c, c\rangle^2 = 1$. Since S creates orthogonal unit vectors, it follows that:

$$1 = \langle c, c\rangle^2 \geq \sum_{i \in S} \langle c, u_i\rangle^2 \geq \alpha(G)/\vartheta(G).$$

and thus $\alpha(G) \leq \vartheta(G)$. □

Now it is proved that the Lovász theta function is an upper bound for $\alpha(G)$, but as before we would like to have an upper bound for the Shannon capacity and thus that $\vartheta(G \boxtimes H) \leq \vartheta(G)\vartheta(H)$, since then

$$\sqrt[d]{\alpha(G^d)} \leq \sqrt[d]{\vartheta(G^d)} \leq \sqrt[d]{(\vartheta(G))^d} = \vartheta(G).$$

Lemma 2.7.
$$\vartheta(G \boxtimes H) \leq \vartheta(G)\vartheta(H).$$

Proof. Let $s \otimes t$ be the tensor product of vector s with vector t. For this it is known that $(s \otimes t)^T(u \otimes v) = \langle s, u\rangle\langle t, v\rangle$. Let $v = \{v_0, v_1, \dots, v_{|V(G)|-1}\}$ be an optimal orthonormal representation for G with handle c and $w = \{w_0, w_1, \dots, w_{|V(H)|-1}\}$ an optimal orthonormal representation for H with handle d. Now take for $G \boxtimes H$ the orthonormal representation $v \otimes w$ with handle $c \otimes d$. This is possible since the tensor product of unit vectors are still unit vectors. Now by minimisation it follows that

$$\vartheta(G \boxtimes H) \leq \max_{v \in U, w \in W} \frac{1}{((c \otimes d)^t(v \otimes w))^2} = \max_{v \in U, w \in W} \frac{1}{\langle c, v\rangle^2} \frac{1}{\langle d, w\rangle^2} = \vartheta(G)\vartheta(H).$$

□

Theorem 2.8.
$$\Theta(G) \leq \vartheta(G).$$

Proof. This directly follows from Lemma 2.7 and Lemma 2.6. □

For finding the Lovász theta function Lovász [2] created a semidefinite program.

Theorem 2.9. *given a graph $G = (V, E)$.*

$$\vartheta(G) = \max_B \sum_{i,j \in V} b_{i,j}$$

were B is a symmetric positive semidefinite $|V| \times |V|$ matrix and

$$\sum_{i \in V} b_{i,i} = 1$$

$$b_{i,j} = 0 \text{ for all } \{i, j\} \in E.$$

Definition 2.10. *(Eigenvectors/eigenvalues) Given a matrix A. A nonzero vector v is an* eigenvector *of A with* eigenvalue *λ when.*

$$Av = \lambda v$$

A symmetric matrix is positive semidefinite when all eigenvalues are non-negative. With this it is possible to calculate $\vartheta(G)$ for all graphs. From now on the focus will be again on the graphs $C_{n,k}$. The structure of an optimal symmetric matrix to calculate $\vartheta(C_{5,2})$ is given by:

$$\begin{bmatrix} a & 0 & b & c & 0 \\ 0 & d & 0 & e & f \\ b & 0 & g & 0 & h \\ c & e & 0 & i & 0 \\ 0 & f & h & 0 & j \end{bmatrix}$$

Now this matrix has a basis of 5 eigenvectors with there corresponding non-negative eigenvalues. Consider the eigenvector $(A, B, C, D, E)^T$ with eigenvalue $\lambda \geq 0$. When the graph

$C_{5,2}$ gets rotated once, then the values in the matrix will rotate one to the right and one down. This matrix now has the eigenvector $(E, A, B, C, D)^T$ with eigenvalue λ. From this follows that all eigenvalues stay the same and thus that all possible rotations of $C_{5,2}$ provide optimal solutions. Since the sum of positive semidefinite matrices are positive semidefinite it follows that the average of these optimal solutions is an optimal solution. So the next structure follows:

$$\begin{bmatrix} a & 0 & b & b & 0 \\ 0 & a & 0 & b & b \\ b & 0 & a & 0 & b \\ b & b & 0 & a & 0 \\ 0 & b & b & 0 & a \end{bmatrix}$$

Since the diagonal has to sum up to 1, it follows that $5a = 1$. Thus $a = 1/5$ and only one variable remains.

$$\begin{bmatrix} 1/5 & 0 & b & b & 0 \\ 0 & 1/5 & 0 & b & b \\ b & 0 & 1/5 & 0 & b \\ b & b & 0 & 1/5 & 0 \\ 0 & b & b & 0 & 1/5 \end{bmatrix}$$

For finding the optimal solution of this matrix and all other matrices for $C_{n,k}$ a set of matrices is created. Take $A_{n,k}$ as the n by n matrix that is filled with 0's and on all places k and $n - k$ from the diagonal 1 is added. For this it is known that for $k \leq n/2$:

$$A_{n,k} = A_{n,1}A_{n,k-1} - A_{n,k-2}$$

This is so since when placing 1's one place further from the diagonal then $A_{n,k-1}$ gets achieved by $A_{n,1}A_{n,k-1}$, but this also places 1's one place closer to the diagonal. This error is removed by $-A_{n,k-2}$. For $k = n/2$ one place further from the diagonal coincides when going right or left from the diagonal thus it follows into 2's, which works for $A_{n,k}$ since 1 is added for $n/2$ and $n - n/2$ from the diagonal.

Lemma 2.11. *Given a symmetric matrix A with eigenvector v and corresponding eigenvalue λ then:*

$$A^d v = \lambda^d v.$$

Proof. For $d = 1$ follows directly from Definition 2.10 and thus:

$$\lambda^d v = \lambda^{d-1}\lambda v = \lambda^d A v = A\lambda^{d-1} v = \cdots = A^{d-1}\lambda v = A^d v.$$

□

Lemma 2.12. *Given two symmetric matrices A and B with eigenvector v such that*

$$Av = \lambda_1 v \text{ and } Bv = \lambda_2 v$$

then

$$(A + B)v = (\lambda_1 + \lambda_2)v$$

Proof.

$$(A+B)v = Av + Bv = \lambda_1 v + \lambda_2 v = (\lambda_1 + \lambda_2)v$$

□

By induction on $A_{n,k}$ it is possible to write it as a sum over the powers of $A_{n,1}$, with $A_{n,1}^0 = A_{n,0}$. From Lemma 2.11 and Lemma 2.12 it follows that the sum of powers of $A_{n,1}$ has the same eigenvectors as $A_{n,1}$ with corresponding eigenvalues, which can be calculated by using the lemmas. Thus now the eigenvalues of $A_{n,k}$ can be calculated, which will be noted as $\lambda_i(A_{n,k})$ corresponding to the eigenvector v_i. By using this the problem for $C_{5,2}$ can be written as a sum of $A_{5,k}$, from which the following maximisation problem follows.

$$\max(1 + 10b).$$

Such that for all eigenvectors v_i with corresponding eigenvalues $\lambda_i(A_{5,k})$

$$0 \leq 1 + b\lambda_i(A_{5,2}).$$

For this it clearly follows that we want b as big as possible thus only the smallest $\lambda_i(A_{5,2})$ matters. This eigenvalue is $-\frac{1+\sqrt{5}}{2}$, which results in $b = \frac{2}{5(1+\sqrt{5}}$ and thus

$$\vartheta(C_{5,2}) = 1 + 10\frac{2}{5(1+\sqrt{5})} = 1 + \frac{4}{1+\sqrt{5}} = \frac{5+\sqrt{5}}{1+\sqrt{5}} = \sqrt{5}\frac{1+\sqrt{5}}{1+\sqrt{5}} = \sqrt{5}.$$

In Chapter 1.2 was shown that $\Theta(C_{5,2}) \geq \sqrt{5}$, thus it actually is so that $\Theta(C_{n,k}) = \sqrt{5}$.

Using the method shown before with the known eigenvalues it follows that for $C_{n,k}$, there are $\lfloor n/2 \rfloor - k + 1$ unknown values with n eigenvalues to check. Not all eigenvalues need to be checked, since for odd n there is always one eigenvalue equal to $2n$ and can be ignored since that eigenvalue is always positive. All other eigenvalues exist 2 times for these matrices and thus only $\lfloor n/2 \rfloor$ need to be checked. From this follows the maximisation problem:

$$\vartheta(C_{n,k}) = max(1 + 2n \sum_{j\in\{0,1,\ldots,\lfloor n/2\rfloor - k\}} a_j)$$

Such that for all eigenvectors v_i with corresponding eigenvalues $\lambda_i(A_{n,k})$

$$0 \leq 1 + \sum_{j\in\{0,1,\ldots,\lfloor n/2\rfloor - k\}} a_j \lambda_i(A_{n,k+j}).$$

With this it is possible to calculate $\vartheta(C_{n,k})$ for all n and k, but Bachoc et al.[9] found a formula for $\vartheta(C_{n,k})$ which computes the answers really fast.

Theorem 2.13. *for $k \geq 2$, $n \geq 2k$ and $0 \leq i \leq d-1$, let*

$$c_i = \cos\frac{2i\pi}{k} \text{ and } a_i = \cos\left\lfloor\frac{ni}{k}\right\rfloor\frac{2\pi}{n}.$$

Then

$$\vartheta(C_{n,k}) = \frac{n}{k}\sum_{i=0}^{k-1}\prod_{j=1}^{k-1}\left(\frac{c_i - a_j}{1 - a_j}\right)$$

Chapter 3

Lower bounds on the shannon capacity of $C_{n,k}$

There are multiple ways to find lower bounds on the Shannon capacity, but they all have something in common. All these lower bounds are created by finding a lower bound for $a(G^d)$ which is better than the last known lower bound. The problem of finding a new lower bound can be divided into three possibilities. First straightforward by testing all or a lot of possible solutions. Secondly searching for optimal solutions by solving packaging problems. As last using a specific characteristic such that the number of options that need to be checked are limited.

3.1 Exhaustive searching

By using exhaustive searching, there are 2 different possibilities. Namely by checking a lot of combinations or by removing points from a working independent set to try to recreate a bigger independent set.

For these the order of the vertices in the independent set does not matter. Since the solution of two vertices {0,1} is equal to {1,0}. By ordering all vertices it is possible to solve this double checking, by assuming that the elements in the independent set also need to be added in this order.

There is also a possibility that there are vertices that should be in a largest independent set. For example a vertex with zero edges will for sure be part of all largest independent sets. It is also so that if a vertex has 1 edge then it is part of a largest independent set. This is so since a largest independent set has for sure either that one vertex or its neighbour. Now it is certain that a set created with that vertex is at least of the same size as when using its neighbour. By using that vertex it is still possible to use the other neighbours of the neighbour for a largest independent set. This is something that can be done before checking all possible combinations such that the problem is faster to solve.

This can also be applied during the search for independent sets. Assume we have an independent set and there exists a vertex, which is not in the set that is part of a largest independent set, consisting of at least the current set. Then it is known that we need to add this vertex. However, it is possible that this vertex has a lower order than the last vertex in the set and thus can not be added anymore. Thus this independent set can never create a larger independent set then the independent set where the vertex was added with correct ordering. It is known that a vertex had to be added when it has at most 1 neighbour in the vertices that still have to be checked. Only the ones that still have to be checked matter since otherwise it still follows that a vertex was missed.

In the appendix a code is given and for this code it was not possible to simply order the vertices. Since the strong product gives every vertex multiple values, but networkx in python does not accept this. To solve this problem we gave every vertex a unique value and sorted by that corresponding value. For example the vertex (3,4,5) is given the value 345 if we work over $(\mathbb{Z}/8\mathbb{Z})^3$. For this we need to consider over which $\mathbb{Z}/n\mathbb{Z}$ we work, since for $n \in \{11, \dots, 100\}$ (10,4) should get the value 1004. Otherwise, vertex (1,11) would have the same value as vertex (11,1). By using this order double checking possibilities is excluded in the code.

3.2 Packaging problem

Baumert et al. [3] did research on the problem for $\alpha(C^d_{n,2})$ using a d-dimensional packaging problem of the $n \times n \times \cdots \times n$ torus with d-dimensional packages with sides of length 2. Now finding the maximum number of packages that can be placed gives the same problem as finding $\alpha(C^d_{n,2})$. This is so since the set consisting of the same corner of every package forms an independent set in $C^d_{n,2}$. Also the other way around every independent set in $C^d_{n,2}$ is changed by making every vertex in the set into a package by expanding in every direction by 2.

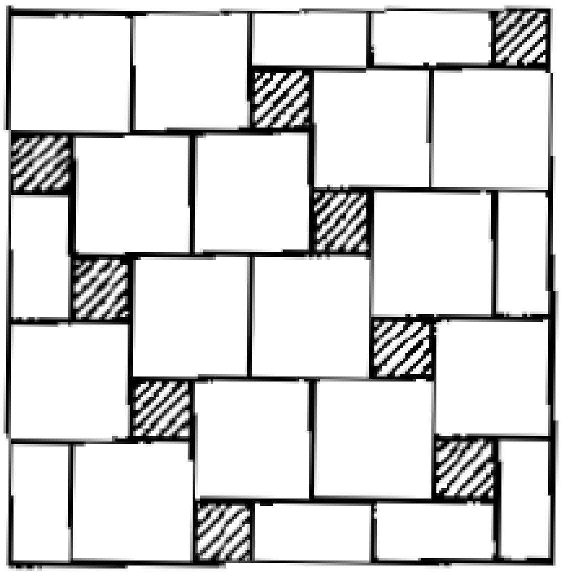


Figure 3.1: A packaging for the torus created by $C^2_{9,2}$ [3].

Baumert showed that we can expand a packaging in $C^d_{n,2}$ to a packaging in $C^d_{n+2,2}$. The process for this is shown in Figure 3.2. For this a cell in the n^d-torus is taken with coordinate $(a_1, \dots, a_d)$. Now first take the hyperplane of the torus where the first coordinate is set to a_1 and replace this hyperplane by 3 times itself on top of each other. This hyperplane contains packages which are split in half. When replacing this hyperplane by 3 times itself then 1 of these halved packages is attached to the other halved package from which it was detached. The other two halved packages form a new package together. This process will be done for every coordinate and this will provide a packaging in the $(n+2)^d$-torus.

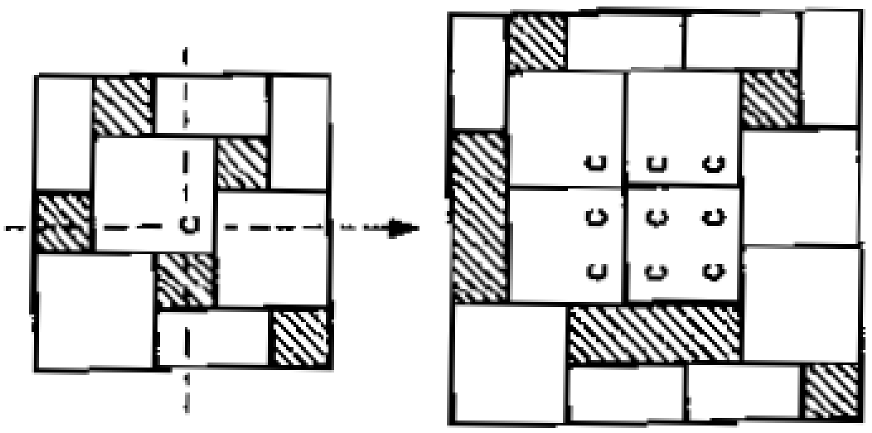


Figure 3.2: expansion of $C^2_{5,2}$ to $C^2_{7,2}$ by expanding a point u [3].

Theorem 3.1. *given a packaging of size N_n in the n^d-torus. Then there exists a packaging of size N_{n+2} in the $(n+2)^d$-torus, where*

$$N_{n+2} \geq N_n((n+2)/n)^d.$$

Proof. The expansion from figure 3.2 can be applied for every cell in the n^d-torus. For these n^d possible expansions every cell is copied the same amount of times, since the process is completely symmetric. From this follows that every cell is now $(n+2)^d$ times in the n^d packagings of the $(n+2)^d$-tori. From this directly follows that for every package we got $(n+2)^d$ packages. There are now $N_n(n+2)^d$ packages in the n^d packagings thus on average there are $N_n((n+2)/n)^d$ packages, which means that there has to be a $(n+2)^d$-torus with at least this amount of packages. □

Baumert also showed that this bound could be improved by a small value. When using this expansion seen from a empty cell then this expansion forms a empty $3^d - cube$ which can be filled with an extra package. The number of empty cells in a package is equal to $n^d - 2^d N_n$ which can be added to the known packagings from Theorem 3.1 and thus follows:

$$\begin{aligned} N_{n+2} &\geq (n^d - 2^d N_n + N_n(n+2)^d)/n^d \\ &= 1 + N_n(((n+2)^d - 2^d)/n^d). \end{aligned}$$

Since this lower bound can be applied for every packaging it surely works also for an optimal packaging. It thus follows that there is a lower bound for $\alpha(C^d_{n,2})$ given by:

$$\alpha(C^d_{n,2}) \geq 1 + \alpha(C^d_{n-2,2})((n^d - 2^d)/(n-2)^d).$$

Lemma 3.2.
$$\alpha(C^d_{n,2}) \le (n/2)\alpha(C^{d-1}_{n,2}).$$

Proof. This follows from the fact that the percentage of empty cells can not decrease when adding a dimension. This is since adding a dimension can be seen as adding hyperplanes on top of each other. Now this will give maximally the same percentage if we use for every hyperplane an optimal packaging of the $n^{d-1} - torus$. Now such an optimal packaging of a higher dimension has n hyperplanes but every package for the $n^d - torus$ is contained in 2 hyperplanes thus $n/2$ more packages at most. □

For all n uneven it is known that $\alpha(C_{n,2}) = (n-1)/2$ and thus by induction follows that:

$$\alpha(C^d_{n,2}) \le (n^d - n^{d-1})/2^d.$$

This upper bound assumes that for every multiplication with $n/2$ the answer is a whole integer. It is possible that this is not true, but every solution has to be integer. It follows that we can create a better upper bound by using Lemma 3.2 step by step. For example for $n = 7$ with $d = 3$ we get by induction $\alpha(C^3_{7,2}) \le (7^3 - 7^2)/2^3 = 36.75$. But if it is done step by step then we get with $\alpha(C_{7,2}) = (7-1)/2 = 3$.

$$\alpha(C^2_{7,2}) \le (7/2) * 3 = 10.5$$

$$\alpha(C^3_{7,2}) \le (7/2) * 10 = 35.$$

Theorem 3.3. *$\alpha(C^d_{n,2}) = (n^d - n^{d-1})/2^d$ when $n = \ell 2^d + 1$*

Proof. the upper bound follows directly by induction on Lemma 3.2. The lower bound is proven by showing a packaging that works, which is given for $\ell = 1$ by:

$$\{(v_1, v_2, \dots, v_d) \in (\mathbb{Z}/n\mathbb{Z})^d : v_d = 2v_1 + 4v_2 + \cdots + 2^{d-1}v_{d-1}\}.$$

packages are here the expansion of the coordinates in every direction by 2. Consider that 2 packages v and w coincide at a cell then $(v_1 - w_1, v_2 - w_2, \dots, v_{d-1} - w_{d-1}) \in \{-1, 0, 1\}^{d-1}$. When all are 1 then $v_d - w_d = 2^d - 2 < 2^d - 1$ and thus there is no need to calculate with modulo. This instantly gives us that $v_d - w_d \notin \{-1, 1\}$ since all values are even. Also it can only be 0 when all values are zero since for $p \in \mathbb{N}$ we got $2^p > \sum_{i<p} 2^i$. Thus all packages are independent. For $\ell > 1$ a packaging is given by:

$$\{(v_1 + 2j, v_2, \dots, v_d) \in (\mathbb{Z}/n\mathbb{Z})^d : v_d = 2v_1 + 4v_2 + \cdots + 2^{d-1}v_{d-1},\ j \in \{0, \ell - 1\}.$$

The number of packages that are given by this are ℓn^{d-1}. Now by rewriting $n = \ell 2^d + 1$ we get $\ell = (n-1)/2^d$. Thus the number of packages is $n^{d-1}(n-1)/2^d = (n^d - n^{d-1})/2^d$ □

With this Baumert proved that for $n = \ell 2^d + 1$ we got a known maximum packaging. A package created by the last theorem is created in such a way that expanding it is made much easier. For now consider packages as the coordinates as seen in Theorem 3.3. For this every slice of the same dimension contains the exact same amount of packages. This follows directly from that in the construction every slice of dimension 1 contains exactly ℓ packages. Thus every slice of dimension $p \le d$ contains ℓn^{p-1} packages. By using this Codenotti et al. [5] found how to create a expansion packaging for $n = \ell 2^d + 1$.

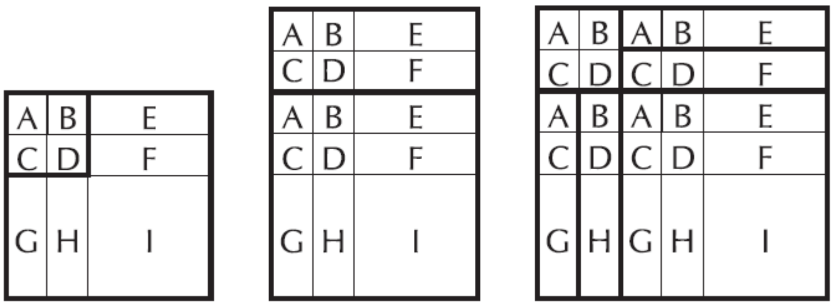


Figure 3.3: expansion of a square [5].

Theorem 3.4. *$\alpha(C^d_{n+i,2}) \geq (i/2)^d + \ell\frac{(n+i)^d - i^d}{n}$ when $n = \ell 2^d + 1$ and i even.*

Proof. Packages are here seen as the cells and not the 2^d cubes. As starting package take the set of coordinates provided by Theorem 3.3. Take C_i as the cut of the torus created by the slices were coordinate i is 0 or 1. In Figure 3.3 a way of expending is shown, which works for expending every direction step for step with C_i. 2 times C_i after each other works since the slices were coordinate i is 0 and 1 can be next to each other thus directly follows that 0,1,0,1 after each other also works. Now instead of expending the torus step for step, we will expend every direction at the same time. First add C_i in every direction i, which only adds the places where 1 direction is expended. Now we still need to consider the places were more directions are expended simultaneously. These expansions in q directions are filled by the intersection of q different C_i where i are the directions. It is known that the expansion in 1 direction had 2 options namely coordinate i is 0 or 1, Thus for the expansion in q directions there are 2^q options by setting all these q coordinates to 0 or 1. Since all slices of the same dimension contain the same amount of packages it is possible to consider the packages for every expansion in q directions together. The amount of expansions in q directions is then given by $\binom{d}{q}$. Now it follows that we add $2^q\binom{d}{q}$ slices were q coordinates are set. Thus a slice of dimension $p = d - q$ gets replicated $2^{d-p}\binom{d}{p}$ times. As said before every slice of dimension $p \geq 1$ has $\ell n^{p-1} = n^p(\ell/n)$ packages. Furthermore, the start packaging had ℓn^{d-1} packages and we can add 1 package in the cube created by the expansion in all directions with the intersection of all C_i. This then gives:

$$\begin{aligned}\alpha(C^d_{n+2,2}) &\geq 1 + \ell n^{d-1} + \ell/n(2^1\binom{d}{1}n^{d-1} + 2^2\binom{d}{2}n^{d-2} + \cdots + 2^{d-1}\binom{d}{d-1}n^1) \\ &= 1 + \ell n^{d-1} + \ell/n((n+2)^d - n^d - 2^d) \\ &= 1 + \ell/n((n+2)^d - 2^d).\end{aligned}$$

When this expansion has been done for $i/2$ times then two things change. A slice of dimension p now gets replicated $i^{d-p}\binom{d}{p}$ times. Also we can add $(i/2)^d$ more package in the cube created by the intersection of all C_i Thus now we get for i is even:

$$\alpha(C^d_{n+i,2}) \geq (i/2)^d + \ell/n((n+i)^d - i^d).$$

□

Codenotti et al. [5] also found a packaging for a higher dimension when using the packages created by $n = \ell 2^d + 1$.

Theorem 3.5.

$$\alpha(C_{n,2}^{d+1}) \geq (n^{d+1} - n^d)/2^{d+1} - n^{d-1}/2.$$

By using the last 2 theorems Codenotti also found a possible packaging for the combination of a higher dimension with expansion. For this he found:

$$\alpha(C_{n+i,2}^{d+1}) \geq (\frac{n+i}{2})^{d+1}\frac{n-1}{n} + \frac{1}{n}(\frac{i}{2})^{d+1} + 2^{d-1}i^d - \frac{n+1^d}{2}.$$

By using all information mentioned combined with some known lower bounds the lower bounds in Table 3.1 were found for $\alpha(C_{n,2}^d)$

n\d	3	4	5	6	$\Theta(C_{n,2})$
5	10	25	50	125	2.2361
7	33	108	343	1101	3.2237
9	81	324	1458	6561	4.3267
11	148	761	3996	21904	5.2896
13	247	1531	9633	61009	6.2743
15	382	2770	19864	145924	7.2558
17	578	4913	39304	334084	8.3721
19	807	7666	68994	651610	9.3571
21	1092	11441	114660	1201305	10.3423

Table 3.1: Lower bounds that are found for $\alpha(C_{n,2}^d)$, with a lower bound for the Shannon capacity found by taking the best d for every n [5].

This packaging problem can easily be changed such that it works for $\alpha(C_{n,k}^d)$. This is done by using packages of size k^d instead of the 2^d packages. Some of the earlier mentioned theorems and lemmas are easily changed into versions such that answers are found for $\alpha(C_{n,k}^d)$. Theorem 3.1 and Lemma 3.2 still work when changing the 2 by a k. For Theorem 3.1, instead of replacing hyperplanes by 3 times itself change this into $k+1$ times itself. Now it follows that

$$N_{n+k} \geq N_n((n+k)/n)^d$$

and

$$\alpha(C_{n,k}^d) \leq (n/k)\alpha(C_{n,k}^{d-1}).$$

Theorem 3.6. $\alpha(C_{n,k}^d) = (n^d - an^{d-1})/k^d$ *when* $n = \ell k^d + a$, *with* $a \in 1, 2, \ldots, k-1$.

Proof. The upper bound follows directly by $\alpha(C_{n,k}^d) \leq (n/k)\alpha(C_{n,k}^{d-1})$ where $\alpha(C_{n,k}) = (n-a)/k$. The lower bound is proven by the following package

$$\{(v_1, v_2, \ldots, v_d) \in (\mathbb{Z}/n\mathbb{Z})^d : v_d = kv_1 + k^2v_2 + \cdots + k^{d-1}v_{d-1}\}.$$

consider that 2 packages v and w coincide at a cell then for sure
$(v_1 - w_1, v_2 - w_2, \ldots, v_{d-1} - w_{d-1}) \in \{-k+1, \ldots, 0, \ldots, k-1\}^{d-1}$. When all are $k-1$

then $v_d - w_d = k^d - k < k^d - k + 1$ and thus there is no need to calculate with modulo. This instantly gives that $v_d - w_d \notin \{-k+1, \dots, -1, 1, \dots, k-1\}$ since all are multiples of k. Since $k^p > \sum_{i<p}(k-1)k^i$ it can only be 0 when all values are zero, thus all packages are independent. For $\ell > 1$ a packaging is given by:

$$\{(v_1 + kj, v_2, \dots, v_d) \in (\mathbb{Z}/n\mathbb{Z})^d : v_d = kv_1 + k^2v_2 + \dots + k^{d-1}v_{d-1},\ j \in \{0, \ell - 1\}.$$

The number of packages that are given by this are ℓn^{d-1}. Now by rewriting $n = \ell k^d + a$ we get $\ell = (n-a)/k^d$ and Thus the number of packages is $n^{d-1}(n-a)/k^d = (n^d - an^{d-1})/k^d$.

□

Theorem 3.7. *$\alpha(C^d_{n+i,k}) \geq (i/k)^d + \ell\frac{(n+i)^d - i^d}{n}$ when $n = \ell k^d + a$ and i a multiple of k.*

Proof. The same as Theorem 3.4, but now the cuts C_i contain the coordinates 0 to $k-1$ and we do i/k expansions. □

With these functions there is a problem, since we need that $n = \ell k^d + a$. This means that if $k = 3$ then for dimension 3 we already got at least $n = 3^3 + 1 = 28$. However, by also using already known values Jurkiewicz et al. [6] found the values in Table 3.2

n\k	2	3	4	5	6	7	8
4	8	1	1	1	1	1	1
5	10	1	1	1	1	1	1
6	27	8	1	1	1	1	1
7	33	8	1	1	1	1	1
8	64	12	8	1	1	1	1
9	81	27	8	1	1	1	1
10	125	30	10	8	1	1	1
11	148	36	13	8	1	1	1
12	216	64	27	8	8	1	1
13	247	69	27	10	8	1	1
14	343	79	33	14	8	8	1
15	382	125	36	27	10	8	1
16	512	133	64	27	12	8	8
17	578	149	64	30	14	9	8
18	729	216	81	36	27	10	8
19	807	224	82	36	27	10	9
20	1000	247	125	64	30	14	10

Table 3.2: Lower bounds that are found for $\alpha(C^3_{n,k})$ [6].

3.3 Solutions with a special characteristic

Definition 3.8. *(Graph homomorphism) Let G, H be graphs. A* graph homomorphism *f from G to H written as $f : G \to H$, is a function from $V(G)$ to $V(H)$ such that if*

$(u, v) \in E(G)$ then $(f(u), f(v)) \in E(H)$. If there exists such a function then we write $G \to H$.

Theorem 3.9. *Let G, H be graphs such that $\overline{G} \to \overline{H}$. Then $\alpha(G^d) \leq \alpha(H^d)$ for all $d \in \mathbb{N}$.*

Theorem 3.9 was proven by Bondy & Hell [8] and they showed that $\overline{C_{n_1,k_1}} \to \overline{C_{n_2,k_2}}$ if and only if $n_1/k_1 \leq n_2/k_2$. Thus it follows that for $n_1/k_1 \leq n_2/k_2$ we got $\alpha(C^d_{n_1,k_1}) \leq \alpha(C^d_{n_2,k_2})$. Thus it is possible to make a figure with lower bounds for the Shannon capacity against their known n/k values as a step function.

Polak & Schrijver[4] created a special set on the values q,d and n with $q < n$:

$$S(n, d, q) = \{t(1, q, \ldots, q^{d-1}) \bmod n : t \in \mathbb{Z}/n\mathbb{Z}\}.$$

The set will be used as S when no value is fixed. This set has a special characteristic that it almost uses every value the same amount of times. This is something nice to have since we want optimal use of the whole space and thus also optimal usage of all numbers. This can be seen in the optimal solution for $C^2_{5,2}$ where all numbers are used twice. In the set S we use all numbers exactly the same amount of times when $\gcd(q, n) = 1$. However, we also accept $\gcd(q, n) > 1$ since this also produced many answers.

Lemma 3.10. *given the set S is independent in $C^d_{n,k}$ then*

$$\Theta(C_{n,k}) \geq \sqrt[d]{n}.$$

Proof. Since S is an independent set in $C^d_{n,k}$ it directly follows that $\Theta(C_{n,k}) \geq \sqrt[d]{|S|}$. Here we got that $|S| = n$ when every $t \in \mathbb{Z}/n\mathbb{Z}$ creates a unique vertex. This directly follows from the first coordinate of the vertex which is equal to t and thus $|S| = n$. □

Lemma 3.11. *The set S is independent in $C^d_{n,k}$ when for all $t \in \mathbb{Z}/n\mathbb{Z}$*

$$\exists i \in \{0, 1, \ldots, d-1\} : k \leq tq^i \bmod n \leq n - k$$

Proof. It is known that a set A is independent in $C^d_{n,k}$ when for all $v, w \in A$

$$\exists i \in \{0, 1, \ldots, d-1\} : k \leq (v_i - w_i) \bmod n \leq n - k$$

Given 2 vertices $v, w \in S$ such that v is constructed with $t = a$ and w is constructed with $t = b$. Now the difference of these 2 vertices is given by the vertex u constructed with $t = (a - b) \bmod n$ and thus $u \in S$. Now follows that $(v_i - w_i) \bmod n = u_i$ and thus follows the theorem by the construction of the set S. □

For the set S independent in $C^d_{n,k}$ only the largest k is needed. This is so since then n/k is as small as possible and by Theorem 3.9 all larger values for n/k follow with the same lower bound.

For t and $-t$ the same maximum value for k follows since only the distance matters. Now by counting mod$n$ it follows that t and $n - t$ give the same k. Thus instead of checking all $n - 1$ elements (S without 0 element), it is possible to check half of the elements. Also when searching for a largest value k only $t \in [1, k - 1]$ has to be checked, since the first value is equal to t and thus larger or equal to k for $t \geq k$.

Polak & Schrijver [4] used this set S to search for lower bounds of $\alpha(C^d_{n,2})$. For this the set $S(n', d, q)$ was used to find independent sets for $\alpha(C^d_{n',k})$. Here $n' = |S|$ is chosen between the best upper and lower bound of $\alpha(C^d_{n,2})$ since n' will become the new lower bound. Before was said that $\alpha(C^d_{n,2}) \geq \alpha(C^d_{n',k})$ when $n'/k \leq n/2$ and thus follows that $k \geq 2\frac{n'}{n}$ is needed to find a new lower bound for $\alpha(C^d_{n,2})$. Since for k the lowest value has the highest chance of working, it follows that only $k = \lceil 2\frac{n_2}{n_1} \rceil$ has to be checked. By checking all possibilities for $n \leq 15$ and $d \leq 5$ Polak & Schrijver found the new lower bound $\alpha(C^5_{11,2}) \geq 4009$. However this is not a new lower bound for $\Theta(C_{11,2})$ since there is a better lower bound for $d = 3$.

Instead of using $k \geq 2\frac{n'}{n}$ Polak & Schrijver decided to also allow values a bit smaller. This would then give a set which is not independent in $C^d_{n,2}$, but by removing a few points it could become independent. By searching for this, Polak & Schrijver [4] found for $C^5_{7,2}$ the feasible set $S(382, 5, 7)$ for $C^5_{382,108}$. Since $382/108 > 7/2$ this will not directly result in a lower bound of 382 for $\alpha(C^5_{7,2})$. However, Polak & Schrijver used the following steps to create a lower bound.

1. Add the value $(40, 123, 40, 123, 40)$ to all elements of $S(382, 5, 7)$. This is a optimized value, which gives the best result.
2. Take $S' = \{\lfloor i/54.5 \rfloor : i \in S\}$. Now for $v \in S'$ then $v \in [0, 6]^5$ and thus $S' \subseteq V(C^5_{7,2})$.
3. Remove all elements, for which $k = 2$ does not work. This will result into a independent set of size 327 in $C^5_{7,2}$.
4. Add new points by exhaustive searching, From which 40 vertices can be added. Now a independent set of size 367 in $C^5_{7,2}$ is found.

For finding solutions using the set S for $\alpha(C^d_{n,k})$ a code was written. This code searches for solutions in dimension d with $n \in [2^d, p^d]$. Here $p \leq 10$ is chosen such that the code generates answers quite fast with at most $n/k = 10$. Using these limitations the code follows the next steps, where $x[-1]$ means the last added x value in the x, y coordinates.

1. Add the value $(\lceil p \rceil, \lceil p \rceil)$ to x, y coordinates and set n as $\lceil p^d \rceil$.
2. Decrease n by 1 and set k as $\lfloor n/x[-1] + 1 \rfloor$. Make the set of integers $Q = \{2, 3, \ldots, \lfloor n/2 \rfloor\}$ and set q as the first element in Q.
3. Create the set $S(n, d, q)$ and check if k works for this set, if it doesn't remove q from Q. If Q is empty go to step 5.
4. Set q as the next element in Q and repeat step 3. If there is no next number in Q then increase k by 1, set q as the first element in Q and return to step 3
5. If $k > \lfloor n/x[-1] + 1 \rfloor$ then add $(n/(k-1), \sqrt[d]{n})$ to x, y coordinates.
6. If $n > 2^d$ return to step 2. Otherwise, print the x, y coordinates in a step plot.

In step 1 we add the point $(\lceil p \rceil, \lceil p \rceil)$. This is a optimal reference point that has $k = \lceil p \rceil^{d-1}$ as feasible solution for the set $S(\lceil p \rceil^d, d, \lceil p \rceil)$

In step 2 we decrease n by 1 and k is chosen in such a way that n/k is smaller than $x[-1]$. A set Q is made so that it contains all viable values for q. It contains only the first $\lfloor n/2 \rfloor$ since $(-q)^i$ has the same distance from 0 as q^i. 0 and 1 are excluded since $\forall i, j \in \mathbb{N}$ with $q \in 0, 1$ we got $q^i = q^j$.

In step 3 the set $S(n, d, q)$ is created and we remove the value q from Q if it doesn't work for the given k. Since this means that q will also not work for $k + 1$ and higher.

In Step 4 we switch to the next value in Q. If there is no next value then we switch back to the first value in Q, which exists since we checked if Q was empty in step 3. It is known that all these values in Q work for k, thus now we check if some of these also work for $k + 1$ by going back to step 3.

After some time Q is empty which means there is no q such that S works with k. Then we go to step 5, where we check if $k - 1$ is a value that was checked. If this is true then there was a q such that S worked with $k - 1$ and thus adds a new coordinate at $(n/(k - 1), \sqrt[d]{n})$.

In step 6 we return to step 2 and check the next value for n as long as $n \geq 2^d$. If this is not true then it would only be interesting if a k value is found such that $n/k < 2$. But for this a largest independent set is known to be of size 1.

Using this code, data sets were created for fixed values d with corresponding maximum values for n such that $n/k \leq 10$ and such that the calculation time isn't too long. These data sets are plotted in figure 3.4 with intervals of 2. By observing Figure 3.4a it can be seen that $d = 6$ and higher almost do not provide any new information for lower bounds on the Shannon capacity. This might be different for higher n/k, but then the calculation time gets too long.

After this it was decided that maybe restricting to a single value q was too extreme thus the next set was created, with $\forall q_i \in q$ we got $1 < q_i < \lfloor n/2 \rfloor$.

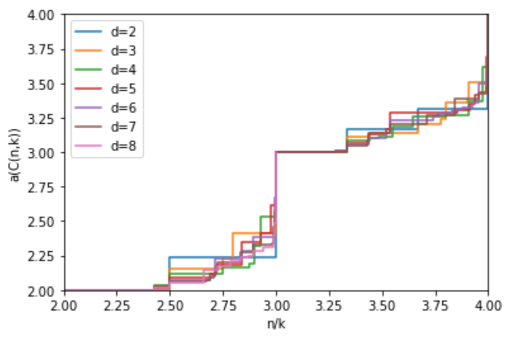
$$T(n, d, q) = \{t(1, q_1, q_2, \ldots, q_{d-1}) \bmod n : t \in \mathbb{Z}/n\mathbb{Z}\}.$$

The code for this problem runs in basically the same way as for the set S, but now the set Q exists of combinations of i elements instead of a single element q. This resulted in the plots shown in Figure 3.5.

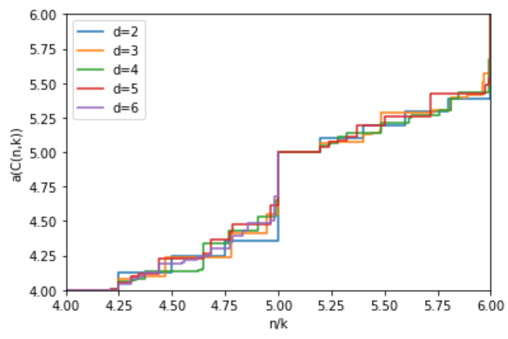
In figure 3.5 it is clear that for these data plots the value d influences the number of effective results. It can be seen that the values for $d = 4$ are always equal or higher to the values for $d = 2$. Assume $T(n, d)$ as a working set for k created by using n and d. Then a working set $T(n^2, 2d)$ exists for $k_{new} = nk$. If $T(n, d)$ worked for k with $(q_1, q_2, \ldots, q_{d-1})$ then $T(n^2, 2d)$ works for $k_{new} = nk$ with the $2d - 1$ q-values $(q_1, q_2, \ldots, qd - 1, n, nq_1, nq_2, \ldots, nq_{d-1})$.

In figure 3.6 it can be seen that by using the set T much better results are obtained. It can be seen that for $n/k \in [2, 6]$ the set T is most of the time above the set S, while for $n/k \in [6, 10]$ the set T creates almost only better independent sets than set S, when set S doesn't increase in size for a while.

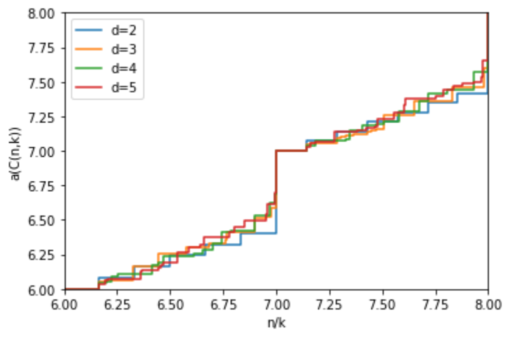
Polak & Schrijver's tactic of also accepting higher values of n/k to find values for lower n/k has been tried. However, for this exhaustive searching is needed and thus only low values for n and k can be used. It follows that only k is small works (2,3 or 4). Sadly except for the already known $C_{7,2}^5 \geq 367$ there was no new lower bound found for this.



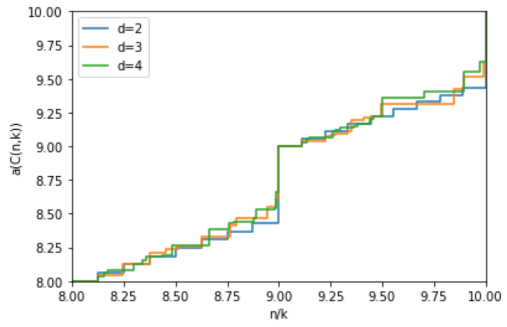
(a) interval [2,4]



(b) interval [4,6]

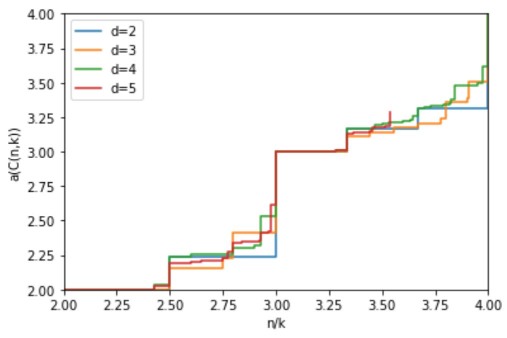


(c) interval [6,8]

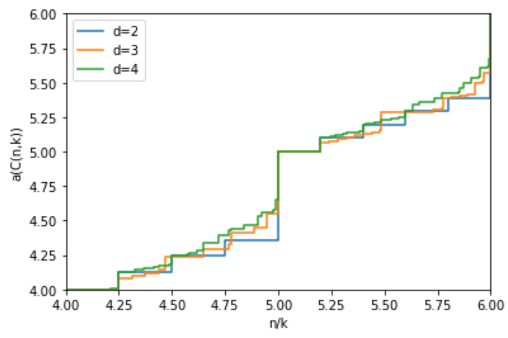


(d) interval [8,10]

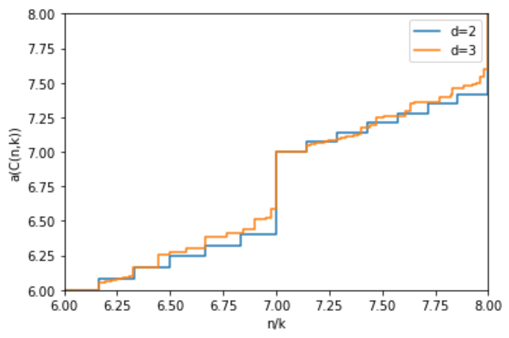
Figure 3.4: Subplots for $n/k \in [2, 10]$ against new lower bounds for $\Theta(C_{n,k})$, with lines corresponding to fixed value for d with the set S.



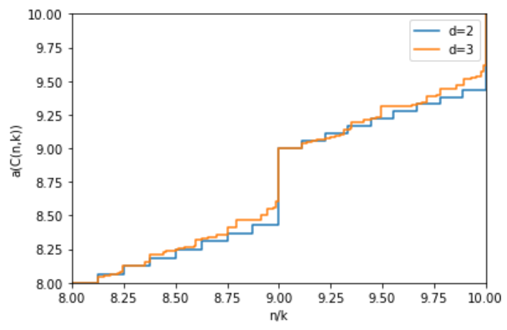
(a) interval [2,4]



(b) interval [4,6]

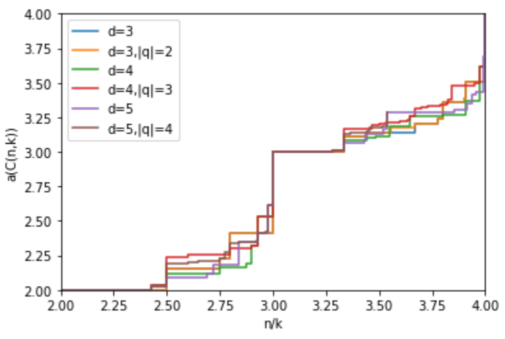


(c) interval [6,8]

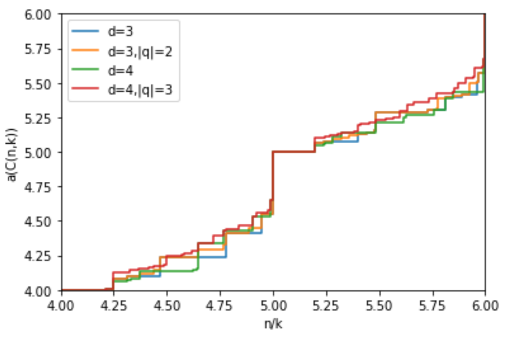


(d) interval [8,10]

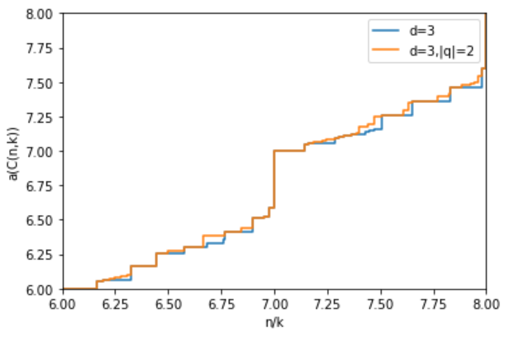
Figure 3.5: Subplots for $n/k \in [2, 10]$ against new lower bounds for $\Theta(C_{n,k})$, with lines corresponding to fixed value for d with the set T.



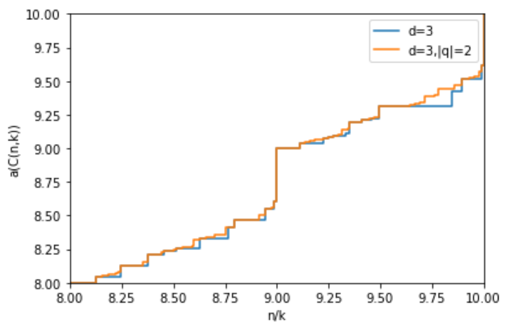
(a) interval [2,4]



(b) interval [4,6]



(c) interval [6,8]



(d) interval [8,10]

Figure 3.6: subplots for $n/k \in [2, 10]$ against new lower bounds for $\Theta(C_{n,k})$, with lines corresponding to fixed value for d and their version with a single element q and multiple q-values.

Chapter 4

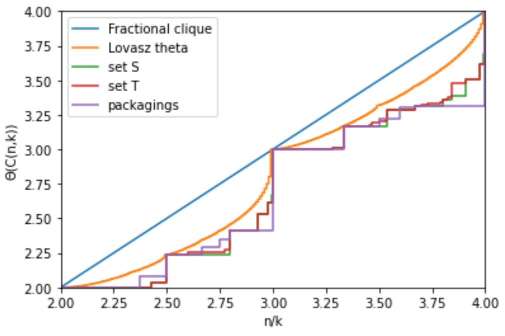
Findings and conclusions

In Chapter 2 some upper bounds for the Shannon capacity on a graph were given and proven. Also for these upper bounds the exact values for $C_{n,k}$ were given. In Chapter 3 we showed multiple ways to calculate lower bounds for the Shannon capacity by searching for solutions for $\alpha(C^d_{n,k})$. All these upper and lower bounds are combined in Figure 4.1.

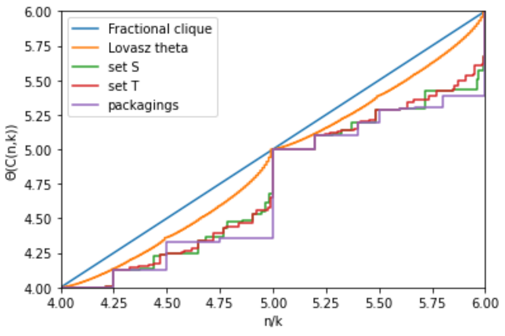
For the upper bounds it is clear that the Lovász theta function always gives a closer upper bound. However, for high n/k the difference between the Fractional clique cover and the Lovász Theta function, seems to become very small.

For the lower bounds created by the set S and set T from Chapter 3.3 only the best values are taken from every checked value d. The lower bound named packagings in the figure is created by known solutions combined with Theorem 3.7. The values on the halves are all known solutions and almost all other values follow from Theorem 3.7. When comparing the sets S and T, it is found that for some n/k the set S is better while for other values the set T is better. The set T is an expansion of the set S so the answers should always be better. This is not so since for the set S more d values can be checked, because it computes answers much faster. When comparing the answers created by the sets S and T and the answers from the packaging problem there is a clear difference. For $n/k \in [a, a+0.5]$ with $a \in \mathbb{N}$ we find mostly better answers by using the packagings, while for $n/k \in (a+0.5, a+1]$ with $a \in \mathbb{N}$ the answers created by the sets are almost always better.

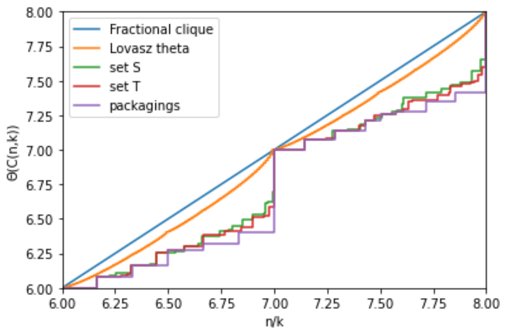
If we consider the difference between the best upper and lower bounds, it is noticed that the smallest differences can be found at $n/k \in \{2\frac{1}{2}, 3\frac{1}{3}, \ldots, p\frac{1}{p}\}$ with $p \in \mathbb{N}$. All these points are found for $d = 2$ with Theorem 3.6 and are all optimal solutions for $d = 2$. In addition to these points the other optimal points created by Theorem 3.6 with $n = \ell k^d + a$ decrease fast in effectiveness for higher values for a. From this can be concluded that these points must be observed for higher dimensions to obtain better results or that the Lovász theta function gets worse for these n/k. Using the two sets it can be seen that for higher dimensions much better sets can be created, but these are not optimal. Thus it seems that higher dimensions have more impact for n/k values the closer they get to a next integer.



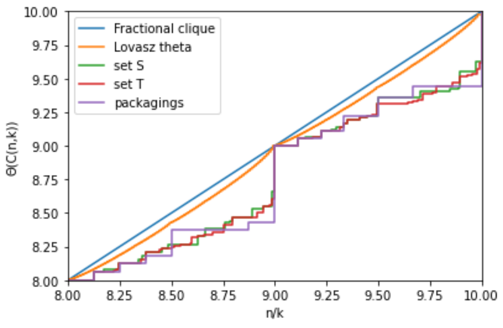
(a) interval [2,4]



(b) interval [4,6]



(c) interval [6,8]



(d) interval [8,10]

Figure 4.1: Subplots for $n/k \in [2, 10]$ against lower and upper bounds for the Shannon capacity of $C_{n,k}$. The Fractional clique and Lovász as upper bounds of the Shannon capacity. The answers provided by the set S and newly created set T from Chapter 3.3 as lower bounds. Packagings contains known solutions combined with answers provided by Chapter 3.2 as lower bounds.

For the sets created in Chapter 3.3 there can still be done research on the correlation between good results and their corresponding q, k and n values. If a correlation can be found then it would be possible to exclude a lot of possibilities, since the biggest disadvantage from these sets is that the time for finding answers increases fast for higher n values.

For the packaging problem it would be interesting if Theorem 3.5 could be recreated for k values with $n = \ell k^d + a$. Since the biggest disadvantage from the packaging problem is that higher dimensions give answer for higher n/k values. For the interval [2,10] we can at most search dimension 3 for k is 2 or 3.

Bibliography

[1] Shannon, C. (1956). The zero error capacity of a noisy channel. IRE Transactions on Information Theory, 2(3), 8-19.

[2] Lovász, L. (1979). On the Shannon capacity of a graph. IEEE Transactions on Information theory, 25(1), 1-7.

[3] Baumert, L. D., McEliece, R. J., Rodemich, E., Rumsey, H. C., Stanley, R., & Taylor, H. (1971). A combinatorial packing problem. Computers in algebra and number theory, 4.

[4] Polak, S. C., & Schrijver, A. (2019). New lower bound on the Shannon capacity of C7 from circular graphs. Information Processing Letters, 143, 37-40.

[5] Codenotti, B., Gerace, I., & Resta, G. (2003). Some remarks on the Shannon capacity of odd cycles. Ars Combinatoria, 66, 243-258.

[6] Jurkiewicz, M., Kubale, M., & Turowski, K. (2014). Some lower bounds on the Shannon capacity. Journal of Applied Computer Science, 22(2), 31-42.

[7] Grissinger, M. (2017). Misidentification of Alphanumeric Symbols Plays a Role in Errors. Pharmacy and Therapeutics, 42(10), 604.

[8] Bondy, J. A., & Hell, P. (1990). A note on the star chromatic number. Journal of Graph Theory, 14(4), 479-482.

[9] Bachoc, C., Pêcher, A., & Thiéry, A. (2013). On the theta number of powers of cycle graphs. Combinatorica, 33(3), 297-317.

Appendix A

Python code

Python code for solving the set S in Chapter 3.3

```
import numpy as np
import math
xy=np.array([[6,6]])#start coordinate
d=4#dimension
working=[]
test=[]
test1=[]
for n in range(1296,15,-1):#range of n values
    for p in range(2,int(n/2+1)):
        for q in range(p+1,int(n/2+1)):
            for r in range(q+1,int(n/2+1)):
                test.append([p,q,r]) #set of all possibilities
    for k in range(math.floor(n/xy[-1][0]+1),4000):
        for q in test:
            f=0
            for i in range(1,k):
                working=[(i*q[0])%n,(i*q[1])%n,(i*q[2])%n]
                #working needs to be changed for more q's
                true=0
                for a in working:
                    if a>=k and n-k>=a:
                        true=1
                if true==0:
                    f=1
                    break
            if f==0:
                k1=k
                n1=n
                test1.append(q)
        if test1==[]:
            if k>math.floor(n/xy[-1][0]+1):
                xy=np.append(xy,[[n1/k1,n1**(1/d)]],axis=0)
            break
        test=list(test1)
        test1=[]
    test=[]
xy=np.flip(xy,0)
```

Code for finding solutions by exhaustive searching

```
import networkx as nx

def neighbours(n,k): # gives neighbours for vertex of dimension 1
    tot=[]
    for i in range(n):
        neigh=[]
        for i in range(i-k+1,i+k):
            neigh.append(i%n)
        tot.append(neigh)
    return tot


def network(vertices,edges): #makes the graph
    G = nx.Graph()
    for i in range(len(vertices)):
        a=0
        for j in range(len(vertices[i])):
            a+=vertices[i][j]*10**j
        G.add_node(a)
    for i in range(len(vertices)):
        a=0
        for j in range(len(vertices[i])):
            a+=vertices[i][j]*10**j
        for j in range(len(edges[i])):
            b=0
            for k in range(len(edges[i][j])):
               b+=edges[i][j][k]*10**k
            G.add_edge(a, b)
    return G

def full(n,d):#gives all vertices
    tot=[]
    for a in range(n**d):
        b=[]
        for i in range(d):
            b.append(a//(n**i)%n)
        tot.append(b)
    return tot

def edges(n,tot):#gives all edges of vertices
    edg=[]
    for a in tot:
        edg1=[]
        for b in tot:
            true=0
            for i in range(len(a)):
                if b[i] not in n[a[i]]:
                    true=1
            if true==0 and a not in edg1 and a!=b:
                edg1.append(b)
        edg.append(edg1)
    return edg

def deep(G,b,se,final,k,d):#exhaustive searching
```

```
    if len(list(G.nodes))==0:
        if len(se)>=len(final):
            final=list(se)
        return(final)
    for a in G.nodes:
        if a<=b:
            nodes=list(G.neighbors(a))
            d=0
            for i in nodes:
                if i>=b:
                    break
                d+=1
            if nx.density(G.subgraph(nodes[d:]))==1:
                break
        if a>b:
            se.append(a)
            G1=nx.Graph(G)
            e=list(G.neighbors(a))
            e.append(a)
            G1.remove_nodes_from(e)
            final=deep(G1,a,se,final,k,d)
            se.pop()
    return(final)

n=10# set for C(n,k)^d
d=2
k=3
tot=full(n,d)
neigh=neighbours(n,k)
edg=edges(neigh,tot)
G=network(tot,edg)
e=[0]
for i in G.neighbors(0):
    e.append(i)
G.remove_nodes_from(e)
final=deep(G,0,[0],[],k,d) # set counting largest independent set
```