8-2012

A set of tournaments with many Hamiltonian cycles

Hayato Ushijima-mwesigwa
Clemson University, hushiji@clemson.edu

A set of tournaments with many Hamiltonian cycles

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mathematics

---

by
Hayato Ushijima-Mwesigwa
August 2012

Accepted by:
Dr. Neil Calkin, Committee Chair
Dr. Beth Novick
Dr. Daniel Warner

Abstract

For a random tournament on 3^n vertices, the expected number of Hamiltonian cycles is known to be $(3^n-1)!/2^{3^n}$. Let T_1 denote a tournament of three vertices v_1, v_2, v_3. Let the orientation be such that there are directed edges from $v_1$to v_2 , from v_2 to v_3 and from v_3 to v_1. Construct a tournament T_i by making three copies of T_{i-1}, T'_{i-1}, T''_{i-1} and T'''_{i-1}. Let each vertex in T'_{i-1} have directed edges to all vertices in T''_{i-1}, similarly place directed edges from each vertex in T''_{i-1} to all vertices in T'''_{i-1} and from T'''_{i-1} to T'_{i-1}.

In this thesis, we shall study this family of highly symmetric tournaments. In particular we shall present two different algorithms to calculate the number of Hamiltonian cycles in these tournaments and compare them with the expected number and with known bounds for random tournaments. This thesis is motivated by the question of the maximum number of Hamiltonian cycles a tournament can have.

Acknowledgments

I would like to acknowledge my committee chair Dr. Calkin for his advice and guidance throughout this project. I would also like to give a special thanks to my committee members Dr. Novick and Dr. Warner for much needed guidance. Lastly, I would like to acknowledge fellow graduate students, Nate Black and Thilo Strauss for their useful questions and comments along the way.

Contents

Chapter 1

Introduction

1.1 Basic definitions

We first present some basic definitions. We mostly follow the treatment in [2].

Definition A *directed graph (digraph)* is a pair (V,E) where V is the set of *vertices (or nodes, or points)* and $E \subset V \times V$ is a set of *edges*, which we regard as ordered pairs of vertices. In the edge (u,v), we refer to u as the initial vertex and v as the terminal vertex. We call (u,v) an edge from u to v (see figure 1). Sometimes we denote (u,v) simply by uv. If $u=v$, then the corresponding edge is called a *loop*. In this thesis, none of the digraphs we present contain loops.

Figure 1

A *(directed) path* is a non-empty directed graph $P=(V,E)$ of the form

$$V = \{x_0, x_1, \dots, x_k\}\ E = \{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\},$$

where the x_i are all distinct. The vertices x_0 and x_k are called its *end vertices*. We often refer to a path by the natural sequence of its vertices, writing, say, $P = x_0x_1 \dots x_k$ and calling P a path *from* x_0 to x_k. If $P = x_0x_1 \dots x_{k-1}$ is a path and $k \geq 3$, then the graph $C := P + x_{k-1}x_0$ is called a *cycle*.

A *Hamiltonian path* of a directed graph G is a path containing every vertex in G. Similarly, a *Hamiltonian cycle* is a cycle containing every vertex in G.

A *tournament* T is a directed graph in which for every $u \neq v$ exactly one of the edges (u,v) and (v,u) is in E. We can think of T as the outcomes of a sports event in which pairs of teams play once and there are no ties, only wins and losses. The name tournament derives from a round-robin tournament.

1.2 Previous work

If we construct a tournament T by independently choosing the edge between vertices u and v to be (u,v) and (v,u) with equal probability, then we can use the linearity of expectation to compute the expected number of Hamiltonian cycles (similarly Hamiltonian paths) in a random tournament. Since the number of cycles is non-negative, there must exists a tournament with at least these many cycles (paths). Szele [7] in 1943 was the first to use this observation and showed that

$$P(n) \geq n!/2^{n-1}, \tag{1.1}$$

where $P(n)$ denotes the maximum possible number of Hamiltonian paths in a tournament on n vertices and the right-hand side of the inequality is the expected number.

Szele's proof is considered to be the first application of the probabilistic method in combinatorics. The same argument shows

$$C(n) \geq (n-1)!/2^{n}, \tag{1.2}$$

where $C(n)$ denotes the maximum possible number of Hamiltonian cycles in a tournament on n vertices and the right-hand side of the inequality is the expected number of Hamiltonian cycles.

In the same paper Szele established an upper bound on $P(n)$ by showing that

$$P(n) \leq c_1 \cdot n!/2^{\frac{3}{4}n}, \tag{1.3}$$

where c_1 is a positive constant independent of n, and conjectured that

$$\lim_{n\to\infty} \left(\frac{P(n)}{n!}\right)^{\frac{1}{n}} = \frac{1}{2}.$$

Later, Alon [1] proved this conjecture and improved the upper bound to

$$P(n) \leq c_2 \cdot n^{\frac{3}{2}} n!/2^{n-1},$$

where $c_2 > 0$ is independent of n.

Kahn and Friedgut [3] later improved this upper bound further by showing that for any $\xi < 2(1-\exp[\sqrt{3/4}-1]) \approx 0.2507\ldots$,

$$C(n) < O(n^{1/2-\xi} n! 2^{-n}), \tag{1.4}$$

and (consequently)

$$P(n) < O(n^{3/2-\xi} n! 2^{-n}). \tag{1.5}$$

These are the best known upper bounds of $C(n)$ and $P(n)$ and we note that these bounds beat the expected number by a factor that is dependent on n. Wormald [8] conjectured that in fact $C(n) \approx 2.855958 \cdot (n-1)!/2^n$.

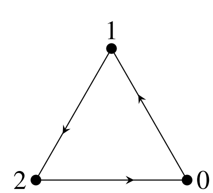
In this thesis, we will restrict our attention to a particular tournament, T_n on 3^n vertices constructed in a manner which we might hope to give a large number of Hamiltonian cycles. We will give an approximate algorithm and an exact algorithm to count the number of Hamiltonian cycles in this tournament, and compute the approximate and exact counts for $n \leq 6$.

1.3 T_n

We consider a sequence of tournaments $T_0, T_1, T_2, \ldots$. We'll construct the tournament T_n recursively as follows:

T_0 is:

T_1 is:



and T_n is a tournament on 3^n vertices consisting of three copies of T_{n-1}, placed in a triangle, with edges between the T_{n-1}'s oriented in a counterclockwise fashion as in figure 2, where T'_{n-1}, T''_{n-1} and T'''_{n-1} represent

$$
\begin{tikzcd}
 & T''_{n-1} \arrow[dl, Rightarrow] & \\
T'''_{n-1} \arrow[rr, Rightarrow] & & T'_{n-1} \arrow[ul, Rightarrow]
\end{tikzcd}
$$

Figure 2. T'_{n-1}, T''_{n-1} and T'''_{n-1} are 3 copies of T_{n-1}

the three copies of T_{n-1} and the $\Longrightarrow' s$ represent the directions of the edges between the copies.

More formally, and for purposes of computation, T_n will have the vertex set $0, 1, \ldots, 3^n - 1$ in base 3; to construct it, we take 3 copies of T_{n-1}, replace each vertex v in T_{n-1} by $3v$, $3v+1$ and $3v+2$ in the three copies respectively. Then the direction of an edge uv, where u and v are from different copies of T_{n-1} is determined by their final ternary digits in such a way that the direction is from 0 to 1, 1 to 2 and 2 to 0.

Chapter 2

Exact counting Algorithm

Let $H(T_n)$ denote the number of Hamiltonian cycles in T_n. In this section we present some theorems and propositions leading to an exact counting algorithm to compute $H(T_n)$.

Definition A *path cover* of a directed graph G is a set of disjoint directed paths in G which together contain all the vertices of G. An *m-path cover* is a path cover of cardinality m.

By definition, the 1-path covers are the Hamiltonian paths. We will first reduce the problem of computing $H(T_n)$ to the problem of counting the number of m-path covers for $1 \leq m \leq 3^{n-1}$ in T_{n-1}. We make this reduction by making the following observation:For $n \geq 1$, let T'_{n-1}, T''_{n-1} and T'''_{n-1} be the three copies of T_{n-1} from which T_n was constructed. Take any Hamiltonian cycle C, of T_n, and consider C restricted to T'_{n-1}, T''_{n-1} and T'''_{n-1}. Since a Hamiltonian cycle on T_n contains every vertex in T'_{n-1}, T''_{n-1} and T'''_{n-1} exactly once, C restricted to T'_{n-1} would form a k-path cover of T'_{n-1} for some $1 \leq k \leq 3^{n-1}$. Similarly for T''_{n-1} and T'''_{n-1}. Now if C restricted to T'_{n-1} induces a k-path cover for a fixed k, then it must be the case that C also induces a k-path cover in T''_{n-1} and T'''_{n-1}. It is easy to show that the number of ways of joining the k-path covers to form a Hamiltonian cycle is $k!^3/k$. Thus if P_k^{n-1} denotes the number of k-path covers of T_{n-1}, then the number of Hamiltonian cycles of T_n that induce k-path covers in T'_{n-1}, T''_{n-1} and T'''_{n-1} is $(k! \cdot P_k^{n-1})^3/k$.

Thus, if P_i^{n-1} is the number of i-path covers of T_{n-1}, for $1 \leq i \leq 3^{n-1}$, then

$$H(T_n) = \sum_{i=1}^{3^{n-1}} \frac{(i! \cdot P_i^{n-1})^3}{i}, \tag{2.1}$$

where the i^{th} term in the sum counts the number of choices for i-path covers for T'_{n-1}, T''_{n-1} and T'''_{n-1}, and the number of ways of joining them to create a Hamiltonian cycle.

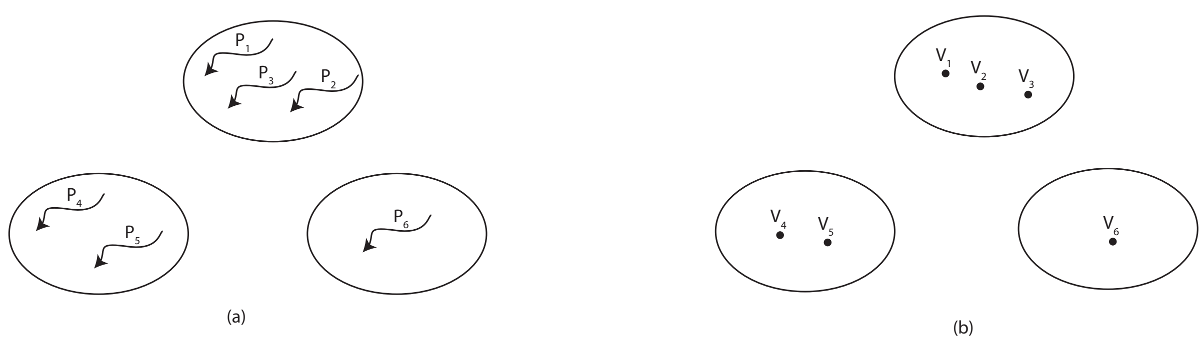
We now focus on calculating P_i^{n-1} for $1 \leq i \leq 3^{n-1}$. For T_1, we can easily count P_i^1 for $1 \leq i \leq 3$ and get $P_1^1 = 3$ (Hamiltonian paths in T_1), $P_2^1 = 3$ and $P_3^1 = 1$ (trivial paths). We will compute P_i^{n-1} for $1 \leq i \leq 3^{n-1}$ recursively, so for now we will assume P_i^{n-2} is known for $1 \leq i \leq 3^{n-2}$.

For each i, j and k- path cover of T'_{n-2}, T''_{n-2} and T'''_{n-2} respectively, we wish to know how many ways they can be joined to give a path cover of T_{n-1}. If we add a directed edge from an end vertex of a path in the path cover of T'_{n-2}, to an end vertex in a path in the path cover of T''_{n-2} to obtain a new path, we form a $(i+j+k-1)$-path cover of T_{n-1}. Thus our problem for counting $H(T_n)$ reduces to the following problem:

Problem 2.0.1. *For each i, j and k- path cover of T'_{n-2}, T''_{n-2} and T'''_{n-2}, how many ways can we connect them with m edges to form a $(i+j+k-m)$-path cover of T_{n-1}?*

Notice that trivially, these i, j and k-path covers together form an $(i+j+k)$-path cover of T_{n-1}. Thus if we consider all ways of creating disjoint paths by adding m edges between the i, j and k-path covers without creating cycles for all $1 \leq i,j,k \leq 3^{n-2}$ and $0 \leq m \leq 3^{n-1}$, we would have in fact constructed all path covers of T_{n-1}

For simplicity, we can view the i disjoint paths in an i-path cover as a set of i independent vertices, as shown in the figure below.



Disjoint paths in (a) correspond to independent vertices in (b)

This can can also be viewed as contracting each disjoint path in T'_{n-2}, T''_{n-2} and T'''_{n-2} to a singleton. Then problem 2.0.1 is equivalent to the following problem. Here m_1, m_2 and m_3 replace i, j and k respectively:

Let M_1, M_2 and M_3 be three sets of vertices with $|M_1| = m_1$, $|M_2| = m_2$ and $|M_3| = m_3$ and let G be a digraph with vertex set $V(G) = M_1 \cup M_2 \cup M_3$, and let the edges in G be such that each vertex in M_1 can only have a directed edge to any vertex in M_2, any vertex in M_2 can only have a directed edge to any in M_3 and any in M_3 can only have a directed edge to any in M_1.

Problem 2.0.2. *How many ways can we add w edges to G such that G*

1. *contains no cycles, and*
2. *for every vertex v in G, $|d^+(v)| \leq 1$ and $|d^-(v)| \leq 1$, where $|d^+(v)|$ is the out degree of v and $|d^-(v)|$ is the in degree of v.*

Let $\mathscr{F}_{w,m_1,m_2,m_3}$ be the set of all digraphs satisfying (1) and (2) formed by adding exactly w edges to G and let $F(w,m_1,m_2,m_3) := |\mathscr{F}_{w,m_1,m_2,m_3}|$. Then P_i^{n-1} is given by:

$$P_i^{n-1} = \sum_{\substack{m_1,m_2,m_3 \\ 1 \leq m_1,m_2,m_3 \leq 3^{n-2},\ m_1+m_2+m_3 \geq i}} P_{m_1}^{n-2} P_{m_2}^{n-2} P_{m_3}^{n-2} \cdot F(m_1+m_2+m_3-i,\ m_1,m_2,m_3). \tag{2.2}$$

2.1 Computing $F(w,m_1,m_2,m_3)$

In this section we answer problem 2.0.2 to get an expression for $F(w,m_1,m_2,m_3)$. Consider a "relaxation" of this problem without the first restriction, i.e., we allow cycles. Call the resulting set of graphs formed by adding w edges in all possible ways $\mathscr{E}_{w,m_1,m_2,m_3}$ and let $E(w,m_1,m_2,m_3) := |\mathscr{E}_{w,m_1,m_2,m_3}|$, then

$$E(w,m_1,m_2,m_3) = \sum_{a+b+c=w} \binom{m_1}{a}\binom{m_2}{a}\binom{m_2}{b}\binom{m_3}{b}\binom{m_3}{c}\binom{m_1}{c} \cdot a!b!c!. \tag{2.3}$$

The above expression for $E(w,m_1,m_2,m_3)$ is derived as follows: In order to satisfy the indegree and out degree constraint (2), choose a vertices from M_1 and M_2 and a bijection between them, b vertices from M_2 and b from M_3 and a bijection between them and lastly c vetices from M_3 and M_1 and a bijection between them subject to $a+b+c=w$.

Clearly $\mathscr{F}_{w,m_1,m_2,m_3} \subseteq \mathscr{E}_{w,m_1,m_2,m_3}$ and $\mathscr{E}_{w,m_1,m_2,m_3} \setminus \mathscr{F}_{w,m_1,m_2,m_3}$ is the set of all graphs in $\mathscr{E}_{w,m_1,m_2,m_3}$ that contain at least one cycle, thus,

$$F(w,m_1,m_2,m_3) = E(w,m_1,m_2,m_3) - |\mathscr{E}_{w,m_1,m_2,m_3} \setminus \mathscr{F}_{w,m_1,m_2,m_3}|. \tag{2.4}$$

Proposition 2.1.1.

$$F(w,m_1,m_2,m_3) = E(w,m_1,m_2,m_3) - m_1 m_2 m_3 \cdot E(w-3,m_1-1,m_2-1,m_3-1)$$

For the remainder of this section, we present a detailed proof for proposition 2.1.1. We prove this by applying the "inclusion-exclusion principle" and state and prove a theorem about integer partitions which we use to simplify the expression we get from the inclusion-exclusion principle.

Theorem 2.1.2. *(*Inclusion-Exclusion principle*)*

For finite sets $A_0, A_1, \ldots, A_m$. The following identity holds;

$$\left|\bigcup_{i=0}^{m} A_i\right| = \sum_{i=0}^{m} |A_i| - \sum_{\substack{i,j \\ 1\le i<j\le m}} \left|A_i \cap A_j\right| + \sum_{\substack{i,j,k \\ 1\le i<j<k\le m}} \left|A_i \cap A_j \cap A_k\right| - \cdots + (-1)^{m-1} |A_1 \cap \cdots \cap A_m|.$$

The above theorem can be proved by induction. The details of the proof can be found in [4].

If we add w edges to the independent sets, cycles of different lengths can be formed. We call a cycle of length k an *k-cycle*. Since we have 3 independent sets, the cycles formed will have lengths a multiple of 3. Let $X_1, X_2, \ldots, X_v$ be all the possible individual cycles that can be formed by adding w edges, and let $A_{X_1}, A_{X_2}, \ldots, A_{X_m}$ be the set of graphs in $\mathscr{E}_{w,m_1,m_2,m_3}$ which contain $X_1, X_2, \ldots, X_v$ respectively. Then we are interested in calculating $|\bigcup_{i=1}^{v} A_{X_i}|$, the number of graphs with at least one cycle. Thus $F(w, m_1, m_2, m_3)$ is now expressed as:

$$F(w, m_1, m_2, m_3) \;=\; E(w, m_1, m_2, m_3) \;-\; |\bigcup_{i=1}^{v} A_{X_i}|, \tag{2.5}$$

where $|\bigcup_{i=1}^{v} A_{X_i}|$ is obtained by directly applying the inclusion-exclusion principle, i.e.

$$\left|\bigcup_{i=1}^{v} A_{X_i}\right| = \sum_{i=1}^{v} |A_{X_i}| - \sum_{\substack{i,j \\ 1\le i<j\le v}} \left|A_{X_i} \cap A_{X_j}\right| + \sum_{\substack{i,j,k \\ 1\le i<j<k\le v}} \left|A_{X_i} \cap A_{X_j} \cap A_{X_i}\right| - \cdots + (-1)^{v-1} |A_{X_i} \cap \cdots \cap A_{X_v}|.$$

Note that the degree constraints imply that all cycles formed are disjoint. Thus if two cycle X_i, X_j are not disjoint then $A_{X_i} \cap A_{X_j} = \emptyset$.

Let σ_j be the number of ways of getting a $3j$-cycle, σ_{j_1,j_2} be the number of ways of getting a $3j_1$-cycle and $3j_2$-cycle concurrently and in general let $\sigma_{j_1,j_2,\ldots,j_v}$ be the number of ways of getting a $3j_1$-cycle, $3j_2$-cycle, $\ldots$, and $3j_v$-cycle concurrently for all $1 \le j_i \le \min(\lfloor \frac{n}{3} \rfloor, m_1.m_2, m_3)$.

Consider again the cycles $X_1, X_2, \ldots, X_v$. If we re-order these cycles by their lengths such that $X_1, \ldots, X_{v_1}$ are all the 3-cycles, $X_{v_1+1}, \ldots, X_{v_2}$ are all the 6-cycles, $\ldots$, and $X_{v_{j-1}+1}, \ldots, X_v$ are all the $3j$-cycles. Then,

$$\sum_{i=1}^{v_1} |A_{X_i}| = \sigma_1 \cdot E(w-3, m_1-1, m_2-1, m_3-1), \tag{2.6}$$

where equation (2.6) can be thought of as: Add 3 of the w edges to the independent sets in such a way that you create a 3-cycle, which can be done in σ_1 ways. For each of these, add the $w-3$ remaining edges to the independent sets M_1, M_2 and M_3 with current size m_1-1, m_2-1, m_3-1 respectively, which can be done in $E(w-3, m_1-1, m_2-1, m_3-1)$ ways.

Let $\Delta := \min(\lfloor \frac{n}{3} \rfloor, m_1.m_2, m_3)$, then 3Δ is the largest possible cycle length. Using the same argument as above, we get:

$$\begin{aligned}
\sum_{i=v_1+1}^{v_2} |A_{X_i}| &= \sigma_2 \cdot E(w-6, m_1-2, m_2-2, m_3-2), \\
\sum_{i=v_2+1}^{v_3} |A_{X_i}| &= \sigma_3 \cdot E(w-9, m_1-3, m_2-3, m_3-3), \\
&\vdots \\
\sum_{i=v_{\Delta-1}+1}^{v} |A_{X_i}| &= \sigma_j \cdot E(w-3\Delta, m_1-\Delta, m_2-\Delta, m_3-\Delta).
\end{aligned}$$

Thus,

$$\sum_{i=1}^{v} |A_{X_i}| = \sum_{i=1}^{\Delta} \sigma_i \cdot E(w-3i, m_1-i, m_2-i, m_3-i). \tag{2.7}$$

A similar argument gives

$$\begin{aligned}
\sum_{\substack{i,j \\ 1\le i<j\le v}} \left|A_{X_i} \cap A_{X_j}\right| &= \sum_{1\le i<j\le\Delta} \sigma_{i,j} \cdot E(w-3(i+j), m_1-(i+j)j, m_2-(i+j), m_3-(i+j)), \\
\sum_{\substack{i,j,k \\ 1\le i<j<k\le v}} \left|A_{X_i} \cap A_{X_j} \cap A_{X_i}\right| &= \sum_{1\le i<j<k\le\Delta} \sigma_{i,j,k} \cdot E(w-3(i+j+k), m_1-(i+j+k), m_2-3, m_3-(i+j+k)), \\
&\vdots \\
\sum_{\substack{i_1,\ldots,i_\Delta \\ 1\le i_1<i_2<\cdots<i_\Delta\le v}} \left|A_{X_{i_1}} \cap \cdots \cap A_{X_{i_\Delta}}\right| &= |A_{X_1} \cap \cdots \cap A_{X_\Delta}| \\
&= \sigma_{\underbrace{1,1,\ldots,1}_{\Delta\ times}} \cdot E(w-3\Delta, m_1-\Delta, m_2-\Delta, m_3-\Delta),
\end{aligned}$$

and the rest of the terms from the inclusion-principle are zero, i.e.,

$$\sum_{k=\Delta+1}^{v} \sum_{\substack{i_1,\ldots,i_k \\ 1\le i_1<i_2<\cdots<i_k\le v}} (-1)^{k-1} \left|A_{X_{i_1}} \cap \cdots \cap A_{X_{i_k}}\right| = 0.$$

Consequently, equation (2.5) can be re-written as:

$$
\begin{aligned}
F(w,m_1,m_2,m_3) = E(w,m_1,m_2,m_3) \quad - \quad & \sum_{i=1}^{\Delta} \sigma_i \cdot E(w-3i, m_1-i, m_2-i, m_3-i) \\
& + \sum_{1\le i<j\le\Delta} \sigma_{i,j} \cdot E(w-3(i+j), m_1-(i+j)j, m_2-(i+j), m_3-(i+j)) \\
& - \sum_{1\le i<j<k\le\Delta} \sigma_{i,j,k} \cdot E(w-3(i+j+k), m_1-(i+j+k), m_2-3, m_3-(i+j+k)) \\
& + \\
& \vdots \\
& + (-1)^{\Delta} \sigma_{\underbrace{1,1,\ldots,1}_{\Delta \ times}} \cdot E(w-3\Delta, m_1-\Delta, m_2-\Delta, m_3-\Delta).
\end{aligned}
$$

We now focus on getting an expression for $\sigma_{j_1,\ldots,j_k}$, the number of ways of getting cycles of length $3j_1,\ldots,3j_k$ concurrently.

Definition For any positive integer n, a *partition* of n, λ, is a non-increasing sequence of positive integers $\lambda_1,\lambda_2,\ldots,\lambda_k$ whose sum is n. Each λ_i is called a *part* of the partition. We let the function $p(\lambda)$ denote the number of parts of λ and $\Lambda(n)$ denote the set of partitions of all positive integers less than or equal to n.

The subscripts of $\sigma_{j_1,\ldots,j_k}$ consist of all nonnegative integers such that $j_1+\cdots+j_k \le \Delta$. These are precisely all partitions of positve integers less or equal to Δ. Thus $F(w,m_1,m_2,m_3)$ can be written as:

$$
F(w,m_1,m_2,m_3) = E(w,m_1,m_2,m_3) + \sum_{\lambda\in\Lambda(\Delta)} (-1)^{p(\lambda)} \sigma_\lambda E(w-3|\lambda|, m_1-|\lambda|, m_2-|\lambda|, m_3-|\lambda|) \quad (2.8)
$$

where $|\lambda|$ is the sum of the parts in λ.

2.1.1 Computing σ_λ

For a partition λ let i_3 be the number of 1's in λ, i_6 the number of 2's, $\ldots$, i_{3k} the number of k's in λ where $k \ge 1$. Then for any λ, σ_λ can be rewritten as:

$$
\sigma_\lambda = \sigma_{\underbrace{1,1,\ldots,1}_{i_3 \ times},\underbrace{2,2,\ldots,2}_{i_6 \ times},\ldots,\underbrace{\Delta,\Delta,\ldots,\Delta}_{i_{3\Delta} \ times}}. \quad (2.9)
$$

where

$$|\lambda| := i_3 + 2i_6 + 3i_9 + \cdots + \Delta \cdot i_{3\Delta} \leq \Delta, \tag{2.10}$$

with

$$i_3, i_6, \ldots, i_{3\Delta} \geq 0.$$

Inequality (2.10) represents the number of vertices that are used from each independent set. Since a 3-cycle uses 1 vertex each, a 6-cycle uses 2 vertices each and so on, the coefficients follow.

The representation (2.9) is useful to compute σ_λ systematically in the following way: First we count the number of ways of choosing vertices from the sets M_1, M_2 and M_3 to get i_3 3-cycles, i_6 6-cycles and so on. Then we multiply this by the number of ways the chosen vertices can be joined to form their respective cycles. We will first focus on getting an expression for counting the number of ways of choosing these vertices.

2.1.2 Choosing vertex sets to form $i_3, \ldots, i_{3k}, \ldots, i_{3\Delta}$, $3k$-cycles

As stated before, we first count the number of ways of choosing vertices from the sets M_1, M_2 and M_3 to get i_3 3-cycles, i_6 6-cycles and so on. We do this by first choosing the vertices that form the i_3 3 cycles, then from the remaining $m_1 - i_3$, $m_2 - i_3$ and $m_3 - i_3$ vertices in the sets M_1, M_2 and M_3 respectively, we choose vertices for the i_6 6-cycles. We repeat the process for all i_{3k}, for $1 \leq k \leq \Delta$. This argument gives the following expressions:

The number of ways of choosing vertices in M_1, M_2 and M_3 to form i_3 3-cycle concurrently is:

$$\frac{1}{i_3!} \cdot \prod_{j=1}^{3} \binom{m_j}{1} \cdot \binom{m_j - 1}{1} \cdot \cdots \cdot \binom{m_j - i_3 + 1}{1},$$

i.e., choose 1 vertex from each set i_3 times. We divide by $i_3!$ to distinguish between the chosen vertices. Expanding this expression we get:

$$\begin{aligned}
& \frac{1}{i_3!} \cdot \prod_{j=1}^{3} \underbrace{\frac{m_j!}{\cancel{(m_j - 1)!} \cdot 1!} \cdot \frac{\cancel{(m_j - 1)!}}{\cancel{(m_j - 2)!} \cdot 1!} \cdot \cdots \cdot \frac{\cancel{(m_j - i_3 + 1)!}}{(m_j - i_3) \cdot 1!}}_{i_3 \ times} \\
= \ & \frac{1}{i_3!} \cdot \prod_{j=1}^{3} \frac{1}{1!^{i_3}} \cdot \frac{m_j!}{(m_j - i_3)!} \\
= \ & \frac{1}{i_3!} \cdot \prod_{j=1}^{3} \frac{1}{1!^{i_3}} \cdot \binom{m_j}{i_3} \cdot i_3!.
\end{aligned} \tag{2.11}$$

Now that we have chosen the vertices for the i_3 3-cycles, from the remaining $m_j - i_3$ vertices of the sets M_j for $1 \leq j \leq 3$, the number of ways of choosing vertices to form i_6 3-cycle concurrently is after

$$\frac{1}{i_6!} \cdot \prod_{j=1}^{3} \binom{m_j - i_3}{2} \cdot \binom{m_j - i_3 - 2}{2} \cdot \dots \cdot \binom{m_j - i_3 - 2i_6 + 2}{2}.$$

A similar simplification as (2.11) gives:

$$\frac{1}{i_6!} \cdot \prod_{j=1}^{3} \frac{1}{2!^{i_6}} \cdot \binom{m_j - i_3}{2i_6} \cdot (2i_6)!. \tag{2.12}$$

We keep doing this up to i_Δ. Where we get,

$$\frac{1}{i_\Delta!} \cdot \prod_{j=1}^{3} \frac{1}{\Delta!^{i_{3\Delta}}} \cdot \binom{m_j - i_3 - 2i_6 - 3i_9 - \dots - \Delta i_{3\Delta}}{\Delta i_{3\Delta}} \cdot (\Delta i_{3\Delta})!. \tag{2.13}$$

We then multiply the expressions from (2.11) to (2.13), to get:

$$\begin{aligned} & \frac{1}{i_3! \cdot i_6! \cdot \dots \cdot i_{3\Delta}!} \cdot \prod_{j=1}^{3} \frac{i_3! \cdot (2i_6)! \cdot \dots \cdot (\Delta i_{3\Delta})!}{1!^{i_3} \cdot 2!^{i_6} \cdot \dots \cdot \Delta!^{i_{3\Delta}}} \cdot \binom{m_j}{i_3}\binom{m_j - i_3}{2i_6} \cdots \binom{m_j - i_3 - 2i_6 - 3i_9 - \dots - (\Delta-1)i_{3(\Delta-1)}}{\Delta i_{3\Delta}} \\ = \; & \frac{1}{i_3! \cdot i_6! \cdot \dots \cdot i_{3\Delta}!} \cdot \prod_{j=1}^{3} \frac{1}{1!^{i_3} \cdot 2!^{i_6} \cdot \dots \cdot \Delta!^{i_{3\Delta}}} \cdot \binom{m_j}{i_3 + 2i_6 + 3i_9 + \dots + \Delta \cdot i_{3\Delta}} \cdot (i_3 + 2i_6 + 3i_9 + \dots + \Delta \cdot i_{3\Delta})! \\ = \; & \frac{1}{i_3! \cdot i_6! \cdot \dots \cdot i_{3\Delta}!} \cdot \prod_{j=1}^{3} \frac{1}{1!^{i_3} \cdot 2!^{i_6} \cdot \dots \cdot \Delta!^{i_{3\Delta}}} \cdot \binom{m_j}{|\lambda|} \cdot |\lambda|!. \end{aligned} \tag{2.14}$$

Expression (2.14) represents the number of ways of choosing the vertices in M_1, M_2 and M_3 to get i_3 3-cycles, i_6 6 cycles, ..., $i_{3\Delta}$ 3Δ cycles. Next we want to know how many ways these vertices can be connected to form the required cycles.

For any $k > 0$, the number of ways of connecting 3 sets of k independent vertices to form a $3k$ cycle is:

$$\frac{k!^3}{k}. \tag{2.15}$$

For any λ, we can view the chosen vertices for *each* of the i_3 3-cycles as 3 disjoint vertices with $\frac{1!^3}{1}$ ways of connecting them to form a 3-cycle. Then we can connect the chosen vertices for *all* of the i_3 3-cycles, in $\left(\frac{1!^3}{1}\right)^{i_3}$ ways. Similarly we connect the $2, 3, \dots, \Delta$ cycles in:

$$\left(\frac{2!^3}{2}\right)^{i_6}, \left(\frac{3!^3}{3}\right)^{i_9}, \dots, \left(\frac{\Delta!^3}{\Delta}\right)^{i_{3\Delta}} \tag{2.16}$$

ways.

We then get a beautiful expression for σ_λ:

$$\begin{aligned}
\sigma_\lambda &= \left\{\frac{1}{i_3!\cdot i_6!\cdot\dots\cdot i_{3\Delta!}}\prod_{j=1}^{3}\frac{1}{1!^{i_3}\cdot 2!^{i_6}\cdot\dots\cdot\Delta!^{i_{3\Delta}}}\binom{m_j}{|\lambda|}|\lambda|!\right\}\cdot\left(\frac{1!^3}{1}\right)^{i_3}\left(\frac{2!^3}{2}\right)^{i_6}\cdots\left(\frac{\Delta!^3}{\Delta}\right)^{i_{3\Delta}} \\
&= \frac{1}{i_3!\cdot i_6!\cdot\dots\cdot i_{3\Delta}!\cdot 1^{i_3}\cdot 2^{i_6}\cdot\dots\cdot\Delta^{i_{3\Delta}}}\prod_{j=1}^{3}\binom{m_j}{|\lambda|}|\lambda|! \\
&= \frac{1}{\prod_{j=1}^{\Delta} i_{3j}!\cdot\prod_{i=1}^{p(\lambda)}\lambda_i}\cdot\prod_{j=1}^{3}\binom{m_j}{|\lambda|}|\lambda|!. \qquad (2.17)
\end{aligned}$$

It follows that (2.8) can be rewritten as:

$$\begin{aligned}
F(w,m_1,m_2,m_3) &= E(w,m_1,m_2,m_3) \\
&+ \sum_{\lambda\in\Lambda(\Delta)}\frac{(-1)^{p(\lambda)}}{\prod_{j=1}^{\Delta} i_{3j}!\cdot\prod_{i=1}^{p(\lambda)}\lambda_i}\prod_{j=1}^{3}\binom{m_j}{|\lambda|}|\lambda|!\cdot E(w-3|\lambda|,m_1-|\lambda|,m_2-|\lambda|,m_3-|\lambda|).
\end{aligned} \qquad (2.18)$$

Theorem 2.1.3. *Let λ be a partition of a fixed integer n , $n \geq 2$.*

Then,

$$\sum_{\lambda}\frac{(-1)^{p(\lambda)}}{\prod_j i_{3j}!\cdot\prod_{i=1}^{p(\lambda)}\lambda_i} = 0,$$

where the sum is taken over all partitions λ of n and the product on the left side of the denominator is over all possible values of j.

We present a simple example to illustrate the above theorem. Let n = 5, then the 7 partitions of 5 and with the respective information are given in the table below.

λ	$\prod_{i=1}^{p(\lambda)} \lambda_i$	$\prod_j i_{3j}!$	$(-1)^{p(\lambda)}$
5	5	1!	-1
4,1	4	1!	1
3,2	6	1!	1
3,1,1	3	2!	-1
2,2,1	4	2!	-1
2,1,1,1	2	3!	1
1,1,1,1,1	1	5!	-1

Then it follows that,

$$-\frac{1}{5\cdot 1!}+\frac{1}{4\cdot 1!}+\frac{1}{6\cdot 1!}-\frac{1}{3\cdot 2!}-\frac{1}{4\cdot 2!}+\frac{1}{2\cdot 3!}-\frac{1}{5!}\;=\;0.$$

Proof. We prove the result using Faa di Bruno's formula. Faa di Bruno's formula is a generalization of the chain rule for higher derivatives. The general form of Faa di Bruno's formula is:

$$\frac{d^n}{dx^n}f(g(x))=\sum\frac{n!}{m_1!\,1!^{m_1}\,m_2!\,2!^{m_2}\,\cdots\,m_n!\,n!^{m_n}}\cdot f^{(m_1+\cdots+m_n)}(g(x))\cdot\prod_{j=1}^{n}\left(g^{(j)}(x)\right)^{m_j}$$

where the sum is over all n-tuples of nonnegative integers $(m_1,\ldots,m_n)$ satisfying the constraint,

$$1\cdot m_1+2\cdot m_2+3\cdot m_3+\cdots+n\cdot m_n=n.$$

In terms of the notation used in the theorem, this can be written as:

$$\frac{d^n}{dx^n}f(g(x))=\sum_{\lambda}\frac{1}{\prod\limits_{i=1}^{p(\lambda)}\lambda_i\cdot\prod\limits_{j}i_j!}\cdot f^{p(\lambda)}(g(x))\cdot g_{\lambda}(x)$$

where,

$$g_{\lambda}(x)=g^{(\lambda_1)}(x)\cdot g^{(\lambda_2)}(x)\cdot\;\cdots\;\cdot g^{(\lambda_\ell)}(x)$$

We want $g^{(\lambda_i)}(x)$ to give $(\lambda_i-1)!$. Thus if $g(x) = -\log(1-x)$, then this would imply that,

$$g^{(\lambda_i)}(x)=\frac{(\lambda_i-1)!}{(1-x)^{\lambda_i}}.$$

Similarly, If $f(y) = e^{-y}$, then $f^{p(\lambda)}(y) = (-1)^{p(\lambda)} f(y)$, thus $e^{(-g(x))} = e^{\log(1-x)} = 1-x$.

Hence,

$$\frac{d^n}{dx^n} f(g(x)) = \begin{cases} 1-x, & \text{if } n = 0, \\ -1, & \text{if } n = 1, \\ 0, & \text{if } n \geq 2, \end{cases}$$

□

From the theorem 2.1.3, it follows that the summands in equation (2.8) add up to zero except when λ is a partition of 1. In other words,

$$F(w, m_1, m_2, m_3) = E(w, m_1, m_2, m_3) - \sigma_1 \cdot E(w-3, m_1 - 1 m_2 - 1, m_3 - 1) \tag{2.19}$$

or equivalently

$$F(w, m_1, m_2, m_3) = E(w, m_1, m_2, m_3) - m_1 m_2 m_3 \cdot E(w-3, m_1 - 1 m_2 - 1, m_3 - 1).$$

which concludes the proof of proposition 2.1.1 and we now formally present the algorithm to compute $H(T_n)$, the number of Hamiltonian cycles in T_n, by recursively computing the i-path covers in T_{n-1}.

Algorithm 1 Algorithm to count the number of i-path covers and compute number of Hamitonian cycles in T_n

INPUT: List $\mathbf{P}^{n-2} = [P_1^{n-2}, P_2^{n-2}, \ldots, P_{3^{n-2}}^{n-2}]$, where $\mathbf{P}^{n-2}$ is a list of number of all path-covers of T_{n-2}
OUTPUT: The number of Hamiltonian cycles in T_n, and number of k-path covers for all k.

```
Start with P^{n-1} as a list of n-1 zeros
for all i in 1 to 3^{n-2} do
  for all j in i to 3^{n-2} do
    for all k in k to 3^{n-2} do
      v = i + j + k
      if i = j and j = k then
        P_v^{n-1}=P_v^{n-1} + P_i^{n-2} · P_j^{n-2} · P_k^{n-2}
        # path-covers before adding edges
        for all w in 1 to 2i + j do
          # for all edges w, to be added to the graph
          if v - e ≥ 0 then
            P_{v-w}^{n-1} = P_{v-w}^{n-1} + P_i^{n-2} · P_j^{n-2} · P_k^{n-2} · F(w, i, j, k)
          end if
        end for
      else if i = j or j = k then
        P_v^{n-1} = P_v^{n-1} + 3 · P_i^{n-2} · P_j^{n-2} · P_k^{n-2}
        # path-covers before adding edges
        for all w in 1 to 2i + j do
          P_{v-w}^{n-1} = P_{v-w}^{n-1} + 3 · P_i^{n-2} · P_j^{n-2} · P_k^{n-2} · F(w, i, j, k)
          # 3 ways of symmetry
        end for
      else
        P_v^{n-1} = P_v^{n-1} + 6 · ω
        # path-covers before adding edges
        for all e in 1 to 2i + j do
          P_{v-w}^{n-1} = P_{v-w}^{n-1} + 6 · P_i^{n-2} · P_j^{n-2} · P_k^{n-2} · F(w, i, j, k)
          # 6 ways of symmetry
        end for
      end if
    end for
  end for
end for
H(T_n) = Σ_{i=1}^{3^{n-1}} (i! · P_i^{n-1})^3 / i
```

Chapter 3

Approximation Algorithm

Definition [2] An *(undirected) graph* is a pair $G = (V, E)$ of sets such that $E \subset [V]^2$. The elements of V are the vertices (or nodes) of G, the elements of E are its edges. An *acyclic* graph, one not containing any cycles, is called a *forest*. A connected forest is called a *tree*. (Thus, a forest is a graph whose components are trees.) A *rooted tree* is a tree with a countable number of nodes, in which a particular node is distinguished from the others and called the *root*. The nodes of degree 1 are called the *leaves* of the tree, except if the node is the root.

Label the vertices of the tournament T_n as $1, 2, \ldots, 3^n$. Let T_n^* be a rooted tree whose nodes represent all possible paths and Hamiltonian cycles in T_n starting at fixed vertex 1. T_n^* can be defined as follows: Let the root of T_n^* represent vertex 1 of T_n, i.e. the starting vertex. Let the children of the root represent all paths of length 1 starting at vertex 1. One node u in T_n^* is a child of another v if it is the extension of the path represented by v by one edge to the new path or to a Hamiltonian cycle represented by u. Hence the nodes of T_n^* at depth k represent paths of length k in the tournament T_n and the leaves at depth 3^n represent the Hamiltonian cycles in T_n. The question of counting the number of Hamiltonian cycles in the tournament T_n reduces to counting the number of leaves in T_n^* at depth 3^n. It is easy to see that the size of T_n^* is very large even for small values of n.

Backtracking is a general algorithm for finding all (or some) solutions to some computational problem. It incrementally builds candidates to the solutions, abandoning each partial candidate c ("backtracks") as soon as it determines that c cannot possibly be completed to a valid solution, see [5]. It is a recursive method of building up a feasible solution to a combinatorial optimization problem one step at a time. A backtrack search is an *exhaustive search*, that is, all feasible solutions are considered, at least implicitly, so it will always find the optimal solution. The *state space* of a backtracking algorithm involves a tree. Estimating the size of this tree is useful in predicting how long a large backtrack search might be expected to take. Kreher and Stinson [6] presented an algorithm to estimate the size of the state space tree T for a backtracking algorithm without actually running the entire algorithm. Informally, their algorithm is as follows: For a tree T, $|T|$ is estimated by probing a *random* path $P = p_0 p_1 \ldots p_m$ where $p_i \in V(T)$ for $i = 0, 1, \ldots, m$, through T, where p_0 is the root and p_m is a leaf. As we follow this path, we compute the number of children c_i of p_i. Then the number of nodes in T at depth i according to the random path P is $c_0 c_1 \cdots c_{i-1}$. Thus the estimate of $|T|$ according to P is given by:

$$|T| \approx 1 + c_0 + c_0 c_1 + c_0 c_1 c_2 + \cdots + c_0 c_1 c_2 \cdots c_{m-1} \tag{3.1}$$

In particular, we can estimate the number of nodes at depth 3^n of T_n^* using Kreher and Stinson's algorithm thus estimating the number of Hamiltonian cycles of T_n . Let $H(P)$ be the estimate of the number of nodes at depth 3^n, with $P = p_0 p_1 \cdots p_m$ a random path in T_n^* from root p_0 to leaf p_m and c_i the number of children of p_i, then

$$H(P) = \begin{cases} c_0 c_1 \cdots c_{m-1}, & \text{if } m = 3^n \\ 0, & \text{otherwise.} \end{cases}$$

In order to increase the accuracy, several runs of $H(P)$ are computed and the average values of $H(P)$ are taken over the different runs. We implemented this using Sage and got estimates for $H(T_n)$ by computing $H(P)$ over a sample size of 100,000 for $n = 1, \ldots, 5$ and a sample size of 10,000 for $n = 6$. These results were particularly helpful in verifying the computational results we were getting while working on the exact algorithm. Note that this method can also be easily used in estimating the number of Hamiltonian cycles in general tournaments. We present the results in the next chapter and the implementation in Sage can be found in Appendix B.

Chapter 4

Computational Results

In this chapter we present the computational results giving the estimates and exact counts of the number of Hamiltonian cycles in T_n. We also present the number of Hamiltonian paths in T_n i.e., the number of 1-path covers of T_n since the exact algorithm computes them concurrently.

4.1 Approximate Counts for $H(T_n)$

We ran the approximation algorithm with sample size of 100,000 ten times and got the following results:

208.096000000000
208.250720000000
208.254240000000
205.009920000000
208.525280000000
206.546080000000
205.280800000000
204.090400000000
205.288960000000

Getting the average of the above results and rounding to the nearest integer,we can conclude that $H(T_2)$ is approximately 207 Hamiltonian cycles.

For T_3 with sample size 100,000 we get:

8.38936393504178e18
8.29415270322695e18
8.41085831064413e18
8.20069048677160e18
8.23054416986776e18
8.38901207085574e18
8.22982685280299e18
8.42540274654137e18
8.27677088387733e18
8.27121370347297e18

with an average of approximately 8.311e18 Hamiltonian cycles.

For T_4 with sample size 100,000 we get:

8.39212935331849e94
8.20984619093887e94
8.33860969614190e94
8.21100465493029e94
8.12149753319329e94
8.06273445583200e94
8.19511790498236e94
8.18968303414667e94
8.24921813852953e94
8.32078388331347e94

with an average of approximately 8.23e94 Hamiltonian cycles.

For T_5 with sample size 100,000 we get:

4.77309584702392e400
4.68917174160924e400
4.74988976385854e400
4.77817310624222e400
4.75087918890168e400
4.48785956506462e400
4.47040112900951e400
4.90979276677279e400
4.69740117978661e400
4.81677881362070e400

with an average of approximately 4.71e400 Hamiltonian cycles.

Lastly for T_6 with sample size 10,000 we get:

1.91468599948298e1550
2.05245624812883e1550
1.74356077128382e1550
1.95092667377627e1550
1.87486011438676e1550
1.98537038673843e1550
1.82301308326100e1550
2.00221518020148e1550
2.00730405281973e1550
2.03615287754156e1550

with an average of approximately 1.94e1550 Hamiltonian cycles.

4.2 Exact Counts for $H(T_n)$

The exact values of the number of Hamiltonian cycles $H(T_n)$ and Hamiltonian paths $P(T_n)$ in tournament T_n are given below. The numbers larger than 10^{19} are presented in scientific form rounded to 18 digits.

n	$H(T_n)$
1	1
2	207
3	8316362583640202859
4	8.243616097444882209e94
5	4.681945708027605746e400
6	1.95133590743535e1550

n	$P(T_n)$
1	3
2	3159
3	4.17382500592116e21
4	1.30121086168815e97
5	2.25541503737347e403
6	2.83662916923917e1553

4.2.1 Exact Vs. Approximate count

Lastly we present the table below that shows the approximate counts and exact counts of $H(T_n)$ side by side in scientific form rounded to the second decimal place for comparison purposes.

n	Approximate count	Exact count
1	1	1
2	207	207
3	8.311e18	8.312e18
4	8.23e94	8.24e94
5	4.71e400	4.68e400
6	1.94e1550	1.95e1550

Chapter 5

Conclusions and Discussion

Recall from chapter 1 that if m is the number of vertices in a tournament, then the expected number of Hamiltonian cycles $E(m)$, it has is $(m-1)!/2^m$ and that the known upper bound due to Kahn and Friedgut is $O(m^{1/2-\xi}m!2^{-m})$ with $\xi = 0.2507$. The table below shows $H(T_n)$, the number of Hamiltonian cycles in T_n, $E(3^n)$, the expected number of Hamiltonian cycles for a tournament on 3^n vertices, Kahn and Friedgut upper bound and the ratio of $H(T_n)$ to $E(3^n)$.

n	$H(T_n)$	$E(3^n)$	$O(3^{n\cdot(\frac{1}{2}-0.2507)}\cdot\frac{3^n!}{2^{3^n}})$	$\frac{H(T_n)}{E(3^n)}$
1	1	0.25	$O(0.9862)$	4
2	207	78.75	$O(1225.7)$	2.62857
3	8.31636258364020e18	3.00475553517495e18	$O(1.84e20)$	2.76773
4	8.24361609744488e94	2.96004336598080e94	$O(7.17e96)$	2.78496
5	4.681945708027605746e400	1.67846452947232e400	$O(1.60e403)$	2.78942
6	1.95133590743535e1550	6.99197412277854e1549	$O(2.63e1553)$	2.79082

From the table above we conclude $H(T_n)$ is at least $2\cdot E(3^n)$ and that T_n is a tournament with a greater number of Hamiltonian cycles than the expected number for a random tournament with the same number of vertices. More results would be useful to see, as n goes to infinity, how close this comes to $2.855958 * E(3^n)$ as conjectured by Wormald on the maximum number of Hamiltonian cycles.

Chapter 6

Future Work

In this thesis, the tournament T_n is constructed by placing three copies of T_{n-1} in a triangle and connecting them accordingly. Since our underlying area of interest is the maximum number of Hamiltonian cycles a tournament can have, it would be interesting to construct and study the tournament T_n by placing m copies of T_{n-1} on regular m-sided polygons and connecting them in a way we hope to maximize the number of Hamiltonian cycles in T_n. In particular, an area of interest would be looking at the tournaments that beat Wormald's conjecture of 2.8559... times the expected number thus giving us more insight to his conjecture.

Appendices

Appendix A Sage(Python) code for building T_n

```
def tournament(n):
    tournament = create_cycles(n,{1:[]}, n)
    return tournament
def create_cycles(n,graph, m):
    if n == 0:
        return graph
    else:
        graph2 ={}
#this part just creates copies and increments them accordingly
        for key in graph:
            newkey = key + 3^(m-n)
            graph2.update({newkey:[]})
            for v in graph[key]:
                newv = v + 3^(m-n)
                graph2[newkey].append(newv)
        graph3={}
        for key in graph2:
            newkey = key + 3^(m-n)
            graph3.update({newkey:[]})
            for v in graph2[key]:
                newv = v + 3^(m-n)
                graph3[newkey].append(newv)
        #end of incrementing the disjoint graphs
        # we now have three disjoint graphs, graph, graph2 and graph3
        # all points in graph => graph2 => graph3 => graph
        for key in graph:
            for vertex in graph2:
                graph[key].append(vertex)
        for key in graph2:
```

```
    for vertex in graph3:
        graph2[key].append(vertex)
for key in graph3:
    for vertex in graph:
        graph3[key].append(vertex)
graph.update(graph2)
graph.update(graph3)
create_cycles(n-1,graph, m)
return graph
```

Appendix B Sage(Python) code for Approximation algorithm

```
def count_ham_cycles_in_T2(m, N): #N is the sample size. m is T_m
    import random
    graph = tournament(m)
    print 'This tournament has %d vertices' %(len(graph))
    print 'Sample size is %d' %N
    map = DiGraph(graph)
    nV = len(graph)
    p =[]
    averages  = []
    visited = {}
    for vertex in graph:
        visited[vertex] = false
    one_in = map.neighbors_in(1)
    for j in range(1,11):
        prod_of_degrees = []
        term_count = 0
       #number of times we terminate we reach a dead end
        ham_count = 0
        for i in range(1,N+1):
            #map = copy(map1)
            #visited
            #counter for remaining place to visit
            for vertex in graph:
                visited[vertex] = false
            walk =[]
            prob_list =[]
            neighlist = []
            walk.append(1)
            neighbor = map.neighbors_out(1)
```

```
visited[1] = true
for neigh in neighbor:
    if visited[neigh] == false:
        neighlist.append(neigh)
if neighlist == []:
    if len(walk) != nV:
        prod_of_degrees.append(0)
        term_count += 1
        break
    else:
        break
else:
    a = random.choice(neighlist)
    visited[a]=true
    degree = len(neighlist)
#neigh =map.neighbors_out(1)
#a = random.choice(neigh)
#degree = len(neigh)
#map.delete_vertex(1)
walk.append(a)
prob_list.append(degree)
#print prob_list
for road in range(1,nV-1):
    if map.neighbors_out(a)== []:
        if len(walk) != nV:
            prod_of_degrees.append(0)
            term_count += 1
            break
        else:
            break
    else:
```

```
            neighlist = []
            neighbor =map.neighbors_out(a)
            for neigh in neighbor:
                if visited[neigh] == false:
   # the available vertices to go to.
                    neighlist.append(neigh)
            if neighlist == []:
                if len(walk) != nV:
                    prod_of_degrees.append(0)
                    term_count += 1
                    break
                else:
                    break
            b = random.choice(neighlist)
#choose at random a vertex to go
            visited[b]=true
            degree = len(neighlist)
            #map.delete_vertex(a)
            walk.append(b)
            prob_list.append(degree)
            a = b
        if len(walk) == nV:
            if walk[-1] in one_in:
                #this is a ham cycle
                ham_count += 1
                x= prod(prob_list)
                prod_of_degrees.append(x)
            else:
                term_count += 1
                prod_of_degrees.append(0)
    #print walk
```

```
av1 = mean(prod_of_degrees)
print 'average=', av1.n()
```

Appendix C   Python code for Exact counting Algorithm

C.1   code for E(n,i,j,k) and P(n,i,j,k)

```
#In the code below the function F(n,i,j,k) = P(n,i,j,k)


# this is used to speed up the execution of the following function


E_cache = {(0, 0, 0,1): 1} #E(n, i, j, k)
from math import factorial
def E(n, i, j, k):
#Everything. This includes all broken, proper and circular paths
    sum = 0
    numera= (memo_factorial[i]*memo_factorial[j]*memo_factorial[k])**2
    if i+j >= n:
        N = 0
    else:
        N = n - i - j
    for a in range(N, i+1):
        #print a, n-a-N,  j+1
        for b in range(max(0, n-a-i), min(j, n-a)+1): #because n-b-a <= N
            c = n- b -a
            denom = memo_factorial[a]*memo_factorial[b]
*memo_factorial[c]*memo_factorial[i-a]*memo_factorial[j-b]*memo_
factorial[k-c]*memo_factorial[i-c]*memo_factorial[j-a]*memo_factorial[k-b]
            sum += numera/denom
    if n > 0 and i > 0 and k != 81:
#Should always be != 3^(n-2) for T_n
        E_cache[(n, i, j, k)] = sum
    return sum


# this is used to speed up the execution
```

```
memo_factorial =  {}
for i in range(3**6 + 1):
#The max factorial to be used, i.e up to 3^{n-2}
    memo_factorial[i] = factorial(i)

def E2(n, i,j, k):
    if n <=0 or i == 0:
        if n < 0:
            return 0
        elif n == 0:
            return 1
        else:
            return binomial(j,n)*binomial(k,n)*memo_factorial[n]
    else:
        get = E_cache.pop((n, i, j, k))
        return get

def P(n, i, j, k):
# this is P(n, i, j, k) = E(n, i, j, k) + C_E(n, i, j, k)
    return E(n, i, j, k) - i*j*k *E2(n-3, i-1, j-1 ,k-1)
```

C.2 Code for computing $H(T_n)$

```
def ham_cycles_in_Tn(N,prev):
    print 'This tournament has %d Vertices' %(3^N)
    c = []
    check = []
    v = 3**(N-1)
    bsize =v/3
    for i in range(v):
        c.append(0)
        #check.append([])
```

```
    for i in range(1,bsize+1):
        for j in range(i,bsize+1):
            for k in range(j,bsize+1):
 #w1 are the ways of getting max components form the given [i,j,k]
                    ways = [0,0,0]
                    ways[0] = prev[i-1]
                    ways[1] = prev[j-1]
                    ways[2] = prev[k-1]
                    nv = i + k + j
                    w1 = ways[0]*ways[1]*ways[2]
                    if i == j == k: #e.g [i, j, k] = [1,1,1]
                        #check[nv-1].append((1, w1))
                        c[nv-1] += 1*w1  #if n=0
                        for n in range(1,2*i + j + 1):
                            if nv-n-1 >= 0:
                                paths = P(n, i, j, k)
                                #print 'fin'
                                c[nv-n-1] += w1 * paths
                    elif i == j or j == k:
 #e.g[i, j, k]= [1,1,3] = [1,3,1] = [3,1,1]..... 3 ways
                        c[nv-1] += 3*w1  #if n=0
                        for n in range(1,2*i + j + 1):
                             paths = P(n, i, j, k)
                             #print 'fin'
                             c[nv-n-1] += 3*w1 * paths
#3 ways of symmetry
                    else:
                    # e.g [i, j, k]= [1,2,3] = [1,3,2] = ...6 ways
                        c[nv-1] += 6*w1  #if n=0
                        for n in range(1,2*i + j + 1):
```

```
                                paths = P(n, i, j, k)
                                c[nv-n-1] += 6*w1 * paths
#6 ways of symmetry

    #print c
    p = c
    ham = []
    for i in range(1,len(c)+1):
        ham.append((memo_factorial[i]*p[i-1])**3/i)

    print sum(ham)
    #print 'T_ %d has %d Hamiltonian Cyles' %(N, sum(ham))
    #print factor(sum(ham))
    return c
```

Appendix D Exact Values for $H(T_n)$

$H(T_1) = 1$

$H(T_2) = 207$

$H(T_3) = 8316362583640202859$

$H(T_4) =$

82436160974448822095605909117340371652183249296327952102912969834086548990320509982103591486399

$H(T_5) =$

46819457080276057466307683938034858702468662057845014052870886885809287529793652259792541636757536636982735344207909684023091976314786055001406793264208299645426826233564664327389492251023438198011757289405511540154814868673358358299827245880866219324935541349357368442219799684891173556263405548470691225154463506711394744842101586001666627303220706375301612046536746191175816294508031389792390069707

$H(T_6) =$

195133590743534532849004708713735747988827346969870858809750680568848875249361386297636673176824176656501812898600724973543825293151024657545668257685581818768192378545573532666296793010265126249134591428249540180961383843225931956719759610569694316039904751631817736858805266546862832410638068580258193066201849420755549335722821597811280572064843505868499504783885508991857600848561349591201044719283127053149781754351535482674437334729148130040364619907968941227826600989432176848051199470768327774744312292093126498048631496130213700625294257351796850434264697697020952126246192501376044986736640294520122759788860681384885594770743632445580242413139289616853289183947939345537972661408886145732258160230509486685338408518926720584285809969813715939362780503520001076562928144898647430604477609502386308252110841535994428990914319523554211753497640316690172744430532030656268734219049930922858231399486858736372305642388920500215924230517702612041258920330389813331715539463604093890573551854931838286879626024145682703084835713843856599499097430405355923523194654300754071840

7031689312288999221332784173470338807615602001172147745662250147731814641674751541478235862564028648999554570461421798362595737721299355505615386047584648318190789359588600327869847223017153024208684200827327073913603199410628837765076428085424856578242030393517110510586876470445030946584592393957294148651318819031535439285205652256309048178281948080910285349167428167342804318135205692872505493881276874487864062140023214417628571223212298332122565315394904939264887O3

Bibliography

[1] N. Alon. The maximum number of hamiltonian paths in tournaments. In *Combinatorica*, pages 319–324, 1990.

[2] R. Diestel. *Graph Theory*. Springer, 2010.

[3] E. Friedgut and J. Kahn. On the number of hamiltonian cycles in a tournament. manuscript.

[4] K.D. Joshi. *Foundations Of Discrete Mathematics*. New Age International, 2003.

[5] D. E. Knuth. *The Art of Computing*. Addison-Wesley, 1968.

[6] D.L. Kreher and D.R. Stinson. *Combinatorial algorithms. Generetion, enumeration, and search*. CRC press, 1999.

[7] T. Szele. Kombinatorikai vizsg alatok az ir anyitott teljes gr a al kapcsolatban.

[8] N. Wormald. Tournaments with many hamiltonian cycles.