The Watchman's Walk Problem on Directed Graphs

by

© Brittany Pittman

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Department of Mathematics and Statistics

Memorial University

July 2020

St. John's, Newfoundland and Labrador, Canada

Abstract

A watchman's walk in a graph is a minimum closed dominating walk. Given a graph and a single watchman, the aim of the Watchman's walk problem is to find a shortest closed walk that allows the guard to efficiently monitor all vertices in the graph. In a directed graph, a watchman's walk must obey the direction of the arcs. In this case, we say that the guard can only move to and see the vertices that are adjacent to him relative to outgoing arcs. In this thesis, we consider the watchman's walk problem on directed graphs. In particular, we study the problem on tournaments, orientations of complete bipartite and multipartite graphs, and directed graphs formed from sequences.

Acknowledgements

I would like to thank my supervisors, Dr. Danny Dyer and Dr. Jared Howell, for their suggestions, kindness, and guidance in this research project.

I wish to acknowledge the financial assistance provided by the School of Graduate Studies, Department of Mathematics and Statistics, and the Natural Sciences and Engineering Research Council of Canada.

I would also like to thank my family and friends for their support and encouragement during this process.

Statement of contribution

This thesis is a collaboration of work by Brittany Pittman, Dr. Danny Dyer, and Dr. Jared Howell. The included results were developed by all parties.

Table of contents

List of tables

List of figures

Chapter 1

Introduction

Suppose we have a building, such as a museum, that we would like to monitor with guards. In this museum, every room is joined to its neighbouring rooms via doors or hallways, and from any room, a guard is able to see all adjacent rooms. We could ensure that each room is constantly monitored by placing a guard in every room. Or, more efficiently, by stationing the guards such that every room is occupied by a guard, or a guard occupies an adjacent room. However, suppose that there are not enough guards to constantly monitor each room. Instead, we have only one guard to monitor all of the rooms. In this case, we require that the guard moves through the building in order to monitor all the rooms. We want the guard to continuously repeat a single route, so the route must start and end in the same room. We also want to ensure that every room is monitored. To do so, we require that at least once during the route, each room is occupied by the guard, or the guard is occupying an adjacent room. In the interest of efficiency, we want to find such a walk of the shortest possible length. We call this route a *watchman's walk*, and we call the problem the *watchman's walk problem*. This problem was introduced by Hartnell, Rall, and Whitehead in 1998 in [14]. We will discuss watchman's walks on graphs further in Section 1.2. In the next section, we give the background and important terminology of graph theory.

1.1 Graphs and digraphs

A graph is an ordered pair $G = (V(G), E(G))$. We say that $V(G)$ is the set of vertices of G, and $E(G)$ is the set of edges. Each element of $E(G)$ is a 2-subset of $V(G)$, and we say that two vertices are *adjacent* if there is an edge between them. A vertex v is *incident* with an edge $e \in E(G)$ if $v \in e$.

For any vertex $v \in V(G)$, the *degree* of v, denoted $\deg(v)$, is the number of edges incident with v. The *(open) neighbourhood* of v in G, denoted by $N_G(v)$ or simply $N(v)$, is the set of vertices adjacent to v. The *closed neighbourhood* of v, denoted by $N[v]$, is the set $N(v) \cup \{v\}$. The maximum degree of a graph G is the maximum value of its vertex degrees, denoted by $\Delta(G)$. Similarly, the minimum degree of G, $\delta(G)$ is the minimum value of its vertex degrees. The *order* of a graph G is the number of vertices in G, $|V|$. This is denoted by $|G|$. The number of edges in G, $|E|$, is called the *size* of G.

A *walk* W in a graph $G = (V, E)$ is a sequence of vertices and edges such that an edge immediately preceding or following a vertex in the sequence is incident with that vertex. If all vertices and edges are distinct, we call W a *path*. The *length* of W is the number of edges in the sequence, counting repetition. A walk is a *trivial walk* if it contains no edges. A walk is *closed* if it begins and ends at the same vertex. A closed path is a *cycle*, and a cycle of length k is called a *k-cycle*. We denote a k-cycle by C_k. Similarly, if G is a path of order n, we denote G by P_n. A graph is *connected* if there is a path between each pair of vertices. If G is not connected, then the maximal connected subgraphs of G are called the *components* of G.

If G is a graph of order n, such that every pair of vertices in G are adjacent, then G is called the *complete graph of order* n, and is denoted by K_n. A connected graph that contains no cycles is a *tree*. In trees, the vertices of degree 1 are called *leaves*. A graph is *bipartite* if its vertex set V can be partitioned into two sets, A and B, such that no two vertices in the same set are adjacent. G is a *multipartite* graph if V can be partitioned into $k \geq 2$ sets, where no two vertices in the same set are adjacent. We say that a bipartite or multipartite graph is *complete* if each pair of vertices from different sets of the partition are adjacent.

Graphs G and H are *isomorphic*, denoted $G \simeq H$, if there exists a bijection f from the vertex set of G to the vertex set of H such that uv is an edge in G if and only if

$f(u)f(v)$ is an edge in H. We say that $H = (V', E')$ is a *subgraph* of $G = (V(G), E(G))$ if $V' \subseteq V(G)$ and $E' \subseteq E(G)$. Further, H is a *spanning subgraph* of G if $V' = V(G)$. An *induced subgraph* of $G = (V(G), E(G))$ is a subgraph H with vertices $S \subseteq V(G)$, where each pair of vertices in S are adjacent in H if and only if they were adjacent in G. We say that H is the subgraph induced by S, and we denote it by $G[S]$. For a subset of vertices $U \subset V(G)$, we denote by $G\backslash U$ the subgraph induced by $V(G)\backslash U$. Similarly, for $E' \subset E(G)$ we denote by $G\backslash E'$ (or $G - e$ for a single edge e) the graph G' with vertex set $V(G)$ and edge set $E(G)\backslash E'$.

A set $S \subseteq V$ is a *dominating set* in G if every vertex in $V(G)$ is either an element of S, or is adjacent to a vertex in S. It is not difficult to find a dominating set in a graph, as we could simply take the entire vertex set. So, we most often are concerned about minimality. A set S is said to be a *minimum dominating set* in G if there is no dominating set in G with fewer vertices. The size of a minimum dominating set is called the *domination number* of G, denoted by $\gamma(G)$. A *total dominating set* is a subset $D \subseteq V(G)$ such that for all $v \in V(G)$, v is adjacent to a vertex $u \neq v$ in D. The minimum size of a total dominating set in a graph G is denoted by $\gamma_t(G)$. A *connected dominating set* is a dominating set such that the subgraph induced by S is connected. The *connected domination number* is denoted by $\gamma_c(G)$. A cycle C in a graph G is a *dominating cycle* if each vertex of G is adjacent to at least one vertex of C. The length of the smallest dominating cycle in G is denoted by $\gamma_{cyc}(G)$.

A walk is a *dominating walk* if its vertices form a dominating set. In a connected graph, a dominating walk can be found by constructing a walk through the vertices of a dominating set. However, as a dominating walk must be connected, the length of a dominating walk will be at least the size of a minimum dominating set in that graph.

A walk that begins at vertex u and ends at vertex v is called a *u-v walk*. If a graph G has a u-v walk, for $u, v \in V(G)$, then the distance from u to v in G, denoted $d_G(u, v)$ (or $d(u, v)$ when G is clear), is the length of a shortest u-v walk in G. A closed walk that repeats no edges is a *circuit*. If a circuit uses every edge of G then it is called an *Eulerian circuit*. A *Hamilton cycle* is a cycle which includes every vertex of G. If such a cycle exists in G, then we say that G is *Hamiltonian*. Similarly, a *Hamilton path* in a graph G is a path that visits every vertex of G exactly once.

A *directed graph* (or *digraph*) is a pair $D = (V(D), A(D))$, where $V(D)$ is the set of vertices of D and $A(D)$ is a set of directed edges or arcs. The arcs in $A(D)$ are ordered pairs of vertices from $V(D)$. We say such an ordered pair (u, v) is directed from u to v. A digraph is *strongly connected* if there exists a directed path between each pair of vertices. In a directed graph, the sequence of vertices of a walk or path must obey the direction of the arcs. If a tournament is strongly connected, we call it a *strong tournament*. We say that D is *weakly connected* if the underlying undirected graph is connected.

For a vertex v in a digraph D, we define the *in-degree* of v as the number of arcs of the form (u, v) in $A(D)$, and we denote this by $\deg^-(v)$. The *in-neighbourhood* of v, denoted $N^-(v)$, is the set of vertices $\{u \in V(D)|(u, v) \in A(D)\}$. We say that u is an *in-neighbour* of v if there is an arc from u to v. Similarly, the *out-degree* of v, denoted $\deg^+(v)$, is the number of arcs that are directed from some vertex u to v, and the *out-neighbourhood* of v, $N^+(v)$, is the set of vertices $\{u \in V(D)|(v, u) \in A(D)\}$. If $deg^-(v) = 0$, then v is called a *source*. If $deg^+(v) = 0$, we say that v is a *sink*.

A *strongly connected component* of a digraph D is a subdigraph of D that is also strongly connected. The *condensation* of a digraph D, denoted by D^*, is the digraph found by contracting each maximal strongly connected component to a single vertex. In this digraph, there is an arc from vertex W to vertex U if all arcs between strongly connected components W and U in T are directed from a vertex in W to a vertex in U.

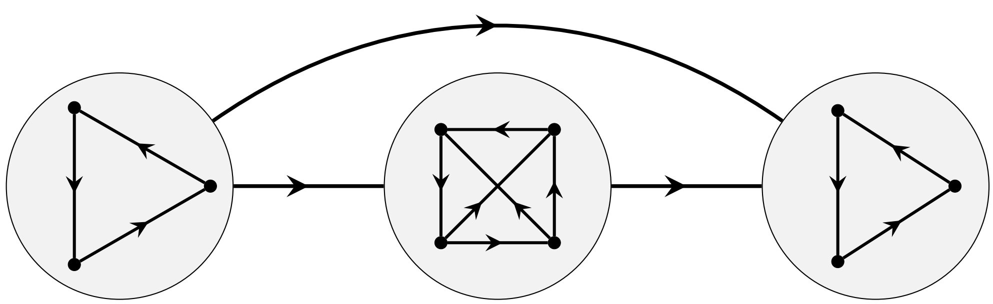


Figure 1.1: The condensation of a tournament with 3 strong components

We can form a directed graph from an undirected simple graph G by assigning exactly one direction to each edge in G. The resulting graph is called an *orientation* of G. A *tournament* is an orientation of a complete graph.

A tournament $T = (V, A)$ is *transitive* if, for $a, b, c \in V(T)$, $(a, b) \in A(T)$ and $(b, c) \in A(T)$ implies that $(a, c) \in A(T)$. Equivalently, there exists an ordering of the vertices of T, $\{v_1, v_2, \ldots, v_n\}$ such that $(v_i, v_j) \in A(T)$ if and only if $i < j$. A tournament of order n is *pancyclic* if there exists a directed cycle of length k for each $3 \leq k \leq n$. The following theorem, Theorem 1.1.1, is a classic result regarding the pancyclicity of tournaments that will be useful in Chapter 2.

Theorem 1.1.1. [2] *Every vertex in a strong tournament of order n is contained in a cycle of length k for $k = 3, 4, \ldots, n$.*

Theorem 1.1.2 and Theorem 1.1.3 are fundamental results on the existence of Hamilton cycles and paths in tournaments. Theorem 1.1.2 is a classic theorem in the study of directed graphs. This statement can be proved inductively, as follows.

Theorem 1.1.2. [18] *Every tournament contains a Hamilton path.*

Proof. First consider the tournament of order 1. This tournament has a trivial Hamilton path. Now suppose that the statement holds for any tournament on n vertices, for some n. Let T be a tournament on $(n+1)$ vertices. Let $V(T) = \{v_1, v_2, \ldots, v_{n+1}\}$. Consider the subtournament $T' = T \backslash \{v_{n+1}\}$. This is a tournament on n vertices, so T' has a directed Hamilton path. Let such a path be $\{v_1, v_2, \ldots, v_n\}$. We can also consider this as a path in T. If (v_{n+1}, v_1) is an arc, $\{v_{n+1}, v_1, v_2, \ldots, v_n\}$ is a Hamilton path in T. Otherwise, we can let $i \in \{1, \ldots, n\}$ be the largest integer such that (v_j, v_{n+1}) is an arc for each $j \leq i$. Now, $\{v_1, \ldots, v_i, v_{n+1}, \ldots, v_n\}$ is a Hamilton path in T. □

Theorem 1.1.3. [4] *A tournament is Hamiltonian if and only if it is strongly connected.*

The existence of Hamilton cycles in tournaments is a particularly important result. A Hamilton cycle is both closed and dominating, and hence this result guarantees that a closed dominating walk will exist in any strong tournament. This will be an important fact when we consider tournaments in Chapter 2.

1.2 Domination

The area of domination in graphs was first motivated by the game of chess. In 1862, C.F. de Jaenisch introduced a preliminary form of domination in [7], where he described a problem in which the aim was to find the minimum number of queens needed to reach any square on an $n \times n$ chessboard in one move. Many modern applications of domination also exist, such as in the areas of communication or electrical networks. Domination in graphs was formally introduced in 1958 by Berge in [3]. The domination problem in graphs aims to find a dominating set of minimum size. Variations of this problem include finding a minimum total dominating set, or a minimum connected dominating set.

A survey of results on the domination number of graphs can be found in [15]. In [17], Ore noted the following result for graphs G with $\delta(G) \geq 1$.

Theorem 1.2.1. [17] *If G is a graph such that $\delta(G) \geq 1$ then $\gamma(G) \leq \frac{n}{2}$.*

There are many parameters that can be related to the domination number of a graph. The largest size of a minimal dominating set in a graph G is denoted $\Gamma(G)$. If S is a subset of vertices in G then S is called an *irredundant set* if, for every vertex v in S, $N[S \backslash \{v\}] \neq N[S]$. The size of the smallest maximal irredundant set is denoted by $ir(G)$, and the size of the largest maximal irredundant set is denoted by $IR(G)$. A set S is an *independent set* if no two vertices in the set are adjacent. We denote the smallest maximal independent set by $i(G)$, and the largest by $\alpha(G)$. Cockayne et al. compare these values in the following domination chain.

Theorem 1.2.2. [5] *For any graph G,*

$$ir(G) \leq \gamma(G) \leq i(G) \leq \alpha(G) \leq \Gamma(G) \leq IR(G).$$

While domination is primarily studied on undirected graphs, the problem has also been studied on directed graphs. Directed domination in tournaments was first considered by Erdös in [11].

A set $S \subseteq V$ is a *dominating set* in a directed graph if for every vertex $u \notin S$ there exists a vertex $v \in S$ such that there is an arc from u to v. The *lower directed domination number* of a graph G is the minimum value of the domination number over all possible orientations of D, and is denoted $\gamma_d(D)$. Similarly, the *upper directed domination number* of a graph D is the maximum value of the domination number over all possible orientations of D, and is denoted $\Gamma_d(D)$.

In [20], Reid et al. consider the domination number and irredundance number of tournaments. They give upper bounds on the domination number of tournaments of small order. We make use of these results in the following chapter. In [8], Duncan and Jacobson prove the existence of tournaments with any given domination number.

1.3 The watchman's walk problem

The problem of finding an optimal route in a museum can easily be considered on graphs. To do this, we can think of the rooms as vertices on a graph G, and the hallways between pairs of rooms as edges. Since each guard in the building can see what is happening in all adjacent rooms, the set of rooms in which we can place guards such that every room is constantly monitored is equivalent to a dominating set in our graph. The minimum number of guards needed to ensure that every room is constantly monitored is the size of a minimum dominating set in G. Now consider finding the shortest route for the guard, starting and ending in the same room, such that every room is seen at least once during the walk. When considered on a graph G, this translates to finding a shortest walk W such that W is a closed walk whose vertices form a dominating set in G. This walk is a *minimum closed dominating walk (MCDW).* Finding such a walk is precisely the aim of the *watchman's walk problem.*

The watchman's walk problem is a variation of the domination problem. The aim of this problem is to find a minimum closed dominating walk in a given graph G. The length of a minimum closed dominating walk in a graph G with 1 guard is denoted by $w(G)$.

While the watchman's walk problem is a variation of the domination problem, the watchman number and the domination number of a given graph G can often be very different. The vertices of a watchman's walk W must form a dominating set. However, since W must also be closed and connected, it does not necessarily contain a minimum dominating set. In Figure 1.2, the graph shown has minimum dominating set $\{v_5, v_6\}$. However, the only watchman's walk in the graph, illustrated by the dashed edges, does not contain any minimum dominating set. Also, since vertices may be repeated in a watchman's walk, $w(G)$ may be much larger than $\gamma(G)$.

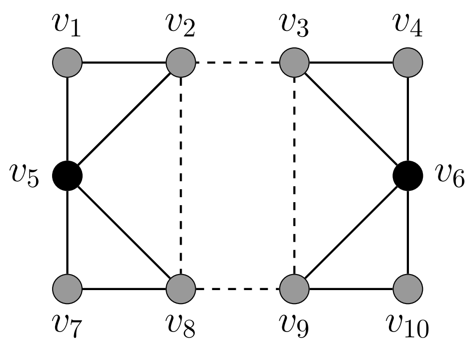


Figure 1.2: Graph with $w(G) = 4$ and $\gamma(G) = 2$

The original watchman's walk problem has often been studied for undirected graphs, and many results for this problem can be found in [9], [10], and [14]. However, little is known about the watchman's walk of directed graphs. On a directed graph, the watchman can only move in the direction of the arc, and also can only see in the direction of an arc. Like in undirected graphs, the directed watchman's walk has to begin and end in the same place, but now it must follow the direction of the arcs. Additionally, in a digraph, vertices dominate only themselves and their out-neighbours. It follows that an arbitrary digraph will always have a minimum dominating set, but may not have a watchman's walk. For example, the path shown in Figure 1.3 has a minimum dominating set of size 3. However, it has no watchman's walk, as there are no closed walks of length greater than zero in this graph.

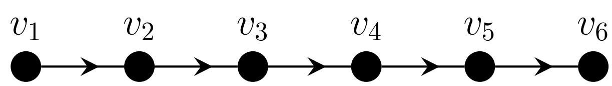


Figure 1.3: Orientation of P_6

The digraph in Figure 1.3 is not strongly connected. Any strongly connected digraph will have a watchman's walk, as it is possible to find a walk that visits every vertex, but a digraph that is not strongly connected may or may not have a watchman's walk. The digraph in Figure 1.4 is not strongly connected. However, it does have a watchman's walk, namely the walk of length 0 at its dominating vertex, v_2.

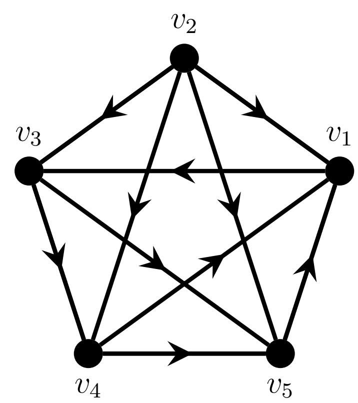


Figure 1.4: A digraph that has a watchman's walk but is not strongly connected.

In a directed graph D, a dominating set can induce a subdigraph that is disconnected, weakly connected, or strongly connected. If a dominating set induces a weakly connected subdigraph, we call it a *weakly connected dominating set*. The size of a minimum weakly connected dominating set is called the *wc-domination number*, denoted by $\gamma_{wc}(D)$. If a dominating set induces a strongly connected subdigraph, we call it a *strongly connected dominating set*. The minimum size of such a set is called the *sc-domination number*, denoted by $\gamma_{sc}(D)$. We call a minimum weakly connected dominating set a γ_{wc}*-set*, and a minimum strongly connected dominating set a γ_{sc}*-set*. These parameters were studied by Arumugam et al. in [1]. They note the following chain of inequalities for these values.

Theorem 1.3.1. [1] *If $D = (V, A)$ is a digraph with a connected underlying undirected graph, then $\gamma(D) \leq \gamma_{wc}(D) \leq \gamma_{sc}(D)$.*

Finding a connected dominating set in an undirected graph, and a weakly or strongly connected dominating set in a directed graph is a problem similar to finding a watchman's walk. The vertices of a minimum closed dominating walk in an undirected graph are always a connected dominating set. In a directed graph, the vertices of a watchman's walk will be a strongly connected dominating set. However, in many directed graphs, these parameters and sets of vertices are not equal. If we consider the directed path in Figure 1.3, the set of vertices $\{v_1, v_2, v_3, v_4, v_5\}$ is a minimum weakly connected dominating set, and $\gamma_{wc}(D) = 5$. However, there are no strongly connected dominating sets or watchman's walks.

In directed graphs that have strongly connected dominating sets and watchman's walks, $\gamma_{sc}(D)$ and $w(D)$ are not necessarily equal. In the directed graph in Figure 1.5, $\{v_2, v_3, v_4, v_5\}$ is a minimum dominating set that induces both a strongly connected subdigraph and a weakly connected subdigraph, and $v_2, v_3, v_4, v_5, v_4, v_3, v_2$ is a minimum closed dominating walk. So, $\gamma_{sc}(D) = 4$, $\gamma_{wc}(D) = 4$, and $w(D) = 6$.

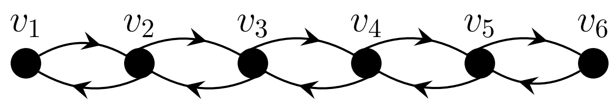


Figure 1.5: Directed graph with $\gamma_{sc}(D) = 4$ and $w(D) = 6$

In this graph, the sc-domination number is equal to the wc-domination number. Figure 1.6 illustrates a directed graph where this is not the case. A γ_{wc}-set in this graph is $\{u_1, v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$, while the only sc-dominating set has size 20 and must use every vertex in the graph. A watchman's walk of this graph, as illustrated by the dashed lines, also must use every vertex. From this graph, we can see that $\gamma_{sc}(D)$ can be arbitrarily larger than $\gamma_{wc}(D)$ in a directed graph D.

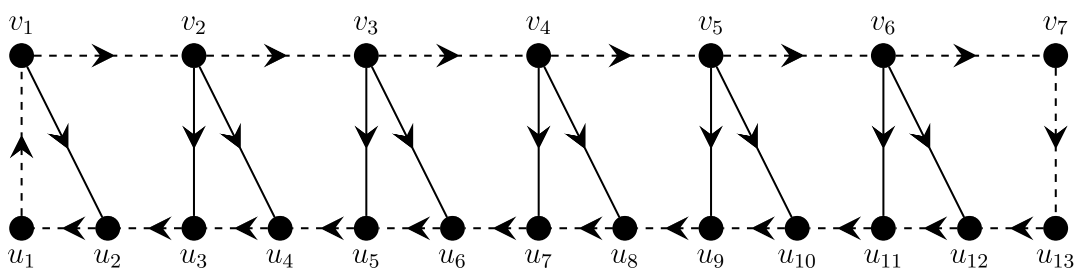


Figure 1.6: Directed graph with $\gamma_{wc}(D) = 7$ and $\gamma_{sc}(D) = 20$

In the directed graphs in Figure 1.5 and Figure 1.6, the vertex set of any watchman's walk of the digraphs contains a γ_{sc}-set. However, this is not the case in general. In the directed graph D in Figure 1.7, vertex v_7 has 5 out-neighbours, and each other vertex has at most 2. In any γ_{sc}-set, each vertex in the set is dominated by another vertex in the set. Hence, each vertex v in an γ_{sc}-set uniquely dominates at most $|N^+(v)|$ vertices. It follows that a strongly connected subset containing 6 vertices dominates at most $5 + 5 \times 2 = 15$ vertices. Since D is a digraph of order 15, no strongly connected subset of size less than 6 is dominating. In a subset of 6 vertices, any strongly connected set not containing v_7 dominates at most 12 vertices. Hence, v_7 would need to be in the subset of size 6 for it to be dominating. If u_7 was in a subset of 6 vertices, at most $5 + (4 \times 2) + 1 = 14$ vertices would be dominated, so u_7

must not be included. It can be checked algorithmically by the program in Appendix C that no dominating set of size 6 induces a strongly connected subset. This program determines all strongly connected subsets that contain v_7 and do not contain u_7, and verifies that no such subsets are dominating in the graph. Thus, $\gamma_{sc}(D) = 7$, and the set $S = \{v_1, v_2, \ldots, v_7\}$ is a γ_{sc}-set. The shortest closed walk containing all of these vertices has length 9. The walk of length 8 illustrated by dashed lines is a closed dominating walk in the graph, and hence $w(D) \leq 8$. Thus, no watchman's walk in this graph contains the γ_{sc}-set S as a subset of its vertices. In particular, the watchman's walk illustrated by the dashed arcs is completely disjoint from the γ_{sc}-set S.

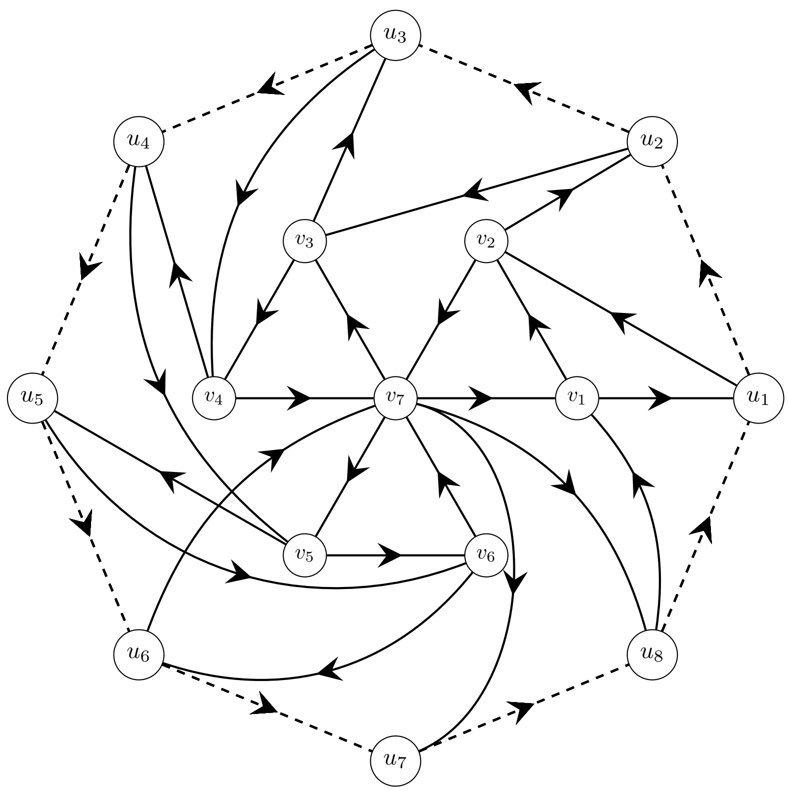


Figure 1.7: A digraph with disjoint γ_{sc}-set and watchman's walk

In the following chapters, we consider families of directed graphs that have watchman's walks. We first consider tournaments, in Chapter 2, and later consider other families of digraphs, including orientations of complete bipartite and multipartite graphs, as well as de Bruijn digraphs and some of their subdigraphs. These families of digraphs each have a large amount of structure, and thus are ideal for a first consideration of the watchman's walk problem on digraphs.

Chapter 2

Watching tournaments

Unlike the case of undirected graphs, a digraph may not have a watchman's walk. In an arbitrary digraph D, having a connected underlying undirected graph does not ensure that D has a watchman's walk. We begin by considering the watchman's walk problem on tournaments, as they are a family of directed graphs with a high density of arcs, and hence a family in which a watchman's walk would seem likely to exist. In fact, we will show that any tournament has a watchman's walk. We also consider other questions regarding the watchman number of tournaments, domination number, and multiplicity of watchman's walks.

2.1 Watchman's walks in tournaments

If a tournament is strongly connected, we call it a strong tournament. In this section, we refer to classical results, primarily those found in [19] and [21], to prove the existence of a watchman's walk in strong tournaments, and later, we generalize our result for tournaments that are not necessarily strong. We also provide bounds on the length of a watchman's walk in tournaments. From Theorem 1.1.3, found in [21], we get the following theorem.

Theorem 2.1.1. *If T is a strong tournament, then T has a watchman's walk.*

Proof. If T is a strong tournament then, by Theorem 1.1.3, T has a Hamilton cycle H. Since every vertex is on this cycle, this is a closed dominating walk. This means that the set of closed dominating walks of T is non-empty. So, a minimum closed dominating walk of T exists. □

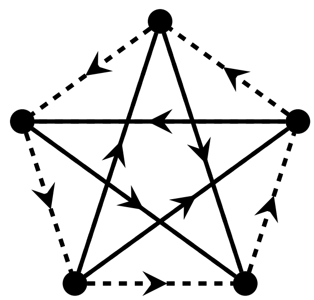


Figure 2.1: A tournament with a Hamilton cycle.

From Theorem 2.1.1, we know that any strong tournament on n vertices has a closed dominating walk of length n. Due to the density and structure of tournaments, it is reasonable to suspect that not all vertices in a tournament would need to be included in a watchman's walk. Indeed, Theorem 2.1.2 and Theorem 2.1.3 show that we can always find a closed dominating walk with less than n vertices.

Theorem 2.1.2. *Let T be a tournament of order $n > 3$. If T is strong, then $w(T) \leq n - 1$.*

Proof. If T is a strong tournament then, by Theorem 1.1.3, T is Hamiltonian. Let $H = v_0, v_1, v_2, \ldots, v_{n-2}, v_{n-1}, v_0$ be a Hamilton cycle in T. It is clear that H is a closed dominating walk in T, so $w(T) \leq n$. Consider each arc between vertices v_i and v_{i+2}, where the indices are considered modulo n.

Case 1: There exists some arc (v_i, v_{i+2}). If $W = v_0, v_1, \ldots, v_i, v_{i+2}, \ldots, v_{n-1}, v_0$, then W is closed, and the only vertex in T that is not in W is v_{i+1}. This vertex is dominated by W, since (v_i, v_{i+1}) is an arc and $v_i \in W$. Thus, W is a closed dominating walk of length $n - 1$.

Case 2: There is no such arc (v_i, v_{i+2}). Since T is a tournament, every arc of the form (v_{i+2}, v_i) is in T. So, if n is even, let $W = v_{n-1}, v_{n-3}, \ldots, v_3, v_1, v_{n-1}$. Each $v_i \notin W$ is dominated by $v_{i-1} \in W$ since (v_{i-1}, v_i) is an arc. Thus, W is a closed dominating walk of length $\frac{n}{2}$. If n is odd, let $W = v_{n-1}, v_{n-3}, \ldots, v_4, v_2, v_0, v_1, v_{n-1}$. Similarly to when n is even, W is a dominating cycle, and W has length $m = \frac{n+1}{2} + 1 = \frac{n+3}{2}$. Note that for all $n > 3$, $m \leq n-1$. In either case, there exists a closed dominating walk of length at most $n-1$ as required. □

In the tournament of order 5 illustrated in Figure 2.1, a 4-cycle exists, but it has a watchman's walk of length 3. In fact, Theorem 2.1.3 shows that we can always do better than the bound given in Theorem 2.1.2 for tournaments of a larger order.

Theorem 2.1.3. *If T is a strong tournament of order $n \geq 5$, then $w(T) \leq n-2$.*

Proof. Let T be a strong tournament. From Theorem 1.1.1, there is some cycle of length $n-2$ in T, call it C. Consider the vertices u and v that are not in C. Since T is strongly connected, there must be at least one arc from a vertex in C to u or v, and at least one arc from u or v to a vertex in C. Without loss of generality let x be a vertex in C such that (x, v) is an arc and y be a vertex in C such that (u, y) is an arc.

Case 1: There exists a vertex in C that dominates u. In this case, since (x, v) is also an arc and $x \in C$, it follows that both u and v are dominated by C. Since all other vertices in T are in C, the set C is a closed dominating walk of length $n-2$.

Case 2: Vertex u is not dominated by a vertex in C. In this case, (v, u) must be an arc. Since (x, u) is not an arc, (u, x) must be an arc. Consider the walk $W = \{u, x, v, u\}$. This walk is dominating, since u dominates each vertex in C, and the only vertices not in C are u and v, which are both in W. Thus, W is a closed dominating walk of length $3 \leq n-2$ for all $n \geq 5$. □

The upper bounds given in Theorem 2.1.2 and Theorem 2.1.3 are both best possible for tournaments of small order. For Theorem 2.1.2, consider a tournament T on 4 vertices, where no vertex in T is a source. In this case, we need at least 2 vertices to dominate T. Thus, to get a cycle, we have that $w(T) = 3 = n-1$. For Theorem 2.1.3, let $n = 5$. Consider the tournament in Figure 2.2. This tournament does not have a source vertex, so $w(T) \geq 3$. It does, however, have a dominating 3-cycle, so $w(T) = 3 = n-2$.

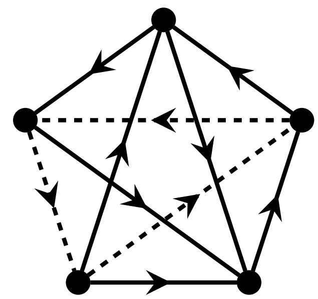


Figure 2.2: A tournament T where $w(T) = n - 2$.

The following theorem draws a connection between a watchman's walk of a spanning subdigraph and the tournament itself.

Theorem 2.1.4. *Let T be a tournament and T' be a spanning subdigraph of T. If T' has a watchman's walk, then $w(T) \leq w(T')$.*

Proof. Let T' be a subdigraph of T such that $V(T') = V(T)$. Suppose W is a watchman's walk in T'. Since T' is a subdigraph of T, W is also a walk in T. Moreover, $V(T') = V(T)$ and W is a dominating walk in T', so W is also a dominating walk in T. Thus, any watchman's walk W in T' is also a closed dominating walk in T. However, there may be a shorter closed dominating walk in T. Therefore, $w(T) \leq w(T')$. □

If we have a spanning subdigraph T' in T, we cannot assume that T' has a watchman's walk, since T' is a digraph. However, if T' has a watchman's walk, Theorem 2.1.4 shows that the length of a watchman's walk in T is bounded above by $w(T')$. Moreover, if we have a (not necessarily spanning) subdigraph T' such that the vertices of T' are a dominating set in T, we will show that $w(T)$ is at most $w(T')$. This fact is a strengthening of Theorem 2.1.4, and is proved in the following theorems.

Recall that a tournament $T = (V, A)$ is transitive if for any vertices a, b, and c, $(a, b) \in A$, and $(b, c) \in A$ implies that $(a, c) \in A$, and that the condensation of a tournament T is the digraph found by contracting each maximal strongly connected component to a single vertex such that there is an arc from vertex W to vertex U if all arcs between components W and U in T are directed from a vertex in W to a vertex in U. From [13], we know that this graph is always transitive for a tournament. We give a proof of this statement in the following theorem.

Theorem 2.1.5. [13] *The condensation of any tournament is a transitive tournament.*

Proof. Let T be a tournament, and T^* be the condensation of T. Let $V(T^*) = \{t_1, t_2, \ldots, t_m\}$, and $T_1, T_2, \ldots, T_m$ be the maximal strong components in T such that vertex t_i in T^* corresponds to component T_i in T. Certainly, T^* is a tournament. If T is a strong tournament, then T^* has only one vertex and no arcs, so it is trivially transitive. Now suppose T is not strong. If T_1 and T_2 are maximal strong components in T then, $t_1, t_2 \in V(T*)$. If T_1 and T_2 each contain exactly one vertex, then there is only one arc between components T_1 and T_2 in T, so either (t_1, t_2) or (t_2, t_1) is an arc in T^*. Now suppose T_1 contains only a single vertex, and T_2 contains move than one vertex. Let $v_1 \in T_1$ and suppose (v_1, u_2) and (v_2, v_1) are both arcs in T for $u_2, v_2 \in T_2$. Since T_2 is a strongly connected component, there is a path $w, \ldots, v_2$ in T_2 from any vertex $w \in T_2$ to v_2, and a path $u_2, \ldots, w$ in T_2. It follows that $w, \ldots, v_2, v_1$ is a path in T from any vertex w in T_2 to v_1, and $v_1, u_2, \ldots, w$ is a path from v_1 to any $W \in T_2$ in T. This contradicts the maximality of T_1 and T_2. Hence, the vertices in T_2 must either be all out-neighbours or all in-neighbours of v_1. Similarly, if T_2 contains only a single vertex, and T_1 contains move than one vertex, the vertices in T_1 must either be all out-neighbours or all in-neighbours of the vertex in T_2. Thus, either (t_1, t_2) or (t_2, t_1) would be an arc in T^*. Now consider the case when both component T_1 and T_2 contain more than one vertex. Let u_1 and v_1 be vertices in component T_1, and u_2 and v_2 be vertices in component T_2. Suppose (u_1, u_2) is an arc in T. Now consider the arc between v_1 and v_2. Since T_1 and T_2 are both strong subtournaments, there is a path from u_2 to any vertex w in T_2, and a path from any vertex x in T_1 to u_1. So, $x, \ldots, u_1, u_2, \ldots, w$ is a path in T from any vertex $x \in T_1$ to any vertex $w \in T_2$. Similarly, if (v_2, v_1) was an arc, there would be a path $w, \ldots, v_2, v_1, \ldots, x$ from any vertex w in T_2 to any vertex in T_1. So, $T_1 \cup T_2$ would be a larger strongly connected component. However, T_1 and T_2 were both maximal. So, if (u_1, u_2) is an arc, we cannot have an arc from a vertex in T_2 to a vertex in T_1. So, all arcs between T_1 and T_2 are from a vertex in T_1 to a vertex in T_2. Thus, (t_1, t_2) is an arc in T^*. Therefore T^* is a tournament.

To show that T^* is a transitive tournament, we will proceed by induction. First suppose T has only two maximal strong components, T_1 and T_2. It follows that there is a single arc in T^*. So, T^* would be transitive. Otherwise, suppose we have a transitive subdigraph of T^* with vertices $\{t_1, t_2, \ldots, t_k\}$ such that (t_a, t_b) is an arc if $a < b$. Consider the maximal component T_{k+1}. We know that for $1 \leq i \leq k+1$, either (t_i, t_{k+1}) or (t_{k+1}, t_i) is an arc in T^*. Also, since $t_1, t_2, \ldots, t_k$ is a directed path in T^*, we cannot have that (t_{k+1}, t_i) and (t_j, t_{k+1}) are both arcs in T^* for $i < j$, as this would create a cycle, contradicting the maximality of the components in T. So, if t_{k+1} has both in-neighbours and out-neighbours, we must have that (t_1, t_{k+1}) is an arc. Thus, either (t_{k+1}, t_i) is an arc for all $1 \leq i \leq k$, or (t_{k+1}, t_1) is an arc and there exists some t_s such that $s \in \{2, \ldots, k\}$ and (t_s, t_{k+1}) is an arc. In the first case, the subtournament is transitive with ordering $t_{k+1}, t_1, \ldots, t_k$. Now let (t_1, t_{k+1}) be an arc and let t_s be the first vertex in the path such that (t_{k+1}, t_s) is an arc. If there exists some vertex t_r such that $s < r \leq k$ and $(t_r, t_{k+1}$ is an arc, then we have that $t_s, \ldots, t_r, t_{k+1}, t_s$ is a cycle in T^*. As shown above, $T_s \cup \ldots \cup T_r \cup T_{k+1}$ would then be a strong component in T, contradicting the maximality of the components. So, for all $r > s$, we must have that (t_{k+1}, t_r) is an arc. Hence, the subtournament of T^* induced by vertices $\{t_1, t_2, \ldots, t_k, t_{k+1}\}$ is transitive with ordering $t_1, t_2, \ldots, t_{s-1}, t_{k+1}, t_s, t_{s+1}, \ldots, t_k$. Thus, by induction, any induced subset of T^* is transitive. Therefore, T^* is transitive. □

Theorem 2.1.5 gives a very useful result. We will show in the following theorems that we can use the condensation of a tournament to simplify the search for a watchman's walk in large tournaments. To do this, we use the notion of a *dominating strong component* in the condensation of a tournament. Recall the definition of a transitive tournament. It is clear from this definition that any transitive tournament has a dominating vertex, namely the first vertex in the transitive ordering. If T is a tournament, and T' is the maximal strong component that corresponds to the dominating vertex in the condensation of T, we call T' the *dominating strong component*. Since any transitive tournament has a dominating vertex, and the condensation of a tournament is a transitive tournament, the dominating strong component in a tournament always exists.

Theorem 2.1.6. *Let T be a tournament. If T' is the dominating strong component in T, then $w(T) = w(T')$.*

Proof. Let T be a tournament. If T is a strong tournament, then the condensation of T has exactly one component, and this component is T itself. In this case, it is clear that $w(T) = w(T')$.

Now suppose that T is not strong. This means that, by Theorem 2.1.5, the condensation of T is transitive with more than one component. Let T' be the dominating vertex of the condensation. The components of the condensation of T are maximal strong components, so for any two components T_i and T_j, we must have that for any $u \in V(T_i)$ and $v \in V(T_j)$, all arcs are of the form (u, v), or all arcs are of the form (v, u). Otherwise, $T_i \cup T_j$ would be a larger strong component of T. So, since T' was the dominating component, any vertex in $V(T')$ dominates every vertex in $V(T) \backslash V(T')$. Thus, any set of vertices that dominate $V(T')$ also dominate $V(T)$. Moreover, no vertex that is in $V(T) \backslash V(T')$ dominates any vertex in $V(T')$, so any minimum dominating set in T is a subset of $V(T')$. Hence, $\gamma(T) = \gamma(T')$. Similarly, if we have a closed walk outside of T', it does not dominate $V(T')$. However, any minimum closed dominating walk in T' dominates all of T, so $w(T) = w(T')$. □

From the proof of Theorem 2.1.6, it is clear that a watchman's walk in a tournament will always be within the dominating strong component. This result is stated in Corollary 2.1.7. It also leads us to an important result in Theorem 2.1.8.

Corollary 2.1.7. *If T is a tournament, and T' is the dominating component in the condensation of T, any watchman's walk of T' is a watchman's walk of T.*

In the following theorem, we prove that any tournament has a watchman's walk. This fact is not immediately obvious for tournaments that are not strong, as many digraphs that are not strongly connected do not have a watchman's walk. In fact, there are many infinite families of digraphs that have no watchman's walk, including the family of orientations of paths on $n \geq 4$ vertices.

Theorem 2.1.8. *If T is a tournament, then T has a watchman's walk.*

Proof. Let T be a tournament. If T is strong, then by Theorem 2.1.1, T has a watchman's walk. Suppose that T is not strong, and consider the dominating vertex in the condensation of T. This vertex corresponds to the dominating maximal strong component in T. This component T' is strongly connected, so the subtournament T' has a watchman's walk. Since T' has a watchman's walk, T also has a watchman's walk, as the watchman's walk of T' also dominates T. Thus, any tournament T has a watchman's walk. □

Corollary 2.1.9. *If T is a tournament of order $n \geq 5$, then $w(T) \leq n-2$.*

Proof. Let T be a tournament on n vertices. If T is a strong tournament, then $w(T) \leq n-2$ by Theorem 2.1.3. Now suppose that T is not strong, and consider the condensation of T. By Theorem 2.1.6, a watchman's walk of T is contained in the dominating strong component T'. If T' has k vertices, then a watchman's walk in the subdigraph T' has length at most $k-2$. Since this walk would also be a watchman's walk in T, $w(T) \leq k-2 \leq n-2$. □

We know that any tournament that is not strongly connected has more than one maximal strong component. So, by considering the condensation of our tournament, we can obtain a slightly better upper bound on the watchman's number for tournaments that are not strongly connected, compared to the bound given in the previous theorem for any tournament.

Theorem 2.1.10. *If T is a tournament on $n \geq 6$ vertices that is not strongly connected, then $w(T) \leq n-3$.*

Proof. Let T be a tournament of order $n \geq 6$ such that T is not strongly connected. It follows that T has more than one maximal strong component. Suppose that two of these components, T_i and T_j both contain a watchman's walk of T. A watchman's walk in T_i must dominate T_j, and a watchman's walk in T_j must dominate T_i. However, this means that there are arcs from vertices in T_i to vertices in T_j, and from vertices in T_j to T_i. Hence, $T_i \cup T_j$ is a strong component, contradicting their maximality. So, only one maximal strong component must contain every watchman's walk of T. It follows that any vertex in this component must dominate each vertex in every other maximal strong component. Let T' be the maximal strong component that contains some watchman's walk W. We can consider T' as a subtournament. Since T was not strongly connected, $|V(T')| \leq |V(T)| - 1$. The vertices in T' dominate $V(T) \backslash V(T')$, so any watchman's walk of T' is also a watchman's walk of T. By Theorem 2.1.3, $w(T') \leq |V(T') - 2| \leq n-3$. By Theorem 2.1.6, $w(T) = w(T')$, so $w(T) \leq n-3$. □

2.2 Domination number

In general, for both undirected and directed graphs, there can be a large difference between the domination number and the length of a watchman's walk in a graph. In this section, we show that this is not the case for tournaments. We begin by proving a relationship between the domination number and size of a watchman's walk in a tournament. We later give further results relating to domination number.

Theorem 2.2.1. *Let T be a tournament of order $n \geq 3$. If $\gamma(T) > 1$, then $w(T) = \gamma(T)$ or $w(T) = \gamma(T) + 1$.*

Proof. Let T be a tournament of order n. Let $\gamma(T) = k$ for some $1 \leq k < n$. From Theorem 1.1.2, any tournament contains a Hamilton path. Suppose that there is some minimum dominating set $D = \{v_1, v_2, v_3, \ldots, v_k\}$ of T, with a Hamilton path $H = v_1, v_2, v_3, \ldots, v_k$ in the subtournament induced by D such that $v_1 \in N_T^+(v_k)$. In this case (v_k, v_1) is an arc, so $W = v_1, v_2,\ v_3, \ldots, v_k, v_1$ is a closed walk in T. Since D was a minimum dominating set in T, and W is a closed walk that uses exactly the vertices of D, W is a watchman's walk for T. This walk has length k. So, $w(T) = k = \gamma(T)$.

Now suppose that for every minimum dominating set in T, and in every Hamilton path in the subtournament induced by the dominating set, the start vertex of the path is not in the out-neighbourhood of the end vertex. It follows that we cannot construct a closed dominating walk in T that is of length k, as this walk must use only the vertices of a minimum dominating set, and there are no closed walks using exactly the vertices of a minimum dominating set. This means that $w(T) > k = \gamma(T)$. Let $D = \{v_1, v_2, v_3, \ldots, v_k\}$ be a minimum dominating set in T, and let $P = v_1, v_2, v_3, \ldots, v_k$ be a Hamilton path in $H = T[D]$. Consider $N_T^+(v_k) \cap N_T^-(v_1)$. If this is empty, every vertex outside of D that is dominated by v_k is also dominated by v_1, and v_k is dominated by v_1. So, $D \backslash \{v_k\}$ is a dominating set, contradicting the minimality of D. So, there is some vertex u in T, such that $u \notin D$, and $u \in N_T^+(v_k) \cap N_T^-(v_1)$. Thus, $W = v_1, v_2, v_3, \ldots, v_k, u, v_1$ is a closed dominating walk of length $k + 1 = \gamma(T) + 1$. We know that $w(T) > k$, and W has length $k + 1$. Thus, W is a minimum closed dominating walk. So, $w(T) = \gamma(T)+1$. Therefore, for any tournament T, $w(T) = \gamma(T)$ or $\gamma(T) + 1$. □

In the tournament in Figure 2.3, $\gamma(T) = 3$, and $\{v_1, v_4, v_6\}$ is a minimum dominating set. Since $W = v_1, v_2, v_6, v_1$ is a closed walk through this set, $w(T) = 3 = \gamma(T)$.

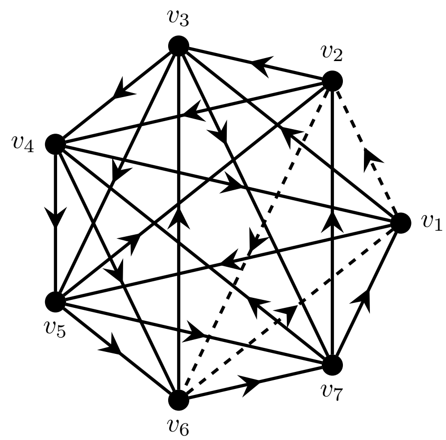


Figure 2.3: A tournament T where $w(T) = \gamma(T)$.

From Theorem 2.2.1, we get the following two corollaries.

Corollary 2.2.2. *If T is a tournament, then no watchman's walk in T repeats vertices.*

Proof. Let T be a tournament. Suppose there is some minimum dominating set D such that the subtournament induced by D, $T[D]$, contains a Hamilton cycle. In this case, $w(T) = |D| = \gamma(T)$. Any watchman's walk must contain a dominating set, and hence a dominating walk of length $\gamma(T)$ must include a minimum dominating set. Since any minimum dominating set contains only distinct vertices, each watchman's walk for T does not repeat any vertices.

Now suppose that there is no minimum dominating set such that the subtournament it induces contains a Hamilton cycle. Any dominating walk in T must use at least $\gamma(T)$ vertices in order to dominate T, and by Theorem 2.2.1, a minimum closed dominating needs at most $\gamma(T) + 1$ vertices. It follows that $w(T) = \gamma(T) + 1$, as we have no closed walk containing exactly the vertices of a minimum dominating set. Let W be any watchman's walk of T, and let D be the set of vertices in the walk. Since D must be a dominating set of T, we have that D is either a minimal dominating set of size $\gamma(T) + 1$, or a minimum dominating set with an additional vertex. In the first case, since W has length $\gamma(T) + 1$ and $|D| = \gamma(T) + 1$, each vertex must be distinct in the walk. Now suppose that D contains a minimum dominating set D' and an additional vertex u. Each vertex in the minimum dominating set in distinct. Suppose that u is in D'. It follows that W is a closed walk in $T[D']$ that uses each vertex in the induced subdigraph. Hence, $T[D']$ is a strong tournament. By Theorem 1.1.3, $T[D']$ has some Hamilton cycle C. Since $T[D']$ is a subtournament of T, C is also a cycle in T. As C contains each vertex in D', it is also a closed dominating walk of T. Thus, as C is a closed dominating walk of length $\gamma(T)$, $w(T) \leq \gamma(T)$. This is a contradiction, so each vertex must be distinct. □

Corollary 2.2.3. *For a tournament T, $\gamma(T) = w(T)$ if and only if there exists some minimum dominating set D in T such that $T[D]$ is itself a strong tournament.*

Proof. Let T be a tournament such that $\gamma(T) = k$, and suppose that for some minimum dominating set D, $T[D]$ is strongly connected. It follows that, by Theorem 1.1.3, $T[D]$ has a Hamilton cycle H. Since $\gamma(T) = k$, we know that $|D| = k$ and H has length k. Since D was a dominating set in T, H is a closed dominating walk in T. Also, we need at least k vertices to dominate T, and H contains exactly k distinct vertices, so H is a minimum closed dominating walk. Thus, $w(T) = k$.

Now suppose that for every minimum dominating set D in T, $T[D]$ is not a strong subtournament. In this case, there is no Hamilton cycle in $T[D]$. So, there is no k-cycle in T containing exactly the vertices of a minimum dominating set. Hence, any cycle of length k does not contain all the vertices of a single minimum dominating set, and since $\gamma(T) = k$, the vertices of this walk cannot dominate T. Thus, any closed walk of length k is not dominating, and $w(T) \neq k$. By Theorem 2.2.1, $w(T) = k$ or $k + 1$, so $w(T) = k + 1 = \gamma(T) + 1$. Therefore, $w(T) = \gamma(T)$ exactly when $T[D]$ is strong for some minimum dominating set D. □

The previous two corollaries describe the structure of a watchman's walk in a tournament. As no vertex is repeated, we know that a watchman's walk is a directed cycle. Recall that the cycle domination number of a digraph D is the length of the shortest directed cycle C in G such that the vertices in C form a dominating set in D. We get the following theorem.

Theorem 2.2.4. *If T is a tournament, then $\gamma_{cyc}(T) = w(T)$.*

Proof. Let T be a tournament. Suppose there is a minimum dominating set D in T such that $T[D]$ is a strong subtournament. By Theorem 1.1.3, this subtournament contains a Hamilton cycle H. This is also a cycle in T, and since D is a minimum dominating set, H is a watchman's walk of T. Thus, $w(T) = \gamma(T)$. Also, since H is a Hamilton cycle in $T[D]$, it is the shortest cycle in T that uses all the vertices of the minimum dominating set D. Hence, H is a minimum dominating cycle of length $\gamma(T)$. Therefore, $\gamma_{\text{cyc}}(T) = \gamma(T) = w(T)$.

Now suppose that there is no minimum dominating set D such that $T[D]$ is a strong subtournament. In this case, if D is any minimum dominating set, $T[D]$ is not Hamiltonian, by Theorem 1.1.3. So, there is no cycle in T that uses only the vertices of a minimum dominating set, and hence any cycle of length $\gamma(T)$ is not dominating. Thus, $\gamma_{\text{cyc}}(T) > \gamma(T)$. By Theorem 2.2.1, T has $w(T) = \gamma(T) + 1$, so T has a closed dominating walk of length $\gamma(T) + 1$. By Corollary 3.3.6, we know that the vertices in a watchman's walk of T are distinct, so a watchman's walk in a T is a cycle. Since these are the shortest dominating cycles, $\gamma_{\text{cyc}}(T) = \gamma(T) + 1 = w(T)$. □

Recall that the total domination number of a digraph D, denoted $\gamma_{\text{t}}(D)$, is the size of a smallest set S such that for every vertex v in D, there is a vertex $u \neq v$ in S such that (u, v) is an arc.

Theorem 2.2.5. *If T is a tournament, then $\gamma(T) \leq \gamma_t(T) \leq w(T)$.*

Proof. Any minimum total dominating set must also be a dominating set, so $\gamma_{\text{t}}(T) \geq \gamma(T)$. If W is a watchman's walk, each vertex in W is dominated by at least one other vertex in the walk, and by Corollary 2.2.2, the vertices of W are unique. The vertices of W form a dominating set in T, so the vertices of W are also a total dominating set in T. Therefore, $\gamma(T) \leq \gamma_{\text{t}}(T) \leq w(T)$. □

The following theorem, from [20], provide bounds on the domination number for small tournaments.

Theorem 2.2.6. [20] *If T is a tournament on n vertices, then $\gamma(T) \leq 2$ if $n < 7$, and $\gamma(T) \leq 3$ if $n < 19$.*

By using these upper bounds on the domination number of small tournaments, we get the following upper bounds on the watchman number.

Corollary 2.2.7. *If T is a tournament on n vertices, then $w(T) \leq 3$ if $n < 7$, and $w(T) \leq 4$ if $n < 19$.*

We know that any watchman's walk for a tournament will be completely contained in a single maximal strong component. From this, it follows that we can also apply these upper bounds to some larger tournaments.

Corollary 2.2.8. *If T is a tournament such that no maximal strong component contains more than six vertices, then $w(T) \leq 3$.*

Proof. Let T be a tournament with no maximal strong component with more than six vertices. If T is strongly connected, then T has exactly one strongly connected component, so $|V| < 7$, and the result follows from Corollary 2.2.7. Now suppose that T is not strongly connected. That is, T contains at least two maximal strong components, each with less than seven vertices. Since T is a tournament, the condensation of T is transitive. Hence, T has some dominating maximal strong component. Let this component be T'. Consider T' as a subtournament of T. By Theorem 2.1.6, $w(T) = w(T')$. Since T' contains less than seven vertices, $w(T') \leq 3$. Therefore $w(T) \leq 3$. □

Similarly, for the second upper bound, we get the following corollary.

Corollary 2.2.9. *If T is a tournament such every maximal strong component contains fewer than 19 vertices, then $w(T) \leq 4$.*

Due to the high arc density of tournaments, we might initially expect that the dominating number of a tournament would remain low, regardless of the order of the tournament. However, the following theorem, from [8], tells us that there exists a tournament with domination number k for any arbitrarily large value of k.

Theorem 2.2.10. [8] *For every $k \in \mathbb{Z}^+$, there exists a tournament T such that $\gamma(T) = k$.*

For many uses and constructions, it is more helpful to have a strong tournament over one that is not strong, due to their useful structure. While, from Theorem 2.2.10, we have that there exists some tournament T with domination number k, this theorem does not guarantee to give a strong tournament. So, it is not immediately clear that this is true for strong tournaments. Thus, we give the following construction to build a strong tournament from any tournament that is not already strong, while preserving the dominating number. This can be used to obtain a strong tournament with domination number k for any integer k.

Lemma 2.2.11. *Let T be any tournament of order n that is not strongly connected. If $\gamma(T) = k \geq 3$, then we can construct a strongly connected tournament, T_s, of order $n+1$ such that $\gamma(T_s) = k$ and T is a subtournament of T_s.*

Proof. Let T be a tournament with vertices $V(T)$, and arc set $A(T)$ such that T is not strongly connected. Since T is a tournament, we know from Theorem 1.1.2 that T has at least one Hamilton path. Let $H = u, v_1, v_2, \ldots, v_{n-1}$ be a Hamilton path in T. Consider the tournament T_s, where $V(T_s) = V(T) \cup \{w\}$, and $A(T_s) = A(T) \cup \{(x, w) | x \in V(T) \backslash \{u\}\} \cup \{(w, u)\}$ for some $w \notin V(T)$. It follows that $H_s = \{u, v_1, v_2, \ldots, v_{n-1}, w, u\}$ is a closed walk in T_s. Since H uses each vertex in T exactly once, H_s uses each vertex in T_s exactly once, except u, which is both the start-vertex and end-vertex of the walk. Thus, H_s is a Hamilton cycle in T_s. Hence, by Theorem 1.1.3, T_s is strongly connected.

Now suppose D is a minimum dominating set in T. Since $\gamma(T) = k \geq 3$, we know $|D| = k \geq 3$. As w is dominated in T_s by all but one vertex from T, at least two vertices in D dominate w in T_s. Also, $A(T) \subset A(T_s)$, so D must also be a dominating set in T_s. Thus, $\gamma(T_s) \leq k$.

Consider a minimum dominating set D_s in T_s. We know $|D_s| \leq k$, and we need to show that $|D_s| = k$. Suppose to the contrary that $|D_s| < k$. Suppose that $D_s \subset V(T)$. Since $A(T) \subset A(T_s)$, D_s would be a dominating set in T. However, $|D_s| \leq k-1$, so D_s doesn't dominate T. Thus, it also doesn't dominate T_s. This means that $D_s \not\subset V(T)$ and $w \in D_s$. By the construction of T_s, vertex w only dominates w and u in T_s. So, $D_s \backslash \{w\}$ must dominate $T \backslash \{u\}$. It follows that $D_s \backslash \{w\} \cup \{u\}$ must dominate T. However, $|D_s \backslash \{w\} \cup \{u\}| \leq k - 1$. This contradicts the fact that $\gamma(T) = k$. Thus, $\gamma(T_s) \geq k$. Therefore $\gamma(T_s) = k$. □

Theorem 2.2.12. *For every integer $k \geq 2$, there exists a strong tournament T such that $\gamma(T) = k$.*

Proof. Let $k = 2$. The directed 3-cycle is a strong tournament of order 3 with $\gamma(T) = 2$. Now let k be any integer such that $k \geq 3$. By Theorem 2.2.10, there exists a tournament T such that $\gamma(T) = k$. If T is strongly connected, we have the desired tournament. However, if T is not strong, by Theorem 2.2.11, there exists a tournament T_s from T such that T is strong and $\gamma(T_s) = k$ as required. □

Due to the density and structure of tournaments, it would be natural to expect that the watchman number of a tournament would remain relatively low. However, in the following theorem, we prove that, as with domination number, there exists a tournament with watchman number k, for any integer k.

Theorem 2.2.13. *If $k \geq 3$, then there exists a tournament T such that $w(T) = k$.*

Proof. Let k be an integer such that $k \geq 3$. We will construct a tournament T with $w(T) = k$ using the construction given in Theorem 2 in [8] for a tournament with $\gamma(T) = k$.

Let T_p be the tournament on p vertices, $\{0, 1, \ldots, p-1\}$, such that (i, j) is an arc if and only if $i - j$ is a quadratic residue of p. Let p be a prime congruent to 3 modulo 4 such that $\gamma(T_p) > k$.

Now, let $J_k(T_p)$ be the tournament constructed as follows:

- Relabel the vertices of T_p as $\{(0, 0), (1, 0), \ldots, (p-1, 0)\}$. Take k copies of T_p, labelled $T_1, T_2, \ldots, T_k$. Let the vertices of T_i be $\{(0, i), (1, i), \ldots, (p-1, i)\}$.
- $((a, i), (b, j))$ is an arc in $J_k(T_p)$ if and only if $((a, 0), (b, 0))$ is an arc in T_p or $a = b$ and $i < j$.

Let X be a tournament on $k+m-1$ vertices, where $m \geq 1$ is any natural number. Let $V(X) = \{x_{1,1}, x_{1,2}, \ldots, x_{1,m}, x_2, \ldots, x_k\}$. For each j such that $2 \leq j \leq k$, we will also let x_j be denoted as $x_{j,r}$ for every r such that $1 \leq r \leq m$. For each r such that $1 \leq r \leq m$, $X_r = \{x_{1,r}, x_2, \ldots, x_k\} = \{x_{1,r}, x_{2,r}, \ldots, x_{k,r}\}$. Let $M = \{x_{1,1}, x_{1,2}, \ldots, x_{1,m}\}$. We want to choose the arcs of X such that $X \setminus M$ dominates each vertex in M. To do this, we can let $(x_3, x_{1,r})$ be an arc for all r such that $1 \leq r \leq m$. Now, since $x_3 \notin M$, $X \setminus M$ dominates each vertex in M. From the construction given in Theorem 1 in [8], we know that the remaining arcs can be defined in any orientation. We can now define these arcs to give a strong tournament for the subdigraph induced by X_r for each r. To do this, we can let (x_i, x_{i+1}) be an arc in X for each i such that $2 \leq i \leq k-1$, and $(x_k, x_{1,r})$ be an arc. Now, $X[X_r]$ has a Hamilton cycle. Hence, by Theorem 1.1.3, it is strong.

We can now construct T. First, let $V(T) = V(J_k(T_p)) \cup V(X)$. Define the arc set of T such that any arc in $J_k(T_p)$ or in X is also an arc in T. For the remaining arcs,

$(x_{i,r}, y)$ is an arc for $x_{i,r} \in X$ and $y \in J_k(T_p)$ if $y \in T_i$, otherwise $(y, x_{i,r})$ is an arc. Each X_r now dominates the vertices of $T_k(T_p)$ since $x_{i,r}$ dominates the vertices of T_i for $q \leq i \leq k$, and X_r dominates each vertex in X. Hence, X_r is a dominating set in T. Now we must show that X_r is a minimum dominating set.

Suppose D is a dominating set that does not contain any X_r. In this case, either $D \cap M = \emptyset$ or $D \cap M \neq \emptyset$. If $D \cap M = \emptyset$, then no $x_{1,i}$ is in D, so no element of T_1 is dominated by $D \cap X$. If we have that $D \cap M \neq \emptyset$, then there is some j where $2 \leq j \leq k$ such that x_j is not in D. Otherwise, we would have that $X_r \subseteq D$. Thus, T_j is not dominated by $D \cap X$. So, there must be some t such that $1 \leq t \leq k$ where no element of T_t is dominated by $D \cap X$. Hence, T_t must be dominated by elements of $J_k(T_p)$. However, $|J_k(T_p) \cap D| \geq \gamma(T) > k$. So, $|D| > k$. Hence, any D is a minimum dominating set if and only if it is X_r for some r such that $1 \leq r \leq m$. Since each X_r induces a strong tournament, and $|X_r| = k$, by Theorem 2.2.3, $w(T) = k = \gamma(T)$. □

In the construction given in the proof of Theorem 2.2.13, the choice of m determines the number of distinct watchman's walks in T. From this, we get the following corollary.

Corollary 2.2.14. *For any integers $k \geq 3$, and $m \geq 1$ there exists a tournament T such that $w(T) = k$ and T has exactly m watchman's walks. Moreover, there exists a tournament T such that T has a unique watchman's walk of length k.*

2.2.1 Computational results

Even when considering only those of small order, the number of non-isomorphic tournaments on a given number of vertices can be very large. As a result, the length, structure, and multiplicity of watchman's walks can vary greatly, even between tournaments on the same number of vertices. In this section, we present and summarize computational results on the watchman number and domination number in tournaments of order up to 10. Table 2.1 presents a summarized collection of this data. In this table, we specify the order, length of watchman's walks, and domination number, and give the number of tournaments with the stated parameters. In Appendix B, we include tables with additional results. These tables contain computational data regarding the watchman's walks in all tournaments of order at most 10, using the collection of adjacency matrices given in [16]. For each order, we specify the watchman number, domination number, and multiplicity of watchman's walks, and give the number of tournaments that satisfy the given values of $w(T)$, $\gamma(T)$, and watchman walk multiplicity. These results were found using the program in Appendix A. In this section, we also state important cases and observations that follow from Table 2.1 and the tables in Appendix B, as well as give a relation for the watchman's walks on tournaments.

Order	$w(T)$	$\gamma(T)$	Number of tournaments
2	0	1	1
3	0	1	1
	3	2	1
4	0	1	2
	3	2	2
5	0	1	4
	3	2	8
6	0	1	12
	3	2	44
7	0	1	56
	3	2	399
	3	3	1
8	0	1	456
	3	2	6419
	3	3	5
9	0	1	6880
	3	2	184430
	3	3	226
10	0	1	191536
	3	2	9511704
	3	3	29816

Table 2.1: Summary of computational results for tournaments of order up to 10

Observation 2.2.15. *There is exactly one tournament of order 7 with domination number 3. This tournament, as illustrated in Figure 2.4, is the Paley tournament of* *order 7.*

Let $q = 3(\text{mod } 4)$ be a prime power. Consider the finite field of order q, F_q. The *Paley tournament* is the digraph with vertex set $V = F_q$, where (a, b) is an arc if $b - a \in (F_q)^2$. The Paley tournament of order 7 is given in Figure 2.4.

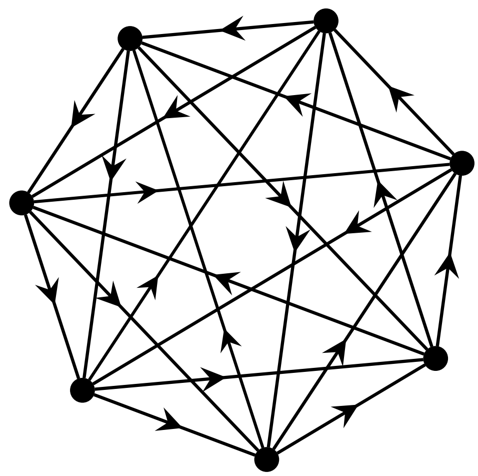


Figure 2.4: Paley tournament on 7 vertices

From Table 2.1, we can see that this tournament is the smallest tournament with domination number 3. If we recall Theorem 2.2.6, which tells us that any tournament on less than seven vertices has domination number less than three, it is clear that this should be the case; our computations confirm this. Similarly, none of the tournaments of order up to 10 have domination number 4, as required by Theorem 2.2.6. If we recall, from Theorem 2.2.1, that $w(T) \leq \gamma(T)+1$, we see that any tournament on less than seven vertices should have a watchman number of at most three. Again, Table 2.1 confirms this claim.

If we consider Table 6 in Appendix B, we can easily identify other unique tournaments.

Observation 2.2.16. *There is a unique tournament T on seven vertices with domination number two such that T has fourteen watchman's walks. This graph is illustrated in Figure 2.5.*

Not only is this tournament the unique tournament with its given parameters, but it also has the highest multiplicity of watchman's walks for the tournaments on seven vertices.

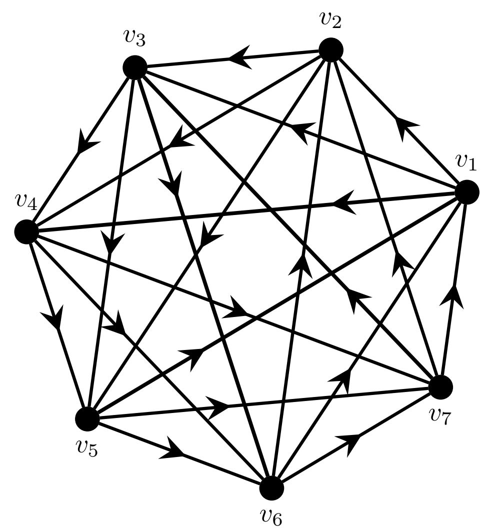


Figure 2.5: Tournament of order 7 with 14 watchman's walks

If we consider the number of tournaments for each set of parameters in the tables in Appendix B, we also get the following two observations. We let $T(n)$ be the set of non-isomorphic tournaments on n vertices.

Observation 2.2.17. *The total number of tournaments of order n is equal to the number of tournaments of order $n+1$ that have a dominating vertex.*

We would expect this to be the case. Consider the set of tournaments $T(n)$. If we add a source vertex to each tournament in this set, we get $T(n)$ non-isomorphic tournaments of order $n+1$, each having a domination number of 1 and a watchman's walk of length 0.

Similarly, if we consider a tournament on $n-1$ vertices with domination number γ, watchman number w, and watchman multiplicity m, we can add a sink vertex to the tournament to get a new tournament on n vertices with the same parameters. We can now define a relation for the domination number, watchman number, and multiplicity of watchman's walks for tournaments of order n. We let $F(n, w, m, \gamma)$ be the number of tournaments of order n with watchman number w, watchman multiplicity m, and domination number γ. The next two theorems follow from the observation.

Theorem 2.2.18. *If $T(n)$ is the set of tournaments on $n \geq 2$ vertices, then $|T(n)| = F(n+1, 0, 1, 1)$.*

Proof. Consider the tournaments in $T(n)$. If we add a source vertex to each tournament $T \in T(n)$, we get a new set of tournaments $T'(n)$, each of order $n+1$. The tournaments in $T(n)$ were pairwise non-isomorphic, so the tournaments in $T'(n)$ will also be non-isomorphic, and $|T(n)| = |T'(n)|$. Each of these tournaments have a source vertex, so they have watchman number 0, watchman multiplicity 1, and domination number 1. Hence, $|T(n)| \leq F(n+1, 0, 1, 1)$. Now, consider some tournament of order $n+1$ with watchman number 0, watchman multiplicity 1, and domination number 1. This tournament must have some source vertex v. Since $T\backslash\{v\}$ is a subtournament on n vertices, $T\backslash\{v\} \in T(n)$. Also, the tournaments in $T(n+1)$ are pairwise non-isomorphic, so the subset of $T(n+1)$ containing the tournaments with a source vertex are also distinct. So, for any $T_1 \in T(n+1)$ with source v_1, and $T_2 \in T(n+1)$ with source v_2, $T_1\backslash\{v_1\}$ and $T_2\backslash\{v_2\}$ are non-isomorphic and are in $T(n)$. Thus, $|T(n)| \geq F(n+1, 0, 1, 1)$ and $|T(n)| = F(n+1, 0, 1, 1)$. □

Theorem 2.2.19. *If* $n \geq 2$, $F(n, w, m, \gamma) \leq F(n+1, w, m, \gamma)$.

Proof. Consider the set $S \subset T(n)$ of tournaments of order n with watchman number w, watchman multiplicity m, and domination number γ. If we take each tournament $T \in S$ and add a sink vertex u, we get a tournament T' of order $n+1$ with T as a subtournament. As u is a sink, it dominates no vertices besides itself, and cannot be included in a nontrivial closed walk. So, we will get no new dominating sets or watchman's walks in T'. However, since u is dominated by any of the vertices that were in T, any dominating set D in T is also a dominating set in T'. Similarly, any watchman's walk for T will also be a watchman's walk for T'. So, T' has watchman number w, watchman multiplicity m, and domination number γ. As the tournaments in S were pairwise non-isomorphic, the tournaments in S' are also non-isomorphic. So, $F(n, w, m, \gamma) \leq F(n+1, w, m, \gamma)$. □

We also get a similar recurrence relation when we disregard the multiplicity of watchman's walks. We let $F'(n, w, \gamma)$ be the number of tournaments of order n with watchman number w, and domination number γ. It is straightforward to see why the results in Theorem 2.2.18 and Theorem 2.2.19 also hold for this relation. For Theorem 2.2.18, it is not necessary to consider multiplicity, as a tournament can only have a single source vertex. We have the following corollary of Theorem 2.2.19.

Corollary 2.2.20. *If $n \geq 2$, then $F'(n, w, \gamma) \leq F'(n+1, w, \gamma)$.*

Proof. Let $n \geq 2$ and let $M = \{m_1, m_2, \ldots, m_k\}$ be all possible multiplicities of watchman's walks for tournaments of order n with watchman number w and domination number γ. It follows that

$$\sum_{m \in M} F(n, w, \gamma, m) = F'(n, w, \gamma).$$

Since, for each value of m, $F(n, w, \gamma, m) \leq F(n+1, w, \gamma, m)$, we get that

$$F'(n, w, \gamma) = \sum_{m \in M} F(n, w, \gamma, m) \leq \sum_{m \in M} F(n+1, w, \gamma, m) = F'(n+1, w, \gamma).$$

□

Given the analogous relation in Theorem 2.2.19, it is not surprising that this relation holds. However, the relations given in both Theorem 2.2.19 and Corollary 2.2.20 are both particularly interesting when we consider tournaments of higher order. These relations tell us that the number of tournaments of order n with a given watchman number and dominating number increases (not necessarily strictly) as we increase n. This holds even for arbitrarily small watchman numbers and domination numbers. This is surprising, however, since we might expect fewer tournaments of higher order to have small watchman and domination numbers.

2.3 Families of tournaments

Unlike for complete graphs, there are a large number of non-isomorphic tournaments on a given number of vertices. For example, on 10 vertices, there are 9733056 tournaments ([16]). Hence, these tournaments can look very different, and their domination or watchman numbers can vary greatly. For many tournaments, however, we would expect their watchman numbers to be low. For many families, we can prove that this is true. In this section, we consider families of tournaments or tournaments having certain characteristics, in order to offer more precise results on the length of their watchman's walks.

2.3.1 Simple tournaments

The *score sequence* of a tournament is the sequence of all vertex out-degrees, typically in non-decreasing order. A tournament with score sequence S is said to be *simple* if there are no other non-isomorphic tournaments with score sequence S. That is, T is the unique tournament with score sequence S. The following theorems demonstrate that any simple tournament has a small watchman number.

Theorem 2.3.1. [22] *A tournament T is simple if and only if every strong component of T has score sequence $(0), (1,1,1)$, $(1,1,2,2)$, or $(2,2,2,2,2)$.*

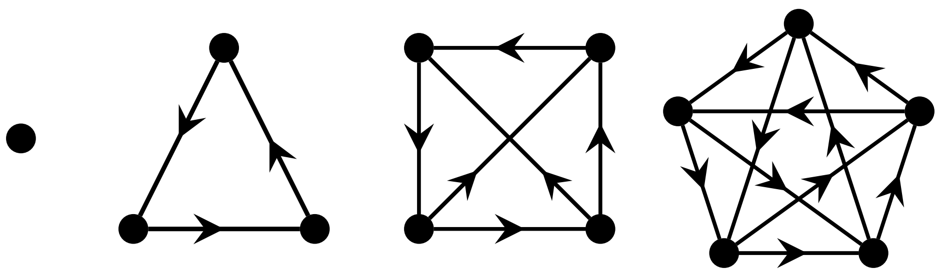


Figure 2.6: Strong components of simple tournaments

Theorem 2.3.2. *If T is a simple tournament, then $w(T) = 0$ or 3.*

Proof. Let T be a tournament with score sequence S such that for any other tournament T' with score sequence S, $T \simeq T'$. Consider the condensation of T. By Theorem 2.3.1, every strong component of T has score sequence (0), $(1,1,1)$, $(1,1,2,2)$, or $(2,2,2,2,2)$. So, the strong components of T have score sequence (0), $(1,1,1)$, $(1,1,2,2)$, or $(2,2,2,2,2)$. It is clear that the tournament defined by (0) has a dominating vertex, and has a watchman's walk of length 0. None of the other score sequences define a tournament with a dominating vertex, and each define a tournament with a dominating walk of length 3, so each of the tournaments with score sequence $(1,1,1)$, $(1,1,2,2)$, or $(2,2,2,2,2)$ has a watchman's walk of length 3. Since the dominating component of T is a strong subtournament, it has one of these score sequences, and we know by Theorem 2.1.6 that $w(T) = 3$. □

2.3.2 Transitive tournaments

Recall that a tournament $T = (V, A)$ is transitive if there exists an ordering of the vertices of T, $(v_1, v_2, \ldots, v_n)$ such that $(v_i, v_j) \in A(T)$ if and only if $i < j$. These tournaments are always acyclic. We can also have tournaments whose structure closely resembles that of a transitive tournament, but do not have such an ordering. Three such families of tournaments are locally transitive, locally-in-transitive, and locally-out-transitive tournaments.

A tournament T is *locally-transitive* if for each vertex v in T, both the in-neighbourhood and out-neighbourhood of v are transitive sub-tournaments. The tournament in Figure 2.7 is locally-transitive. However, it is not a transitive tournament as it contains 3-cycles.

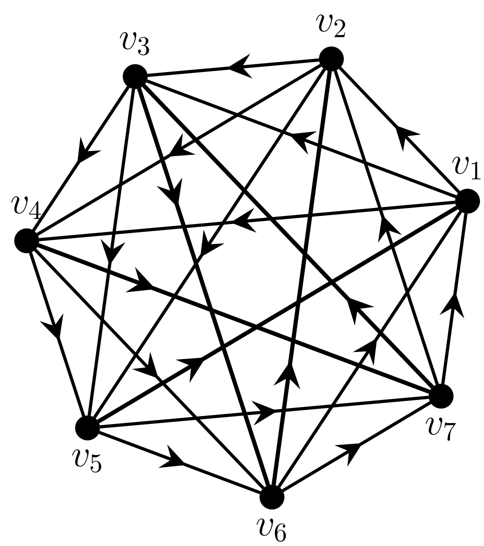


Figure 2.7: A locally-transitive tournament that is not transitive

A tournament T is *locally-in-transitive* if for each vertex v in T, the subdigraph induced by the in-neighbourhood of v is a transitive sub-tournament. Similarly, a tournament is *locally-out-transitive* if for each vertex v in T, the subdigraph induced by the out-neighbourhood of v is a transitive sub-tournament.

Recall that any transitive tournament has a dominating vertex. This is not necessarily the case for tournaments that are locally-in-transitive or locally-out-transitive. However, due to their ordering, or similar structure to transitive tournaments, we would expect these tournaments to have small dominating sets, and hence, small watchman's walks. In this section, we prove this idea to be correct.

Theorem 2.3.3. *If T is a locally-in-transitive or locally-out-transitive tournament, then $\gamma(T) \leq 3$ and $w(T) \leq 3$.*

Proof. Let T be a tournament, and let $\gamma = \gamma(T)$. By Theorem 2.2.1, $w(T) = \gamma$ or $\gamma + 1$. Suppose $\gamma \geq 4$, and $w(T) = \gamma$. Let $W = v_1, v_2 \ldots, v_\gamma, v_1$ be a watchman's walk in T, and let D be the set of vertices in W. Since $w(T) = \gamma(T)$, D is a minimum dominating set of T. Each vertex in D is dominated by at least one other vertex in D, namely the vertex that precedes it in the closed walk W. So, each vertex in D has at least one private out-neighbour in $V(T \backslash D)$ with respect to the dominating set. Let p_i be a private neighbour of v_i for each $1 \leq i \leq \gamma$. It follows that v_1, p_1, and $p_{\gamma-1}$ are all in-neighbours of v_2. So, if T is a locally-in-transitive tournament, $T[\{v_1, p_1, p_{\gamma-1}\}]$ is a transitive subtournament of T. Since $(p_{\gamma-1}, v_1)$ and (v_1, p_1) are both arcs, $(p_{\gamma-1}, p_1)$ must be an arc in T. Now, $v_{\gamma-1}, p_{\gamma-1}$, and p_1 are all in-neighbours of v_γ, but $v_{\gamma-1}, p_{\gamma-1}, p_1, v_{\gamma-1}$ is a cycle in the in-neighbourhood of v_γ. However, this cannot happen since T is locally-in-transitive. Thus, if T is locally-in-transitive, we cannot have that $\gamma \geq 4$ when $w(T) = \gamma$. Similarly, $v_2, \ldots, v_{\gamma(T)}$, are all out-neighbours of p_1. So, if T is locally-out-transitive, $T[\{v_2, \ldots, v_{\gamma(T)}\}]$ is a transitive subtournament of T. Since $v_2, v_3, \ldots, v_\gamma$ is a path in this subtournament, (v_2, v_γ) must be an arc. Now, v_1, v_2 and v_γ are all out-neighbours of p_2. However, since (v_2, v_γ) and (v_γ, v_1) are both arcs, we have that $v_\gamma, v_1, v_2, v_\gamma$ is a cycle. However, this cannot happen since T is locally-out-transitive. Thus, if T is locally-out-transitive, we cannot have that $\gamma(T) \geq 4$ when $w(T) = \gamma(T)$.

Now let $\gamma \geq 4$ and $w(T) = \gamma + 1$. Suppose W is a watchman's walk in T, and let D denote the set of vertices in W. Since W is a dominating walk, D is either a minimal dominating set of size $\gamma + 1$, or D contains as a proper subset a minimum dominating set. Suppose first that D is a minimal dominating set. Similar to the case when $w(T) = \gamma$, each vertex in D has at least one private out-neighbour in $V(T \backslash D)$ with respect to the dominating set. Let p_i be a private neighbour of v_i for each $1 \leq i \leq \gamma + 1$. Also, $v_2, \ldots, v_\gamma$, are all out-neighbours of p_1. So, if T is locally-out-transitive, $T[\{v_2, \ldots, v_\gamma\}]$ is a transitive subtournament of T. Since $v_2, v_3, \ldots, v_\gamma$ is a path in this acyclic subtournament, (v_2, v_γ) must be an arc. Now, v_1, v_2 and v_γ are all out-neighbours of p_2. However, we have that $v_\gamma, v_1, v_2, v_\gamma$ is a cycle. This is a contradiction. Similarly, v_1, p_1, and p_γ are all in-neighbours of v_2. So, if T

is a locally-in-transitive tournament, $T[\{v_1, p_1, p_\gamma\}]$ is a transitive subtournament of T. Since (p_γ, v_1) and (v_1, p_1) are both arcs, $(p_{\gamma-1}, p_1)$ must be an arc in T. Now, $v_{\gamma-1}, p_{\gamma-1}$, and p_1 are all in-neighbours of v_γ, but $v_{\gamma-1}, p_{\gamma-1}, p_1, v_{\gamma-1}$ is a cycle in the in-neighbourhood of v_γ. This is a contradiction, since T is locally-in-transitive.

Now suppose that D contains as a subset a minimum dominating set. Let $D' = \{v_1, v_2, \ldots, v_{\gamma-1}, v_\gamma\}$ be a minimum dominating set contained in D. We know that $T[D']$ is a subtournament of T. Since D' is a minimum dominating set, any Hamilton cycle in $T[D']$ would be a closed dominating walk of length γ in T. However, $w(T) > \gamma$, so no such cycle exists in the subtournament. However, by Theorem 1.1.2, $T[D']$ has at least one Hamilton path H. Label the vertices of D' such that $H = v_1, v_2, \ldots, v_{\gamma-1}, v_\gamma$ is a path. Since $T[D']$ is a subdigraph of T, H is also a path in T. Since $T[D']$ has no Hamilton cycle, H is not a closed walk and (v_1, v_γ) is an arc. Each vertex $v_i \in D$ such that $i \neq 1$ is dominated by vertex v_{i-1}, so each vertex in $D' \backslash \{v_1\}$ has at least one private out-neighbour. If p_γ is a private out-neighbour of v_γ, then p_γ dominates all vertices in D' except for v_γ. In particular, it dominates v_1 and v_2. If v_1 does not have a private out-neighbour, then $\{v_2, v_3, \ldots, v_\gamma, p_\gamma, v_2\}$ is a closed dominating walk of length γ. This is a contradiction, as $w(T) = \gamma + 1$. So v_1 must have at least one private out-neighbour. Let p_i be a private out-neighbour of v_i for each $1 \leq i \leq \gamma$. Vertices $p_{\gamma-1}, v_1$, and $v_{\gamma-1}$ are all in-neighbours of v_γ, since (v_1, v_γ) is an arc. If T is locally-out-transitive, we know $T[\{v_\gamma - 1, p_{\gamma-1}, v_1\}]$ is a transitive subtournament of T, and $(v_{\gamma-1}, p_{\gamma-1})$ and $(p_{\gamma-1}, v_1)$ are both arcs, so $(v_{\gamma-1}, v_1)$ is an arc. However, p_1 is also an in-neighbour of v_γ, so $T[\{v_{\gamma-1}, p_1, v_1\}]$ is also a transitive subtournament, and $v_{\gamma-1}, p_1, v_1, v_{\gamma-1}$ is a cycle. So, if T is locally-in-transitive and $\gamma \geq 4$, $w(T) \neq \gamma + 1$.

Similarly, p_i dominates v_j for all $v_j \in D$ and $j \neq i$. Thus, we can see that $T[\{v_2, \ldots, v_\gamma]$ is a transitive subtournament if T is locally-out-transitive, since $v_2, \ldots,$ v_γ are all out-neighbours of p_1. So, we know that (v_i, v_j) is an arc for all $1 < i < j \leq \gamma$. By considering the out-neighbours of p_γ, which include $v_1, \ldots, v_{\gamma(T)-1}$, it is clear that (v_1, v_i) is also an arc for $2 \leq i \leq \gamma(T) - 1$. If (p_2, p_3) is an arc, then v_1, p_3, and v_3 are out-neighbours of p_2, and v_1, v_3, p_3, v_1 is a cycle in the out-neighbourhood of p_2. However, this out-neighbourhood should induce a transitive subtournament. Thus, (p_2, p_3) cannot be an arc. So, (p_3, p_2) must be an arc. However, p_2, v_2, and v_1 would be out-neighbours of p_3, and v_1, v_2, p_2, v_1 is a cycle. So, (p_3, p_2) also cannot be an arc if T is locally-out-transitive. This is a contradiction, since T is a tournament. So, we cannot have that $\gamma(T) \geq 4$ when $w(T) = \gamma(T) + 1$ if T is locally-out-transitive. $\square$

Since each locally-transitive tournament is both locally-in-transitive and locally-out-transitive, we get the following bound on the watchman number for locally-transitive tournaments.

Corollary 2.3.4. *If T is a locally-transitive tournament, then $\gamma(T) \leq 3$, and $w(T) = 0$ or 3.*

Chapter 3

Other digraphs

When considering the watchman's walk problem on connected undirected graphs, we know that we will always find at least one closed dominating walk, and hence a minimum closed dominating walk. However, we cannot always guarantee this for directed graphs. Recall that a digraph is strongly connected if there is a directed path from u to v for any vertices u and v in our graph. It follows that, if our graph is strongly connected, there exists a walk in our graph that visits every vertex. We will prove, in Theorem 3.1.1, that any strongly connected digraph has a watchman's walk. We begin with some basic results on the existence of watchman's walks in general digraphs, and later consider the watchman's walk problem on specific families of digraphs.

3.1 Watchman's walks in general digraphs

In Chapter 2, we proved that any tournament has a watchman's walk. In this section, we begin with results on the existence of watchman's walks in digraphs. In the following theorem, we generalize the result in Corollary 2.1.7 for tournaments to general digraphs.

Theorem 3.1.1. *If D is a digraph and D has some strongly connected subdigraph D' such that the vertices of D' are a dominating set in D, then D has a watchman's walk.*

Proof. Suppose D' is a subdigraph of D such that D' is a strongly connected digraph.

Since D' is strongly connected, there exists a directed path between any pair of vertices in D'. So,there exists a closed walk W that passes through every vertex in D', possibly repeating some vertices. It follows that W is a dominating walk in D', but is not necessarily of minimal length. As D' is a subdigraph of D, W is also a walk in D. Moreover, since the vertices of D' are a dominating set in D, and W uses every vertex in D', W is a dominating walk in D. Hence, D has a closed dominating walk, and therefore it has some minimum closed dominating walk. □

We now know that any strongly connected digraph has a watchman's walk, regardless of its structure. For the remainder of this section, we consider digraphs that are not necessarily strongly connected. Recall that a source vertex in a digraph is a vertex with no in-neighbours. Any nontrivial digraph with a source vertex is not strongly connected, as there is no nontrivial walk that ends at a source vertex. While a digraph may have a trivial watchman's walk including just a source vertex, source vertices do not guarantee the existence of watchman's walks in a digraph. Consider a digraph D with $k > 1$ source vertices. Let u and v be source vertices in D. Both u and v have no in-neighbours. Hence, no non-trivial closed walk contains or dominates both u and v. So, D has no watchman's walk if it has more than one source vertex. If $|D| = n$ and D has a single source vertex v, it must have out-degree $n - 1$ for a watchman's walk to exist. Otherwise, a watchman's walk for D would have to be a non-trivial walk, but no non-trivial walk would contain or dominate v. Thus, we get the following theorem.

Theorem 3.1.2. *If D is a digraph of order n with at least one source vertex, then D has a watchman's walk if and only if D has exactly one source vertex v and* $\deg^+(v) = n - 1$.

If we take an orientation of any path on at least four vertices, we get a digraph that is not strongly connected. Thus, it is not guaranteed that this digraph will have a watchman's walk. In fact, we will demonstrate in the following theorem that no such orientation has a watchman's walk.

Theorem 3.1.3. *If D is an orientation of P_n for $n \geq 4$, then D does not have a watchman's walk.*

Proof. Let D be an orientation of P_n for $n \geq 4$. It follows that the out-degree of any vertex in D is at most 2. Thus, D does not have a dominating vertex, and any closed dominating walk W in D must be a nontrivial walk. Since D is an orientation, it has no directed 2-cycles. That is, there is at most one arc between any pair of vertices. Additionally, since the underlying undirected graph was acyclic, D has no directed cycles and no non-trivial walk can be closed. Therefore, D does not have a watchman's walk. □

While no orientation of paths of length at least 4 has a watchman's walk, it is possible that an orientation of a n-cycle will have a watchman's walk. We will next consider when a watchman's walk will exist in such a digraph. Since an undirected cycle contains a single closed walk, we would expect that it is unlikely that an orientation of a cycle has a watchman's walk. Indeed, we prove that to be correct in the following theorem.

Theorem 3.1.4. *If D is an orientation of C_n for $n \geq 4$, with vertices $\{v_1, v_2, \ldots, v_n\}$, then D has a watchman's walk if and only if the arcs of D are oriented as a directed n-cycle.*

Proof. Let D be an orientation of C_n, for $n \geq 4$. Each vertex in C_n has degree 2. It follows that, for each vertex v in D, v has at most 2 out-neighbours. So, D has no dominating vertex. Thus, any closed dominating walk W in D is non-trivial. Also, as D has no 2-cycles, if W is closed then W must correspond to a cycle in the underlying undirected graph. The underlying undirected graph $D' = C_n$ has exactly one cycle, namely the entire n-cycle. Thus, if W is a closed dominating walk in D, W must be an n-cycle. This is only possible if the arcs of D are oriented as a directed n-cycle.

Now suppose that D is a directed n-cycle. It follows that D is strongly connected. Thus, by Theorem 3.1.1, D has a watchman's walk. □

3.2 Orientations of complete multi-partite graphs

In this section, we will consider the existence of watchman's walks in orientations of complete bipartite and multipartite graphs. Like tournaments, these digraphs have a high density of arcs. It would be reasonable to expect that these digraphs often have a watchman's walk. However, unlike in tournaments, this is not always the case. Consider, for example, a complete multipartite graph with more than one source vertex. In the following observation, we consider the existence of watchman's walks in multipartite graphs with at least one source vertex.

Observation 3.2.1. *Let $D = (X_1 \cup \ldots \cup X_k, A)$ be an orientation of a complete multipartite graph for $k \geq 2$, where $X_1, \ldots, X_k$ are the vertex sets of the partition. If $v \in X_i$ is a source vertex for any $1 \leq i \leq k$, then D has a watchman's walk if and only if $|X_i| = 1$.*

This fact is clear if we consider the set of vertices X_1 containing v. If there is another vertex $u \in X_1$, it is not dominated by v. Also, v is not dominated by any vertex other than itself. Since no walk of nontrivial length contains or dominates v, we cannot have a walk that dominates both u and v. However, if $|X_1| = 1$, v dominates all vertices in the graph, so we get a trivial dominating walk at v. What happens if D does not have a source vertex? We begin by considering orientations of complete bipartite graphs.

Theorem 3.2.2. *Let T be an orientation of a complete bipartite graph with partition (A, B) such that $\delta^-(D) \geq 1$. If there is a set of vertices $U \subseteq B$ such that each vertex in A is dominated by exactly one vertex in U, then T has a watchman's walk.*

Proof. Let T be an orientation of a $K_{m,n}$ such that each vertex has at least one in-neighbour. Let $V(T) = A \cup B$, such that $|A| = m$ and $|B| = n$. Let $A = \{v_1, v_2, \ldots, v_m\}$. Note that the vertices of A dominate U, since each vertex in U has at least one in-neighbour from A. Similarly, the vertices of U dominate A. So, to find a closed dominating walk, we will construct a walk that visits each vertex of A and each vertex of U.

By definition, each vertex in U has a private out-neighbour in A. Thus, $|U| = m$; let $U = \{u_1, u_2, \ldots, u_m\}$. Label the vertices of U such that vertex u_i dominates v_1. So, u_1 dominates v_1. It follows that v_i dominates each vertex in $U \backslash \{u_i\}$. Thus, we can consider the closed walk $W = \{u_1, v_1, u_2, v_2, \ldots, u_m, v_m, u_1\}$. The walk W uses every vertex in A, and each vertex in B is dominated by at least one vertex in A. Hence, W is a closed dominating walk in D. □

Theorem 3.2.2 tells us that there exists orientations of complete bipartite graphs that have a watchman's walk. We would like to generalize this result to orientations of complete multipartite graphs. We conclude this section by considering the existence of watchman's walks in orientations of complete multipartite graphs that do not contain any source vertices.

Theorem 3.2.3. *Let D be an orientation of a complete k-partite graph for $k > 1$. If $\delta^-(D) \geq 1$, then the condensation of D is a transitive digraph, and T has a watchman's walk.*

Proof. Let D be an orientation of a complete multipartite graph such that $\delta^-(D) \geq 1$. That is, each vertex in D has at least one in-neighbour. Let $\{T_1, T_2, \ldots T_k\}$ be the maximal strongly connected components of D. Consider the condensation graph D^* of D. Let the vertices of D^* be $\{t_1, t_2, \ldots, t_k\}$, where vertex t_i in D^* corresponds to the component T_i in D. Let t be any vertex in D^*, and let P be a maximal path ending at t. Consider the start vertex u of P. Any cycle in D^* would correspond to a larger strongly connected component in D^*, contradicting the maximality of each T_i. Hence, u has no in-neighbours on P. Since P was maximal, u also does not have any in-neighbours in the subdigraph $D^* \backslash P$. Otherwise, if u' was such an in-neighbour of u, $u'P$ (where $u'P$ is the path P with the vertex u and the arc from u to the first vertex of P appended to the beginning of the path) would be a longer path, contradicting the maximality of P. So, u has no in-neighbours in D^*. This means that, in D, any vertex in the component U corresponding to the vertex $u \in D^*$ has no in-neighbours from any vertex in another strongly connected component. As each vertex must have at least one in-neighbour, U must contain more than one vertex. Hence, U contains vertices from more than one set in the partition of V. As D is an orientation of a complete multipartite graph, it follows that there is at least one arc between a vertex in U and a vertex in each set in the partition of V, and hence an arc between U and every other maximal strong component. This guarantees that there is an arc between u in D^* and each other vertex in D^*. Since u is a source in D^*, the component U in D is dominating. As U is a strongly connected component, there is a closed walk containing all of the vertices in U. As U is a dominating set, this walk is a closed dominating walk in D. Therefore, there is some watchman's walk for D. □

3.3 Semicomplete digraphs

While the underlying undirected graph of a tournament is a complete graph, tournaments are not the closest analogy to complete graphs when considering digraphs.

Instead, *complete digraphs* are much closer in structure to the undirected complete graphs. A digraph D is a *complete digraph* if, for each pair of vertices, u and v, both (u, v) and (v, u) are arcs in D. In any complete digraph on n vertices, each vertex dominates the other $n-1$ vertices in D, and is also dominated by every other vertex. Since there is an arc from each vertex to any other vertex, these digraphs are always strongly connected. The proceeding theorem follows directly from the structure and definition of these digraphs.

Theorem 3.3.1. *If D is a complete digraph of order $n \geq 1$, then $w(D) = 0$, and D has watchman multiplicity n.*

Proof. Let v be any vertex in D. Since D is a complete digraph, (v, u) is an arc for any vertex $u \neq v$ in D. So, v dominates every other vertex in D, as well as itself. Therefore, we have that the trivial walk at v is a minimum closed dominating walk of length 0. Since v was chosen arbitrarily, we have that the trivial walk at any vertex in D is a watchman's walk. Thus, $w(D) = 0$ with multiplicity n. □

We now know the length and structure of a watchman's walk in any nontrivial complete digraph. Much like complete graphs, complete digraphs are not very interesting when considering the watchman's walk problem. For the remainder of this section, we will turn our attention to another generalization of tournaments. A *semicomplete digraph* is a digraph in which there is at least one arc between any pair of vertices. Both complete digraphs and tournaments are examples of semicomplete digraphs. It follows that many of our results regarding the watchman's walk on tournaments can be generalized to semicomplete digraphs. It is straightforward to see that any semicomplete digraph is either a tournament, or contains a tournament as a spanning subdigraph. As a consequence, we get the following theorem.

Theorem 3.3.2. *If D is a semicomplete digraph, then D has a watchman's walk.*

Proof. If D is a tournament, then we know, by Theorem 2.1.8, that D has a watchman's walk. Otherwise, D is not a tournament, but contains some spanning subtournament D'. By Theorem 2.1.8, D' has some watchman's walk W. Since D' is a spanning subdigraph, W is a closed dominating walk in D, but not necessarily of minimum length. Hence, D has a watchman's walk. □

We can also say that there exists a semicomplete digraph with watchman number k for any integer $k \geq 3$. We know that this is true for tournaments, by Theorem 2.2.13, and any tournament is also a semicomplete digraph. For any tournament T with $w(T) > 0$, we know that the $w(T)$ is bounded below by $\gamma(T)$, and bounded above by $\gamma(T) + 1$. In the following theorem, we show that this is also true for any nontrivial semicomplete digraph.

Theorem 3.3.3. *Let D be a semicomplete digraph of order $n \geq 2$. If $\gamma(D)$ is the size of a minimum dominating set and $\gamma(D) > 1$, then $w(D) = \gamma(D)$ or $\gamma(D) + 1$.*

Proof. Let D be a semicomplete digraph on at least 2 vertices. Any watchman's walk W in D must be dominating, and we require at least $\gamma(D)$ vertices to dominate D. Hence, $w(D) \geq \gamma(D)$. We also know that D contains a spanning subdigraph D', where D' is a tournament. We can choose D' such that $\gamma(D') = \gamma(D)$ as follows. Consider some minimum dominating set S in D, and consider each pair of vertices u, v such that both (u, v) and (v, u) are arcs in D. If $u \in S$ and $v \notin S$, choose (u, v) to be an arc in D'. For any other pair of vertices with two arcs between them, we can choose which arc to include in D' arbitrarily. Now, D' has domination number $\gamma(D)$ since S is also a dominating set in D'. By Theorem 2.2.1, $w(D') \leq \gamma(D') + 1 = \gamma(D) + 1$. Since any watchman's walk for D' is also a closed dominating walk in D, $w(D) \leq \gamma(D) + 1$. Hence, $\gamma(D) \leq w(D) \leq \gamma(D) + 1$. □

The corollaries given for this theorem in the case of tournaments (Theorem 2.2.1) can also be generalized to semicomplete digraphs. To prove a corollary of Theorem 3.3.3, we first state two theorems.

Theorem 3.3.4. [4] Every strongly connected semicomplete digraph has a Hamilton cycle.

Theorem 3.3.5. [24] Every semicomplete digraph has a Hamilton path.

The previous theorems tell us about the hamiltonicity of semicomplete digraphs. These results will useful in proving the following corollary. The first case considered in the following proof proceeds similarly to that of the first case in Corollary 2.2.2, the analogous result for tournaments, so we will omit the details for this case.

Corollary 3.3.6. *If T is a semicomplete digraph, then no watchman's walk repeats any vertices.*

Proof. Let T be a semicomplete digraph. Consider all minimum dominating sets of D. We have two possible cases:

Case 1: There is some minimum dominating set D such that $T[D]$ is strong. By Theorem 3.3.4, $T[D]$ has some Hamilton cycle H. All vertices in a Hamilton cycle are distinct. It follows that H is also closed and dominating. By Theorem 3.3.3, $w(T) \geq \gamma(T)$. This means that, H is a closed dominating walk of length $\gamma(T)$, H is a watchman's walk for T with no repeated vertices.

Case 2: There is no minimum dominating set D such that $T[D]$, the subdigraph induced by D, contains a Hamilton cycle. In this case, there is a closed dominating walk of length $\gamma(T)$. Thus, any closed dominating walk has length at least $\gamma(T) + 1$. In particular, $w(T) \geq \gamma(T) + 1$. So, by Theorem 3.3.3, $w(T) = \gamma(T) + 1$. Hence, any watchman's walk either contains as a proper subset a minimum dominating set, or contains a minimal dominating set of size $\gamma(T) + 1$. Suppose we have a watchman's walk for T that uses the vertices of some minimum dominating set D, and at most one other vertex, u. Since $T[D]$ is a subdigraph of T, any path in $T[D]$ is also a path in T. Let W be a watchman's walk that uses some Hamilton path H of $T[D]$ and an additional vertex u (if necessary). Let $\gamma = \gamma(T)$, and $H = \{v_1, v_2, \ldots, v_\gamma\}$. If (v_γ, v_1) is an arc in T, we would have that $v_1, v_2, \ldots, v_\gamma, v_1$ is a closed dominating walk of length $\gamma(T)$. So, this is not an arc. Moreover, this is the case for any Hamilton path in $T[D]$. Thus, we need to visit another vertex u to ensure that the walk is closed. This means that u must be an out neighbour of v_γ, and an in-neighbour of v_1. Since there was no Hamilton cycle in $T[D]$, we know that $T[D]$ is not strongly connected. It follows that there is no path from v_γ to v_1 in $T[D]$. Thus, there is no vertex in D that is both an out-neighbour of v_γ, and an in-neighbour of v_1. If there was no vertex in T that was both an out-neighbour of v_γ and an in-neighbour of v_1, we would have that any vertex dominated by v_γ is also dominated by v_1. Since v_γ is dominated by v_1 also, it follows that $\{v_1, v_2, \ldots, v_{\gamma-1}\}$ is a dominating set in T of size $\gamma - 1$. This is a contradiction, so there exists a vertex u such that it is both an out-neighbour of v_γ, and an in-neighbour of v_1. As $u \notin D$ and each vertex in D appears in W exactly once, W does not repeat any vertices.

Now consider a watchman's walk for T that uses the vertices of some minimal dominating set D of size $\gamma(T)+1$. The subdigraph $T[D]$ must be a strong tournament. Otherwise, we would not be able to construct a closed walk of length $\gamma(T)$. Since D is minimal, the elements of D are distinct. Thus, for any strongly connected tournament T, there exists a watchman's walk of T that does not repeat any vertices. □

3.4 De Bruijn digraphs

In 1946, de Bruijn studied the problem of finding a shortest possible binary sequence that contains every binary string of length k. In [6], he solved this problem and introduced both de Bruijn sequences and de Bruijn digraphs. A *de Bruijn sequence of order* k is a binary sequence of length 2^k such that:

- the last bit is said to be adjacent to the first; and
- every binary k-tuple occurs exactly once in the sequence.

For example, a de Bruijn sequence of order 2 is 1001. As shown in Figure 3.1, every binary 2-tuple occurs in this sequence.

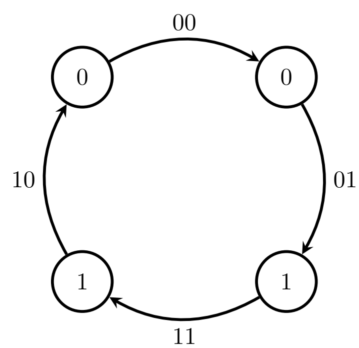


Figure 3.1: A de Bruijn sequence of order 2

A k-bit substring $b_1b_2\ldots b_k$ is obtained from another substring $a_1a_2\ldots a_k$ such that $b_i = a_{i+1}$ for $1 \le i < k$ by a left shift operation. There are two possible left shift operations:

- a *cycle shift* $a_1a_2\ldots a_k \to b_1b_2\ldots b_k$, when $b_k = a_1$. Ex: $100 \to 001$.
- or a *de Bruijn shift*, $a_1a_2\ldots a_k \to b_1b_2\ldots b_k$, when $b_k \neq a_1$. Ex: $100 \to 000$.

In a de Bruijn sequence S of order k, the *k-tour* is the sequence of substrings of length k, in order of their occurrence in s, starting with the initial k-string in S. Since S is considered to be cyclic, there are 2^k substrings in the tour.

A de Bruijn graph of order k is a directed graph, denoted $G(k)$, whose 2^k vertices are labelled by each possible binary string of length k. There is an arc from the vertex labelled by string a to the vertex labelled by string b if and only if b can be obtained from a using one of the left shift operations. The de Bruijn graph of order 3 is shown in Figure 3.2. Given a de Bruijn sequence B of order k and its corresponding de Bruijn graph, we say that the *subdigraph induced by a subsequence* D is the subdigraph induced by the k-tour of D.

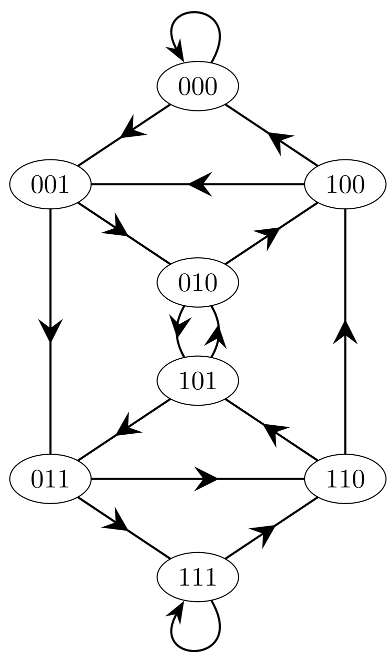


Figure 3.2: A de Bruijn graph of order 3

From Figure 3.2 we can see that $G(3)$ has a watchman's walk, namely the cycle given by $100, 001, 011, 110, 100$. In the following theorem, we show that this is also true of $G(k)$ for any value of k. In Theorem 3.4.2, we give the watchman number of any de Bruijn graph.

Theorem 3.4.1. *The de Bruijn digraph $G(k)$ has a watchman's walk for any $k \geq 1$.*

Proof. Let $D = G(k)$ for some $k \geq 1$. Since D is a de Bruijn digraph, D corresponds to some de Bruijn sequence S of order k. Consider the k-tour of S. This sequence of strings contains all 2^k strings that label the vertices of D. Also, there is an arc from each string a in this sequence to the following string b, as the initial $k-1$ substring in b is the final $k-1$ substring in a. Thus, we have a directed walk containing all vertices of D. Since S and its k-tour are considered cyclically, this k-tour corresponds to a Hamilton cycle in D. This cycle is a closed dominating walk in D, and hence D has some minimum closed dominating walk. □

Theorem 3.4.2. *If $G = G(k)$ is a de Bruijn graph of order $k > 1$ then $w(G) = 2^{k-1}$.*

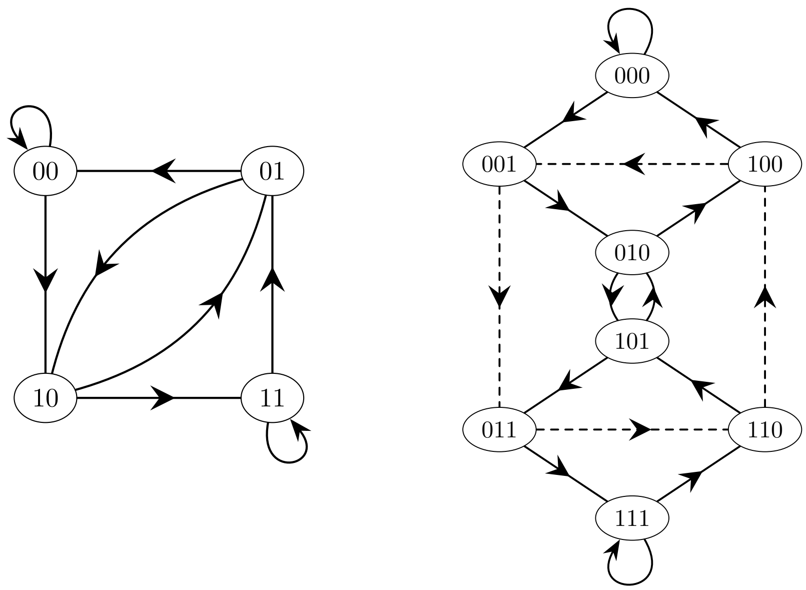
Proof. Consider the de Bruijn graph of order $k-1$. This graph has 2^{k-1} vertices, and as in the proof of Theorem 3.4.1, it has a Hamilton cycle. Let H be some Hamilton cycle in $G(k-1)$. Construct a de Bruijn sequence S of order $k-1$ by appending the last bit of each $k-1$ string in $G(k-1)$, in the order that they occur in H. Note that each possible $k-1$ binary string occurs exactly once in that sequence, and this is a de Bruijn sequence of order $k-1$.

Now consider $G(k)$. If we again consider the sequence S, we can generate a set D of 2^{k-1} substrings of length k. For each string $v \in D$, we obtain a $(k-1)$-string by removing the first bit of a string $v \in D$. As S was a de Bruijn sequence of order $k-1$, these $(k-1)$-strings are all distinct. By appending a 0 or 1 to the end of each $(k-1)$-string, we obtain the two neighbours of v in G. As the $(k-1)$-strings are all distinct, the neighbourhoods of any two elements in D are disjoint. Thus, any vertex in G is either in D or is an out-neighbour of an element of D.

We now know that D is a dominating set in G. Also, if we consider the k-tour of S, each k-string in the tour can be obtained by a cycle shift or de Bruijn shift from the previous string. So, each string in the tour is an out-neighbour in G of the preceding string. As the k-tour is considered cyclically, and the strings in this tour are all the possible k-bit subsequence of D, this tour corresponds to a dominating cycle in G. Hence, D induces a dominating cycle in G.

Let v be a vertex in G such that there is no arc from v to itself. Consider forming a dominating cycle W that starts at v. Each vertex has at most two out-neighbours, and one of these out-neighbours must also be in W. So, each vertex dominates at most 1 vertex outside of W. So, if W is dominating, W must have length at least $\frac{2^k}{2} = 2^{k-1}$. Now, D induces a dominating cycle of length 2^{k-1}, so D is a minimum dominating cycle. That is, the subdigraph of G induced by D is a minimum closed dominating walk in G. So, since the length of D is 2^{k-1} and D is a watchman's walk for G, $w(G) = 2^{k-1}$. □

Example 3.4.3. The de Bruijn sequence of order 2, 1001, has 3-tour {100, 001, 011, 110}. This induces a watchman's walk in $G(3)$, as illustrated in Figure 3.3b.



(a) A de Bruijn graph of order 2 for the sequence 0110

(b) The walk defined by the de Bruijn sequence of order 2

Figure 3.3: Watchman's walk in a de Bruijn graph of order 3

We can also consider de Bruijn sequences of order k on alphabets of size greater than two. In these sequences, we still must have that every possible k sequence occurs exactly once as a subsequence. As a result, a de Bruijn sequence of order k on an alphabet A of size a has length a^k. We denote the corresponding de Bruijn graph by $G(a, k)$. In the following theorem, we generalize our method of obtaining a watchman's walk in a binary de Bruijn sequence to de Bruijn sequences of order k on an alphabet of size a.

Theorem 3.4.4. *If G is the de Bruijn graph of a de Bruijn sequence of order k on an alphabet of size a, then $w(G) = a^{k-1}$.*

Proof. Consider $G(a, k-1)$. This graph has a^{k-1} vertices. The $(k-1)$ tour of the corresponding de Bruijn sequence S corresponds to a Hamilton cycle of the digraph, as each substring is included exactly once in the tour, and there is an arc from the vertex corresponding to any substring to the vertex corresponding to the following substring in the tour. Let H be a Hamilton cycle in $G(a, k-1)$. We will show that the Hamilton cycle H generated by S corresponds to a watchman's walk of $G(a, k)$.

Consider $G = G(a, k)$. If we again consider the $(k-1)$-tour of the sequence S of order $k-1$, we can generate a set D of a^{k-1} strings of length k. For each string $v \in D$, we obtain a $(k-1)$-string v' by removing the first bit of a string $v \in D$. As D was a de Bruijn sequence of order $k-1$, these $(k-1)$-strings are all distinct. By appending an element of the alphabet A to the end of v' for each $v \in D$, we obtain the a neighbours of v in G. Since the $(k-1)$ strings were distinct, the neighbourhoods in G of any two elements of D are disjoint. Thus, we get that the union of the neighbourhoods of the elements of D is all of the a^k vertices in G. Thus, any vertex in G is either in D or is dominated by an element of D. This means that D is a dominating set in G.

If we again consider the k-tour of S, each k-substring in this tour can be obtained by a cycle shift or de Bruijn shift from the previous string. As the k-tour is considered cyclically, this tour corresponds to a closed dominating walk in G. D is also a dominating set of G, so D induces a closed dominating walk in G.

Finally, D must induce a minimum dominating cycle. Let v be a vertex in G. Consider forming a dominating cycle W that starts at v. Each vertex has at most a out-neighbours, and one of these out-neighbours must also be in W. So, each vertex dominates at most $a-1$ vertices outside of W. So, if W is dominating, W must have length at least $\frac{a^k}{a} = a^{k-1}$. Now, D induces a dominating cycle of length a^{k-1}, so D induces a minimum dominating cycle. Since the length of D is a^{k-1} and D is a watchman's walk for G, $w(G) = a^{k-1}$. □

If S is a de Bruijn sequence of order k on an alphabet A, we call a sequence D on the same alphabet A a *generating sequence* if D has length at least k. In this case, k-sequences in D may be repeated. Call the graph induced by the set of k-sequences in D as well as the k-sequences obtainable from the k-sequences in D via a left shift operation the *de Bruijn subdigraph generated by* D. If D does not contain every possible k-string, the subdigraph generated by D may be a proper subdigraph of the de Bruijn graph corresponding to S. If we have a generating sequence D that does not contain all possible k-strings, we would like to know when the k-tour of D induces a watchman's walk for the de Bruijn subdigraph generated by D. We give two families of sequences that never induce a watchman's walk in the de Bruijn subdigraph.

Lemma 3.4.5. *If D is a generating sequence of order k on any alphabet of size at least 2, then the walk induced by D is never a watchman's walk in the de Bruijn subdigraph if there are k consecutive occurrences of the same bit in D.*

Proof. Let $D = a_1, a_2, \ldots, , a_i, a_{i+1}, \ldots, a_{i+k}, a_{i+k+1}, \ldots, a_n$ where $a_{i+1} = a_{i+2} = \ldots = a_{i+k}$. In the subdigraph G' generated by D, let $v_1 = a_i, a_{i+1}, \ldots, a_{i+k-1}$, $v_2 = a_{i+1}, \ldots, a_{i+k}$, and $v_3 = a_{i+2}, \ldots, a_{i+k+2}$. Since $a_{i+1} = a_{i+2} = \ldots = a_{i+k}$, we know that (v_1, v_2) is an arc in G', and $N[v_2] \subseteq N(v_1)$. Hence, we would not need to visit v_2 in a watchman's walk for G'. Thus, D does not induce a watchman's walk for the subdigraph generated by D. □

Lemma 3.4.6. *If D is a de Bruijn generating sequence, where D is the concatenation of two identical sequences of length at least k, then the cycle induced by D is not a watchman's walk in the de Bruijn subdigraph.*

In this case, we would only need to traverse at most half of the k-tour of the generating sequence to get a watchman's walk. Thus, the entire k-tour would be superfluous.

We also know that in some cases, our sequence will always induce a watchman's walk. We give such a sequence in the following lemma.

Lemma 3.4.7. *If D is some de Bruijn generating sequence of order k such that there are no repeated $(k-1)$-tuples in D, then the walk induced by the sequence is a watchman's walk in the de Bruijn subdigraph generated by D.*

Proof. In the subdigraph generated by D, the neighbourhood of each vertex is determined by the final $(k-1)$ bits in the k-substring that labels the vertex. If every $(k-1)$-tuple in D is unique then each vertex in the subdigraph generated by D will have only one out-neighbour in that subdigraph, and their out-neighbourhoods in $G_D(|A|, k)$ will be unique. Thus, if D is a sequence of length n, then $G_D(|A|, k)$ will have $k \times n$ vertices. The vertices of the subdigraph generated by D induce a closed walk in $G_D(|A|, k)$. This walk is clearly dominating, and has length n. In $G_D(|A|, k)$, each vertex has at most k out-neighbours, so any closed dominating walk must have length at least $\frac{k \times n}{k} = n$. Hence, the closed dominating walk induced by the vertices of the subdigraph generated by D is a watchman's walk in $G_D(|A|, k)$. □

Chapter 4

Conclusion

In this thesis, we considered the Watchman's Walk Problem on directed graphs. We first explored the problem on tournaments. In Chapter 2, it was proved that for any tournament T, $\gamma(T) \leq w(T) \leq \gamma(T) + 1$. By generalizing results in [8], we noted that for any integer $k \geq 3$, there exists a tournament with watchman number k. Chapter 2 also presents computational results for all tournaments with at most 10 vertices. The number of tournaments with a specified order, watchman number and domination number are given. We also considered families of tournaments, including simple tournaments and transitive tournaments.

In the previous chapter, we explored other digraphs that generally have watchman's walks. We considered orientations of complete multipartite graphs and de Bruijn digraphs. It was also shown that all de Bruijn digraphs of order $k \geq 1$ have a watchman's walk. Moreover, if k is at least 2, a watchman's walk in the de Bruijn digraph of order k can be constructed using a Hamilton cycle in the de Bruijn graph on the same alphabet of order $k - 1$.

There are many further directions that this research could take in the future. In particular, it would be interesting to classify the tournaments based on whether they have watchman number $\gamma(T)$ or watchman number $\gamma(T) + 1$. We know that for tournaments $\gamma(T) \leq \gamma_t(T) \leq w(T) = \gamma_{cyc}(T)$. It would be reasonable to expand this chain for other parameters, or generalize the domination chain given by Reid et al. in [20]. Further, it would be interesting to determine when a de Bruijn generating sequence induces a watchman's walk in the subdigraph it generates.

Additionally, variations of the watchman's walk problem on directed graphs are largely unstudied. Both the fixed time and multiple guard variations of the problem could be studied on directed graphs. For families of directed graphs, the minimum number of guards needed to achieve a given maximum unseen time is not currently known. The most efficient route for a given number of guards is also not known in directed graphs. The study of variations of the watchman's walk problem on directed graphs would not only help to fill a gap in the current research, but by looking at variations for digraphs, we could gain insight about the watchman's walk problem for digraphs where no traditional watchman's walk exists.

Bibliography

[1] S. Arumugam, K. Jacob, and L. Volkmann. Total and connected domination in digraphs. *The Australasian Journal of Combinatorics*, 39, 2007.

[2] L. W. Beineke and R. J. Wilson. *Selected topics in graph theory.* Academic Press, London; New York, 1978.

[3] C. Berge. The theory of graphs and its applications. *The bulletin of mathematical biophysics*, 1962.

[4] P. Camion. Chemins et circuits hamiltoniens des graphes complets. *C. R. Acad. Sci. Paris*, 249:2151–2152, 1959.

[5] E. J. Cockayne, S. T. Hedetniemi, and D. J. Miller. Properties of hereditary hypergraphs and middle graphs. *Canadian Mathematical Bulletin*, 21(4):461468, 1978.

[6] N. G. de Bruijn. A combinatorial problem. *Nederl. Akad. Wetensch., Proc.*, 49, 1946.

[7] C. F. de Jaenisch. *Applications de l'analyse mathematique au jeu des echecs*, volume 3. Saint-Pétersbourg, 1862.

[8] J. Duncan and M. Jacobson. On tournaments with domination number exactly k. *The Journal of Combinatorial Mathematics and Combinatorial Computing*, 31, 1999.

[9] D. Dyer and J. Howell. The multiplicity of watchman's walks. *Congr. Numer.*, 226:301–317, 2016.

[10] D. Dyer and J. Howell. The watchman's walk of steiner triple system block intersection graphs. *Australasian J. Combinatorics*, 68:23–34, 2017.

[11] P. Erdös. On a problem in graph theory. *Math. Gaz.*, 47:220–223, 1963.

[12] A. Hagberg, P. Swart, and D. Chult. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11–15, 2008.

[13] F. Harary. *Structural models: an introduction to the theory of directed graphs*. Wiley, New York, 1965.

[14] B. L. Hartnell, D. F. Rall, and C. A. Whitehead. The watchmans walk problem: An introduction. *Congressus Numerantium*, 130:149–155, 1998.

[15] T. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. Marcel Dekker, New York, 1998.

[16] B. McKay. Digraphs. Data retrieved from Australian National University.

[17] O. Ore. *Theory of graphs*. American Mathematical Society Colloquium Publications, Vol. XXXVIII. American Mathematical Society, 1967.

[18] L. Rédei. Ein kombinatorischer satz. *Acta Litteraria Szeged*, 7:39–43, 1934.

[19] K. Reid. Tournaments. In *Handbook of graph theory*, Discrete mathematics and its applications, chapter 3, page 201. Chapman and Hall, second edition, 2013.

[20] K. B. Reid, A. A. McRae, S. M. Hedetniemi, and S. T. Hedetniemi. Domination and irredundance in tournaments. *Australas. J. Combin.*, 29:157–172, 2004.

[21] D. Soroker. Fast parallel algorithms for finding hamiltonian paths and cycles in a tournament. *Journal of Algorithms*, 9(2):276 – 286, 1988.

[22] P. Tetali. A characterization of unique tournaments. *Journal of Combinatorial Theory, Series B*, 72(1):157 – 159, 1998.

[23] G. Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[24] E. Witt. Ein kombinatorischer satz der elementargeometrie. *Mathematische Nachrichten*, 6:261–262, 1952.

Appendix A

The following program was written to algorithmically determine the number of tournaments with a given watchman's walk number, domination number, and multiplicity of watchman's walks. The program uses the upper triangles of adjacency matrices for every tournament of order up to 10 given in [16] to compute an adjacency list of each tournament, followed by the condensation of the tournament, watchman's walks, and domination number. This program uses the graph class functions given in the Networkx package in [12].

```
import os

T=10 #up to order T
array=[[] for j in range (T)]

# Reads tournaments (as upper triangle of the adjacency
    #  matrix) from tournaments.txt and writes all
    #  tournaments of order n to filename{n}.txt
def tournamentss():
    desktop=os.path.join(os.path.expanduser("~"),"Documents")
    filepath=os.path.join(desktop, "tournaments.txt")
    with open(filepath,'r') as rf:
        record = 1
        output = open('filename{}.txt'.format(record),'w')
        for line in rf:
            if line == "\n":
                record += 1
                output.close()
                output=open('filename{}.txt'.format(record)
```

```
                    ,'w')
        else:
            output.write(line)
tournament=1
i=0
while tournament < T:
    while tournament < 8:
        with open("filename{}.txt".format(tournament),
        "rb") as f:
            for line in f:
                line = line.replace('\n' , '')
                line=str(line)
                array[i].append(line)
            tournament +=1
            i+=1
    if T > 8:
        if tournament == 8:
            with open("tourn9.txt","rb") as f:
                for line in f:
                    line = line.replace('\n' ,'')
                    line=str(line)
                    array[i].append(line)
                tournament +=1
                i+=1
    if T > 9:
        if tournament == 9:
            with open("tourn10.txt","rb") as f:
                for line in f:
                    line = line.replace('\n' ,'')
                    line=str(line)
                    array[i].append(line)
                tournament +=1
                i+=1
return array
```

```
x=[0]
y=[1]

# fills in the upper triangle of the
    # adjacency matrix for each tournament.
    # array: array[i] contains the adjacency info
    # (as a list of the upper triangle.
    # entries for all tournaments of order i+1
    # v: order of the tournaments
def adjmatrix(array,v):
    tournamentss()
    num=len(array[v-2])
    arr=[None]
    mats=[None]
    mats=[]
    mats=list(array[v-2])
    tri=v*(v-1)*0.5 # length of the upper triangle list
    matrix=[[[None]*v for l in range (v)]
            for g in range (num)]
    for b in range (num):
        k=0
        while k < tri:
            for i in range (v-1):
                n=v-1-i
                j=0
                while j <n:
                    if mats[b][k] == '0':
                        matrix[b][i][i+j+1]=0
                    if mats[b][k] == '1':
                        matrix[b][i][i+j+1]=1
                    j+=1
                    k+=1
    return matrix
```

```
# Fills in the diagonal and lower triangle
    #of each adjacency matrix
def finishadjmat (array,v):
    matrix=[]
    matrix=adjmatrix(array,v)
    n=len(matrix)
    m=len(matrix[0])
    for i in range(n):
        for j in range (m):
            matrix[i][j][j]=0
            for k in range (j+1,m):
                if matrix[i][j][k]==0:
                    matrix[i][k][j]=1
            for k in range (j+1,m):
                if matrix[i][j][k]==1:
                    matrix[i][k][j]=0
    return matrix

# converts an adjacency matrix to an adjacency list
def matToList (matrix):
    r=len(matrix[0])
    neighbours=[[] for i in range (r)]
    for i in range (len(matrix[0])):
        for j in range (len(matrix[0])):
            if matrix[i][j]==1:
                neighbours[i].append(j)
    return neighbours

from collections import defaultdict

class Graph: # All class Graph functions are
           # found in Networkx or modified
           # from functions found in Networkx
```

```
def __init__(self,vertices):
    self.V= vertices #No. of vertices
    self.graph = defaultdict(list) # default
        # dictionary to store graph
def isAdj(self, v, pos, cycle):
# Check if current vertex and last vertex
# in path are adjacent
    if self.graph [ cycle[pos-1] ][v] == 0:
        return False
        # Check if current vertex is not in path
    for vertex in cycle:
        if vertex == v:
            return False
    return True
# function to add an edge to graph
def addEdge(self,u,v):
    self.graph[u].append(v)

# A function used by DFS
def DFSUtil(self,v,visited,j, comps):
    # Mark the current node as visited and print it
    visited[v]= True
    if v not in comps[j]:
        comps[j].append(v),
    for i in self.graph[v]:
        if visited[i]==False:
            self.DFSUtil(i,visited,j, comps)

def fillOrder(self,v,visited, stack):
    # Mark the current node as visited
    visited[v]= True
    for i in self.graph[v]:
        if visited[i]==False:
            self.fillOrder(i, visited, stack)
```

```
        stack = stack.append(v)
    def getTranspose(self):
        g = Graph(self.V)
        for i in self.graph:
            for j in self.graph[i]:
                g.addEdge(j,i)
        return g
    # prints the strongly connected
        # components of a digraph
    def printSCCs(self):
        stack = []
        scc=[]
        comps=[]
        del scc [:]
        # Mark all the vertices as not visited
            #(For first iteration of D–F S)
        visited =[False]*(self.V)
        # Fill vertices in stack according to
            # their finishing times
        for i in range(self.V):
            if visited[i]==False:
                self.fillOrder(i, visited, stack)

        # Create a reversed graph
        gr = self.getTranspose()

        # Mark all the vertices as not visited
            # (For 2nd D–F S)
        visited =[False]*(self.V)

        # Process vertices in order defined by Stack
        j=0
        while stack:
            i = stack.pop()
```

```
            if visited[i]==False:
                comps.append([])
                gr.DFSUtil(i,visited,j, comps)
                j=j+1
        scc=filter(None, comps)
        return scc
    def hamCycleProb(self,cycle,pos):
        if pos == self.V:
                # Must be an arc from the end vertex
                # to the first vertex of the path
                if self.graph[cycle[pos-1]][cycle[0]]==1:
                    return True
                else:
                    return False

            # Try other vertices to find a vertex we can
            # use next in the cycle, besides 0 (since it
            # is fixed as the start vertex)
        for v in range(1, self.V):
            if self.isAdj(v,pos,cycle)==True:
                cycle[pos] = v
                if self.hamCycleProb(cycle,pos+1)==True:
                    return True
                    #Remove current vertex
                        #if we get stuck
            cycle[pos] = -1
        return False
    # checks if digraph is strongly connected
    def isItStrong(self):
        cycle = [-1] * self.V
        cycle[0] = 0
        if self.hamCycleProb(cycle,1) == False:
            return False
        return True
```

```
# converts adjacency list to a graph
def makeGraph(digraph):
    n=len(digraph)
    g=None
    g=Graph(n)
    for i in range (n):
        for j in digraph[i]:
            g.addEdge(i,j)
    return g

# Finds the dominating strong component given adjacency
# list and number of vertices
def ordering(g,v):
    dig=[]
    dig=Graph(v)
    dig=makeGraph(g)
    comps=[]
    comps=dig.printSCCs()
    n=len(comps)
    if len(comps)==1:
        return comps
    else:
        for i in range (n):
            if comps[i]==None:
                i+=1
            if len(comps[i])==1:
                ver=comps[i][0]
                if g[ver]==None:
                    i+=1
                if g[ver]!=None and len(g[ver])==v-1:
                    return comps[i]
            else:
                m=0
```

```
                for j in range (n):
                    if j != i:
                        ver=comps[j][0]
                        vert=comps[i][0]
                        if ver in g[vert]:
                            m+=1
                            if m==(n-1):
                                return comps[i]


def main(array):
    for v in range (2,T+1):
        opts=[]
        number=[1,2,4,12,56,456,6880,191536, 9733056]
        matr=finishadjmat (array,v)
        output = open('outputfile{}.txt'.format(v),'w')
        l=number[v-2]
        i=0
        while i < l:
            g1=Graph(v)
            g1.graph=matr[i]
            adlist=matToList(g1.graph)
            line=str(getww(adlist, v, matr[i]))
            c=0
            z=len(opts)
            for q in range(z):
                if line in opts[q]:
                    opts[q][1]+=1
                    c+=1
            if c==0:
                opts.append([line,1])
            i+=1
        for r in range(len(opts)):
            output.write(str(opts[r]))
```

```
                output.write('\n')

# Finds the length of a ww of a tournament given the
# adjacency list, adjacency matrix, and number of vertices.
def getww(adlist,v, matr):
    g1=Graph(v)
    g1.graph=matr
    tad=[]
    if g1.isItStrong()==True:
        tournament=g1.graph
    if g1.isItStrong()==False:
        agraph=None
        agraph=Graph(v)
        agraph=makeGraph(adlist)
        tournament=agraph.printSCCs()
        tournament=ordering(adlist, v)
        tournament=makesubgraph(adlist, tournament)
    n=len(tournament)
    if n>1:
        tad=matToList(tournament)
    if n==1:
        line='w=0, \gamma=1, m=1'
        return line
    m=0
    ww=[]
    c=0
    for i in range (n):
        adi=[]
        for d in range (n):
            if tournament[i][d]==1:
                adi.append(d)
        for j in range (i+1, n):
            adj=[]
            for d in range (n):
```

```
                if tournament[j][d]==1:
                    adj.append(d)

            domd=[]
            domd.extend(adi)
            domd.extend(adj)
            domd.append(i)
            domd.append(j)
            domd=list(set(domd))
            if len(domd)==(n):
                c+=1 # found a dominating set of size 2
                break

        for j in adi: # check for dominating triangles
            adj=[]
            for d in range (n):
                if tournament[j][d]==1:
                    adj.append(d)
            for k in adj:
                if k not in adi:
                    adk=[]
                    for d in range (n):
                        if tournament[k][d]==1:
                            adk.append(d)
                    domd=[]
                    domd.extend(adi)
                    domd.extend(adj)
                    domd.extend(adk)
                    domd.append(i)
                    domd.append(j)
                    domd.append(k)
                    domd=list(set(domd))
                    if len(domd)==n:
                        ww.append([i,j,k])
```

```
                    m+=1
    if m!=0:
        if c !=0:
            line = ("& 3 & 2 & {}".format(m))
            return line
        if c == 0:
            line = ("& 3 & 2 & {}".format(m))
            return line

    if m==0:
        wwsets=[]
        for item in ww:
            [i,j,k]=item
            if j in tad[i]:
                if k in tad[j]:
                    for vert in tad[k]:
                        if i in tad[vert]:
                            wwset=[i,j,k,vert]
                            wwset.sort()
                            if wwset not in wwsets:
                                wwsets.append(wwset)
                if k not in tad[j] and k in tad[i]:
                    for vert in tad[j]:
                        if i in tad[vert]:
                            wwset=[i,j,k,vert]
                            wwset.sort()
                            if wwset not in wwsets:
                                wwsets.append(wwset)
                if k not in tad[j] and k not in tad[i]:
                    for vert in tad[j]:
                        if k in tad[vert]:
                            wwset=[i,j,k,vert]
                            wwset.sort()
                            if wwset not in wwsets:
```

```
                                wwsets.append(wwset)
                if j not in tad[i] and k in tad[i]:
                    for vert in tad[k]:
                            if j in tad[vert]:
                                wwset=[i,j,k,vert]
                                wwset.sort()
                                if wwset not in wwsets:
                                    wwsets.append(wwset)
                if j not in tad[i] and k not in tad[i]:
                    if k in tad[j]:
                        for vert in tad[i]:
                                if j in tad[vert]:
                                    wwset=[i,j,k,vert]
                                    wwset.sort()
                                    if wwset not in wwsets:
                                        wwsets.append(wwset)
                    if k not in tad[j]:
                        for vert in tad[i]:
                                if k in tad[vert]:
                                    wwset=[i,j,k,vert]
                                    wwset.sort()
                                    if wwset not in wwsets:
                                        wwsets.append(wwset)
        mult=len(wwsets)
        line = ('& 3 & 4 & {}'.format(mult))
        return line


# Finds the adjacency matrix of a subdigraph of the
#   tournament defined by adjacency list 'adlist'
#   induced by subset 'subs'.
def makesubgraph (adlist, subs):
    n=len(subs)
    subs.sort()
```

```
    mat=[[None]*n for i in range (n)]
    for i in range (n):
        for j in range (n):
            mat[i][i]=0
            if j!= i:
                p=subs[i]
                if subs[j] in adlist[p]:
                    mat[i][j]=1
                if subs[j] not in adlist[p]:
                    mat[i][j]=0
    return mat

adlist=[]
main(array)
```

Appendix B

The following tables specify the number of tournaments with a given watchmans walk number, domination number, and multiplicity of watchmans walks for each order up to 10. The values of $w(T)$,$\gamma(T)$, and watchman walk multiplicity were found using the program in Appendix A.

Table 1: Computational results for tournaments of order 2

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	1

Table 2: Computational results for tournaments of order 3

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	1
3	2	1	1

Table 3: Computational results for tournaments of order 4

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	2
3	2	1	1
3	2	2	1

Table 4: Computational results for tournaments of order 5

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	4
3	2	1	1
3	2	2	2
3	2	3	3
3	2	4	1
3	2	5	1

Table 5: Computational results for tournaments of order 6

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	12
3	2	1	2
3	2	2	4
3	2	3	10
3	2	4	12
3	2	5	6
3	2	6	8
3	2	8	2

Table 6: Computational results for tournaments of order 7

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	56
3	2	1	4
3	2	2	12
3	2	3	38
3	2	4	74
3	2	5	69
3	2	6	63
3	2	8	40
3	2	7	53
3	2	9	26

Table 6 – continued

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
3	2	11	4
3	2	10	11
3	2	12	3
3	2	13	1
3	2	14	1
3	3	7	1

Table 7: Computational results for tournaments of order 8

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	456
3	2	1	12
3	2	2	48
3	2	3	208
3	2	4	544
3	2	5	770
3	2	6	820
3	2	7	788
3	2	8	892
3	2	9	704
3	2	10	657
3	2	11	387
3	2	12	294
3	2	13	114
3	2	14	99
3	2	15	36
3	2	16	27
3	2	17	8
3	2	18	9
3	2	20	2
3	3	7	2
3	3	8	1
3	3	9	1

Table 7 – continued

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
3	3	10	1

Table 8: Computational results for tournaments of order 9

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	6880
3	2	1	56
3	2	2	296
3	2	3	1648
3	2	4	5684
3	2	5	11125
3	2	6	14911
3	2	8	18889
3	2	7	15929
3	2	9	20493
3	2	10	21489
3	2	11	19734
3	2	12	17157
3	2	13	12413
3	2	14	8912
3	2	15	6108
3	2	16	3884
3	2	17	2319
3	2	18	1519
3	2	19	801
3	2	20	461
3	2	21	286
3	2	22	147
3	2	23	72
3	2	24	60
3	2	25	19
3	2	26	4
3	2	27	8

Table 8 – continued

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
3	2	28	4
3	2	29	1
3	2	30	1
3	3	7	6
3	3	8	19
3	3	9	48
3	3	10	65
3	3	11	46
3	3	12	22
3	3	13	14
3	3	14	3
3	3	15	1
3	3	18	1
3	3	27	1

Table 9: Computational results for tournaments of order 10

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
0	1	1	191536
3	2	1	456
3	2	2	3040
3	2	3	20808
3	2	4	90528
3	2	5	232866
3	2	6	395927
3	2	7	493369
3	2	8	590172
3	2	9	714023
3	2	10	874685
3	2	11	952415
3	2	12	1013385

Table 9 – continued

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
3	2	13	933658
3	2	14	823741
3	2	15	665467
3	2	16	527268
3	2	17	377567
3	2	18	277459
3	2	19	184796
3	2	20	126674
3	2	21	78776
3	2	22	52721
3	2	23	31016
3	2	24	21213
3	2	25	11643
3	2	26	7727
3	2	27	4137
3	2	28	2622
3	2	29	1437
3	2	30	1015
3	2	31	400
3	2	32	367
3	2	33	112
3	2	34	121
3	2	35	24
3	2	36	47
3	2	38	18
3	2	40	4
3	3	5	1
3	3	7	45
3	3	8	360
3	3	9	1603
3	3	10	3933
3	3	11	5672

Table 9 – continued

$w(T)$	$\gamma(T)$	Watchman multiplicity	Number of tournaments
3	3	12	5752
3	3	13	4869
3	3	14	3298
3	3	15	2015
3	3	16	1176
3	3	17	585
3	3	18	255
3	3	19	127
3	3	20	58
3	3	21	22
3	3	22	19
3	3	23	7
3	3	24	2
3	3	25	3
3	3	26	3
3	3	27	5
3	3	29	1
3	3	30	2
3	3	31	1
3	3	36	2

Appendix C

The following program algorithmically checks for strongly connected dominating sets of size six in the graph in Figure 1.7. The program uses a depth-first search function from [12] to find all strongly connected subsets of size 6, and then determines if each set is a dominating set in the graph. This program uses the Itertools package from [23], as well as the Networkx package from [12].

```
import networkx as nx
import itertools
matrix=[[0,1,0,0,0,0,0,1,0,0,0,0,0,0,0],
[0,0,0,0,0,0,1,0,1,0,0,0,0,0,0],
[0,0,0,1,0,0,0,0,0,1,0,0,0,0,0],
[0,0,0,0,0,0,1,0,0,0,1,0,0,0,0],
[0,0,0,0,0,1,0,0,0,0,0,1,0,0,0],
[0,0,0,0,0,0,1,0,0,0,0,0,1,0,0],
[1,0,1,0,1,0,0,0,0,0,0,0,0,1,1],
[0,1,0,0,0,0,0,0,1,0,0,0,0,0,0],
[0,0,1,0,0,0,0,0,0,1,0,0,0,0,0],
[0,0,0,1,0,0,0,0,0,0,1,0,0,0,0],
[0,0,0,0,1,0,0,0,0,0,0,1,0,0,0],
[0,0,0,0,0,1,0,0,0,0,0,0,1,0,0],
[0,0,0,0,0,0,1,0,0,0,0,0,0,1,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,0,0,0,0,0,0,1,0,0,0,0,0,0,0]]
def makeGraph(matrix):
  G=nx.DiGraph()
   vlist=[]
   alist=[]
```

```
  for i in range(1,len(matrix)+1):
      vlist.append(i)
  G.add_nodes_from(vlist)
  for i in range(0,len(matrix)):
      for j in range (0,len(matrix)):
          if matrix[i][j]==1:
              G.add_edge(i+1,j+1)
  return G
def isstronglyconnected(G):
  firstsort = {}
  secondsort = {}
  scc_found = {}
  scc_queue = []
  i = 0
  for root in G:
      if root not in scc_found:
          queue = [root]
          while queue:
              v = queue[-1]
              if v not in firstsort:
                  i = i + 1
                  firstsort[v] = i
              done = 1
              v_nbrs = G[v]
              for w in v_nbrs:
                  if w not in firstsort:
                      queue.append(w)
                      done = 0
                      break
              if done == 1:
                  secondsort[v] = firstsort[v]
                  for w in v_nbrs:
                      if w not in scc_found:
                          if firstsort[w] > firstsort[v]:
```

```
                                    secondsort[v]=
                                          min([secondsort[v],
                                          secondsort[w]])
                                else:
                                    secondsort[v] =
                                          min([secondsort[v],
                                          firstsort[w]])
                        queue.pop()
                        if secondsort[v] == firstsort[v]:
                            scc_found[v] = True
                            scc = {v}
                            while scc_queue and
                                      firstsort[scc_queue[-1]]
                                      > firstsort[v]:
                                k = scc_queue.pop()
                                scc_found[k] = True
                                scc.add(k)
                            yield scc
                        else:
                            scc_queue.append(v)
def main(matrix):
  scsets=[]
  scdom=[]
  G=makeGraph(matrix)
  choices={1,2,3,4,5,6,8,9,10,11,12,13,15}
  subsets=findsubsets(choices, 5)
  subset=[[] for j in range (0,len(subsets))]
  for i in range (0, len(subsets)):
    subset[i]=list(subsets[i])
    subset[i].append(7)
    H = G.subgraph(subset[i])
    if isstronglyconnected(H) == True:
      scsets.append(subset[i])
  for i in range (0,len(scsets)):
```

```
    item=scsets[i]
    domintd=[0 for k in range (15)]
    for v in range (15):
      for u in item:
        domintd[v]=domintd[v]+matrix[u-1][v]
    if 0 not in domintd:
      scdom.extend(item)
  return scdom
def findsubsets(s, n):
    return list(itertools.combinations(s, n))
main(matrix)
```