ON K-BROADCASTING IN GRAPHS

Bin Shao

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

March 2006

Your file Votre référence
ISBN: 978-0-494-16285-9
Our file Notre référence
ISBN: 978-0-494-16285-9

Abstract

On k-broadcasting in Graphs

Bin Shao, Ph.D.

Concordia University, 2006

Broadcasting is a fundamental information dissemination problem, wherein a message is sent from one vertex, the originator, to all other vertices in a graph. In k-broadcasting, an informed vertex can sends the message to at most k uninformed neighbors in each time unit. This thesis presents several algorithms to perform efficient k-broadcasting. The algorithm KBT generates the optimal k-broadcast scheme in trees, while the algorithm KBC finds the k-broadcast center of a given tree. This thesis presents an efficient heuristic for k-broadcasting. The heuristic has a low time complexity and generates fast k-broadcast schemes in many network topologies.

A k-broadcast graph G is a graph on n vertices where the k-broadcast time of G is $\lceil log_{k+1} n \rceil$. $B_k(n)$ stands for the minimum possible number of edges in a k-broadcast graph on n vertices. A k-broadcast graph on n vertices with $B_k(n)$ edges is a minimum k-broadcast graph, which is denoted by k-mbg. This thesis presents several new k-mbg's and an improved lower bound on $B_k(n)$.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Hovhannes Harutyunyan, for his insightful advice and invaluable encouragement, which have helped me through my studies at Concordia.

I would also thank Mr. Guotai Chen and Mr. Edward Marashlian, who have worked in the simulation of the algorithms presented in this thesis. Mr. Guotai Chen provided all the test results of the Tree Based Algorithm for Gossip, and Mr. Edward Marashlian provided the test results of the Tree Based Algorithm on the generalized chordal rings, the double fixed step graphs and the triple fixed step graphs.

Finally, I would like to express my gratitude to Ms. Kimberley Hamilton for her excellent editing and proofreading work, which have contributed to dramatically improve the expression of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The subject of information dissemination problems has a steadily growing body of literature, which is surveyed by [25], [44] and [48]. The k-broadcasting and the gossiping are two fundamental information dissemination problems. This thesis addresses how to efficiently perform k-broadcasting and gossiping in arbitrary networks.

1.1 Problem Statement

In ancient China, the beacon-tower system played a vital role in military communication. When the enemy approached a beacon tower on the border, the soldiers in the tower sent a signal by fires during the night or by smoke signals during the day. Upon seeing these signals, soldiers in other beacon towers also set a fire or smoke signal. Thus, in several hours, the alarm could spread hundreds or even thousands of kilometers from the border. Nowadays, computer networks, from local area networks to the Internet, have become essential to many aspects of modern society. For example, we can reach a friend in several minutes by sending an e-mail on the Internet. Also, with the help of a camera, we can talk with families face to face via the computer, no matter how far away they are.

The main purpose of these networks, whether the beacon-tower system or the Internet, is to share and spread information. Communication efficiency becomes particularly important when a computer network supports a distributed file or database system, where large amounts of information need to be disseminated among the computers in the network. There are many problems that could not be solved by a single processor in an acceptable amount of time. One solution is to divide the problem into subproblems that can be performed simultaneously in a parallel system. A single processor handles one of these subproblems. The results of certain subproblems must be transferred among these processors for further computing [67].

The performance of the information dissemination often determines the efficiency of a whole network or a parallel system. There are two approaches to reduce the delay of information dissemination: one is to reduce the amount of data being transferred, while the other is to minimize the delay of information spreading [75]. The first goal can be achieved by data compression or by reducing redundant information. This thesis tries to minimize the delay of information spreading by designing efficient algorithms and network topologies.

The study of information dissemination can be traced back to the following problem: "There are n ladies, and each one of them knows an item of scandal that is not known to any of the others. They communicate by telephone, and whenever two ladies make a call, they pass on to each other, as much scandal as they know at the time. How many calls are needed before all ladies know all the scandal?" [29]. This problem is the origin of the Gossip Problem, which is also the source of dozens of papers on the information dissemination problems in networks.

Most of the research discusses the gossip problem and/or the broadcast problem and their variants. Broadcasting is a process in which a single message is sent from one member of a network, the originator, to all other members, while in gossiping every member in a network has a message to send to all other members. A *call* refers to the action of messages being exchanged among a vertex and one or several of its neighbors. Both broadcasting and gossiping are performed by a series of calls over the communication lines of a network or egdes in a graph. A *round* refers to the set of parallel calls in the same time unit.

The communication modes precisely describe the different laws used to model real communications.

Given two neighbor vertices p and q in a graph, under *one-way* mode, only one message can travel between p and q, either from p to q or from q to p. Under *two-way* mode, two messages can use the link at the same time in opposite directions [48].

Depending on where the communication bottleneck occurs, communications in networks could be classified into three types [24]:

(1) If, during communication, a processor can only use one of its links, we call this situation *processor-bound* because processors cannot quickly relay messages and will

hamper the efficiency of the network. This pattern is also called 1-*port* or *whispering.*

(2) On the contrary, when a processor can use all of its links at the same time, communications are said to be *link-bound*, because it is now the number of links that limits communications. This pattern is also called *n-ports* or *shouting.*

(3) Between these two extremes, we have the case of *DMA-bound*, where a processor can only use k links at the same time.

The communication time T of sending a message between two adjacent vertices depends on the length L of the message [24]. Thus, T is often modeled as $T = \beta + L\tau$, where β stands for the start-up time and τ stands for the transmission time of a data of unit length [24]. In the constant model, we shall make the assumption that the time of communication between two vertices in a network is equal to one time unit, such that $T = 1$ [24].

In this thesis, the broadcast problem is discussed under two-way mode and constant model with a DMA-bound constraint. The broadcast problem with a DMA-bound constraint is also called the k-broadcast problem, wherein each call involves a caller who sends the message to as many as k of its neighbors. The gossip problem is discussed under two-way mode and constant model with a processor-bound constraint. More precisely, in both k-broadcasting and gossiping, each vertex can participate in only one call per round and each call requires one round. However, in k-broadcasting a vertex can communicate with as many as k of its neighbors, while in gossiping a vertex can only communicate with one of its neighbors.

Normally, a network can be modeled as a graph $G = (V, E)$, where the vertex-set

V represents the set of nodes and the edge-set E represents the set of communication links in a network. For the purpose of message dissemination, it is natural to assume that the network is represented by a connected graph. Two vertices $u \in V$ and $v \in V$ are *adjacent* if there is an edge $e \in E$, such that $e = (u, v)$. We may also say vertex u or v is a *neighbor* of the other vertex. The *degree* of a vertex refers to the number of neighbors of this vertex. The *degree* of a graph G is the maximum degree among all vertices in this graph. We use Δ to denote the degree of a graph. The distance between a vertex u and a vertex v, which is denoted by $dist(u, v)$, is the length of the shortest path between u and v. The *diameter* of a graph G, denoted by D is the maximal distance between any pair of vertices in the graph G.

An algorithm for a communication problem (such as gossiping or broadcasting) generates a communication scheme, which is a sequence of communication rounds. We use the number of rounds to measure the broadcast time and gossip time. The k-broadcast time $b_k(u, G)$, or simply $b_k(u)$, is the minimum k-broadcast time of graph G originated at vertex u. The k-broadcast time of graph G is defined as follows: $b_k(G) = max\{\, b_k(u, G) \mid u \in V\}$. $g(G)$ stands for the gossip time of graph G. A k-broadcast scheme or k-broadcast schedule is a series of calls that perform k-broadcast. A k-broadcast scheme that finishes the k-broadcasting in $b_k(u, G)$ is called an optimal k-broadcast scheme for the originator u in G.

This thesis focuses on how to improve the efficiency of information dissemination in networks. This goal can be achieved either by using efficient algorithms and heuristics or efficient network topologies. More specifically, an efficient topology should have less communication links, and it can efficiently perform information dissemination problems.

1.2 Contributions

This thesis addresses the efficient k-broadcasting in tree and arbitrary networks. Except where otherwise stated, all the results in Chapter 2, 3, 4 and 5 are original. The list below presents the main contributions of this thesis.

1. A linear algorithm for optimal k-broadcasting for a given originator in trees (Chapter 2)

2. A linear algorithm for determining the k-broadcast center of a given tree (Chapter 2)

3. Theorems on the structure of k-broadcast center in a tree (Chapter 2)

4. An efficient algorithm, named TBA, for k-broadcasting in arbitrary graphs (Chapter 3)

5. The proof that any 1-broadcast scheme from a corner vertex of the grid graph generates optimal broadcast time (Chapter 3)

6. An algorithm for gossip derived from TBA (Chapter 3)

7. A minimum 1-broadcast graph on 1023 vertices, a minimum 1-broadcast graph on 4095 vertices, and a new minimum 2-broadcast graph on 10 vertices (Chapter 4)

8. An improved lower bound on $B_k(n)$ (chapter 5)

9. A theorem on the monotonicity of the k-broadcast function $B_k(n)$ (chapter 5)

1.3 Commonly Used Topologies

This section presents commonly used network topologies, their k-broadcast times and their gossip times. Most of the results presented in this section were surveyed in [24], [44], [48] and [60]. All the figures in this section were presented in [24], [48] and [60]. Most of the previous results on k-broadcast time have been for k=1.

- *Arbitrary graphs*: It is well known that, for any graph on n vertices, $\lceil log_2 n \rceil \leq b_1(G) \leq n-1$ [24]. Because any vertex holding a message could only send it to one of its adjacent vertices, the number of informed vertices could at most be doubled in each round. Thus, at least $\lceil log_2 n \rceil$ rounds are needed for finishing 1-broadcasting. On the other hand, in 1-broadcasting, at least one vertex must be informed in each round. A situation in which no new vertex is informed means that the broadcasting has been completed. Therefore, 1-broadcasting takes at most $n-1$ rounds. For any graph of maximum degree Δ and diameter D, the formula $D \leq b_1(G) \leq \Delta D$ holds, since it is possible to broadcast in any *shortest path spanning tree* of G of height D and maximum degree Δ in at most ΔD rounds [25]. In [24], the following lemma is proved.

Lemma 1. *In any graph of diameter D, if three different vertices u, v_1 and v_2, with both v_1 and v_2 at a distance D from u, exist, then $b_1(G) \geq D+1$.*

We can derive k-broadcast time of an arbitrary graph based on these results of 1-broadcast time. For any graph on n vertices, $\lceil log_{k+1} n \rceil \leq b_k(G) \leq n-1$. Given a graph G with degree Δ, $b_k(G) = D$ when $k \geq \Delta$. When $k = \Delta - 1$, $D \leq b_k(G) \leq D+1$,

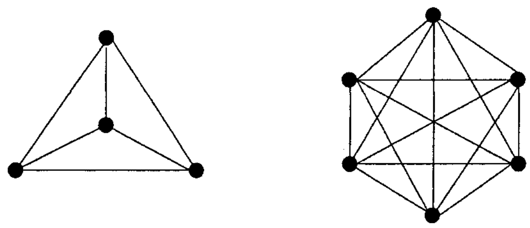

Figure 1: Complete graphs for n=4 and n=6

because after the first round, each informed vertex has at most $\Delta - 1 = k$ uninformed neighbors. Generally speaking, when $k < \Delta$, $D \leq b_k(G) \leq D \cdot \lceil \frac{\Delta}{k} \rceil$.

Some bounds on the gossip time are given in [24]: $\lceil log_2 n \rceil \leq b_1(G) \leq g(G) \leq 2b_1(G) - 1 \leq 2n - 3$. The inequality $b_1(G) \leq g(G)$ comes from the fact that a gossip scheme can be used as a 1-broadcast scheme. The inequality $g(G) \leq 2b_1(G) - 1$ comes from the fact that gossip can be performed in two phases by first collecting all messages in one vertex by accumulation, and then broadcasting the full information to all vertices.

- *The complete graph K_n*: Any vertex in a complete graph is linked to all other vertices (see Figure 1) [24]. Each vertex has a degree of $n - 1$, while the diameter is one and the number of edges is $n(n-1)/2$ [24]. It is easy to see that $b_k(K_n) = \lceil log_{k+1} n \rceil$, because any informed vertex can send the message to any of its k uninformed neighbors in each round [24]. The following results are shown in [52]: if n is even, $g(K_n) = \lceil log_2 n \rceil$, and if n is odd, $g(K_n) = \lceil log_2 n \rceil + 1$.

- *The path graph P_n*: The path of length n, denoted by P_n, is the graph whose vertices are all labeled by integers from 1 to n, and whose edges connect the vertex

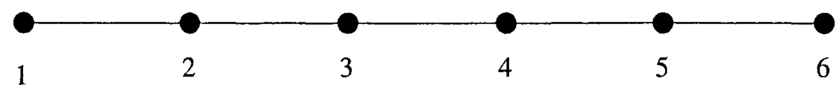

Figure 2: The path graph for n=6

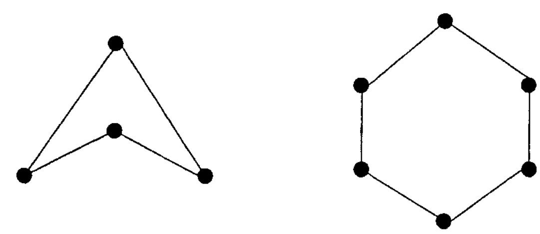

Figure 3: Cycle graphs for n=4 and n=6

labeled by integer i $(1 \leq i \leq n)$ with the vertex labeled by $i+1$ [48]. P_n has n vertices, a diameter of $n-1$ and a maximum degree of 2 (see Figure 2) [48]. When the originator u is at either end of P_n, $b_k(u, P_n)$ is at the maximum. In such a case $b_k(u, P_n) = b_k(P_n) = n-1$.

- *The cycle graph* C_N: Each vertex is linked to only two neighbors, thus the degree is two (see Figure 3) [24]. In the *cycle* graph C_n, the k-broadcast time is $\lceil \frac{n}{2} \rceil$ when $k = 1$ [24]. When $k \geq 2$, the k-broadcast time in C_n is $\lfloor \frac{n}{2} \rfloor$, which is the diameter of the graph. For the gossip problem in a *cycle* graph: if n is even, $g(C_n) = \frac{n}{2} = D$; if n is odd, $g(C_n) = \frac{n-1}{2} + 2 = D + 2$ [21].

- *The d-grid graph*: $GD_N = P_{n_1} \square \cdots \square P_{n_i} \square \cdots \square P_{n_d}$, for $1 \leq i \leq d$, where P_{n_i} is a path on n_i vertices [24]. A *2-grid* is shown in Figure 4. The following result is shown in [19].

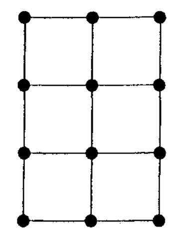

Figure 4: A 2×6 2-grid graph on 12 vertices

$$b_1(P_{n_1}\square \cdots \square P_{n_i}\square \cdots \square P_{n_d}) = \sum_{i=1}^{d} n_i - d = D \tag{1}$$

A 2-grid with m columns and n rows is denoted by $G_{m,n}$. The worst case of the k-broadcast in the grid graph is that the originator is on a corner of the graph. The 1-broadcast time of a grid graph $G_{m,n}$ is $m + n - 2$ [19]. When $k \geq 2$, in each round an informed vertex has at most 2 uninformed neighbors when the originator is on a corner. Therefore, the k-broadcast time is the diameter of the graph, which is still $m + n - 2$. The following results on gossip time are proven in [64]: if i is odd, then $g(P_3\square \cdots \square P_3\square P_i\square P_3\square \cdots \square P_3) = D + 1$; otherwise, $g(P_{n_1}\square \cdots \square P_{n_d}) = D$.

- *The d-Torus graph*: $T_d = C_{p_1}\square \cdots \square C_{p_i}\square \cdots \square C_{p_d}$, for $1 \leq i \leq d$, where C_{p_i} is a cycle on p_i vertices [24]. A 2-*Torus* is shown in Figure 5. The optimal 1-broadcast time of the two-dimensional torus graph, denoted by $Torus(m, n)$, is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil$ when m or n is even; and it is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 1$ when both m and n are odd [19]. For 1-broadcast on multidimensional torus, $D \leq b_1(T_d = C_{p_1}\square \cdots \square C_{p_d}) \leq D + \max(0, s - 1)$, where s is the number of odd dimensions in $C_{p_1}\square \cdots \square C_{p_d}$ [24].

When $k \geq 2$, we can achieve the optimal k-broadcast time in 2-*Torus* by the

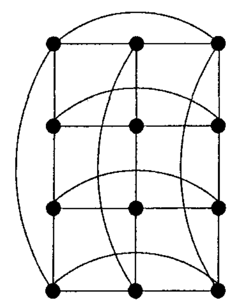

Figure 5: A 3×4 2-torus graph on 12 vertices

following scheme: first, an informed vertex sends the message to its uninformed column neighbors (if it has such neighbors). Then, after all vertices on the column at which the originator is located are informed, each informed vertex sends the message to uninformed row neighbors (if it has such neighbors). This scheme gives $b_k(Torus(m,n)) = \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor = D$.

For gossip, $D \leq g(T_d) \leq D + 2d$ [21].

- *Hypercube graph H_d*: The d-dimensional hypercube has $n = 2^d$ vertices and $d2^{d-1}$ edges. Each vertex corresponds to an d-bit binary string, and two vertices are linked with an edge if and only if their binary strings differ by precisely one bit [24]. As a consequence, each vertex is adjacent to d other vertices, one for each bit position [24]. Any d-dimensional hypercube could be derived from two $(d-1)$-dimensional hypercube graphs [60]. Figure 6 presents the construction of a 4-dimensional hypercube (b) from two 3-dimensional hypercubes (a). Dashed edges form a match between the two 3-dimensional cubes. The diameter of H_d is d. 1-broadcasting in H_d can be done in d rounds by using the following scheme: at step

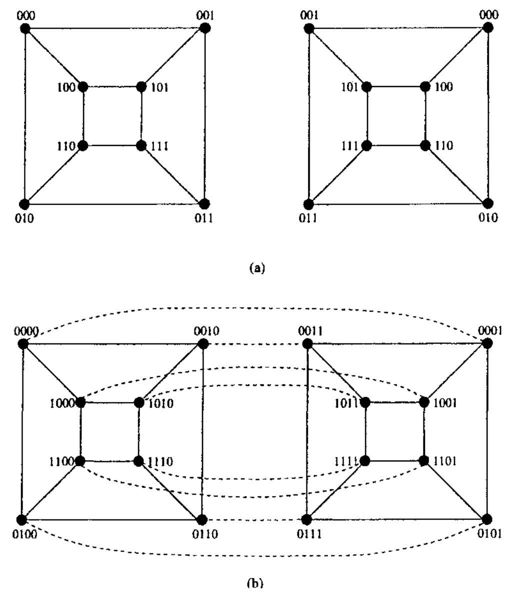

Figure 6: Hypercube graphs

i, each informed vertex sends the message in dimension i $(1 \leq i \leq d)$ [24]. Gossiping in H_d can also be done in d rounds, and the gossip scheme is the same [24].

• *Cube connected cycles* CCC_d : The CCC_d (see Figure 7) is a modification of H_d, where the d-dimensional CCC_d is constructed from the d-dimensional hypercube by replacing each vertex of the hypercube with a cycle of d vertices [24]. When $d > 3$, the diameter of the CCC_d is $2d + \lfloor d/2 \rfloor - 2$ [66]. A straightforward algorithm gives a 1-broadcast time of $\lceil \frac{5d}{2} \rceil - 1$ [63]. This algorithm first relays the message to the hypercube neighbor, then to the right neighbor on the ring, then to the left one.

(1) If d is even, then $\lceil \frac{5d}{2} \rceil - 1 = D + 1$. Therefore

$$D \leq b_1(CCC_d) \leq D + 1 \tag{2}$$

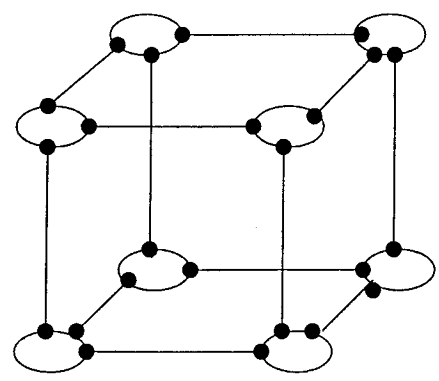

Figure 7: The CCC_3 graphs

with the exception of $d = 4$, where Lemma 1 applies, and therefore

$$b_1(CCC_4) = D + 1 \tag{3}$$

(2) If d is odd: $\lceil \frac{5d}{2} \rceil - 1 = D + 2$. Lemma 1 still applies, and therefore

$$D + 1 \leq b_1(CCC_d) \leq D + 2 \tag{4}$$

The upper and lower bounds on $g(CCC_d)$ are presented in [46] and [48] respectively: $\lceil \frac{5d}{2} \rceil - 1 \leq g(CCC_d) \leq 5\lceil \frac{d}{2} \rceil$.

• *The butterfly graph BF_d*: The d-dimensional butterfly BF_d has $d2^d$ vertices (see Figure 8) [24]. Each vertex is labeled with a pair of numbers (l, x). l represents the level $(0 \leq l \leq d-1)$, and $x = x_0 \cdots x_{d-1}$ is a d-bit binary string called the *position-within-level* [24]. Two vertices (l_0, x_0) and (l_1, x_1) in BF_d are linked by an edge if and only if $l_1 = l_0 + 1$ *mod* d and either $x_0 = x_1$ or x_0 and x_1 differ by the l_0th bit [60]. BF_d has degree four and diameter $\lfloor 3d/2 \rfloor$ [24].

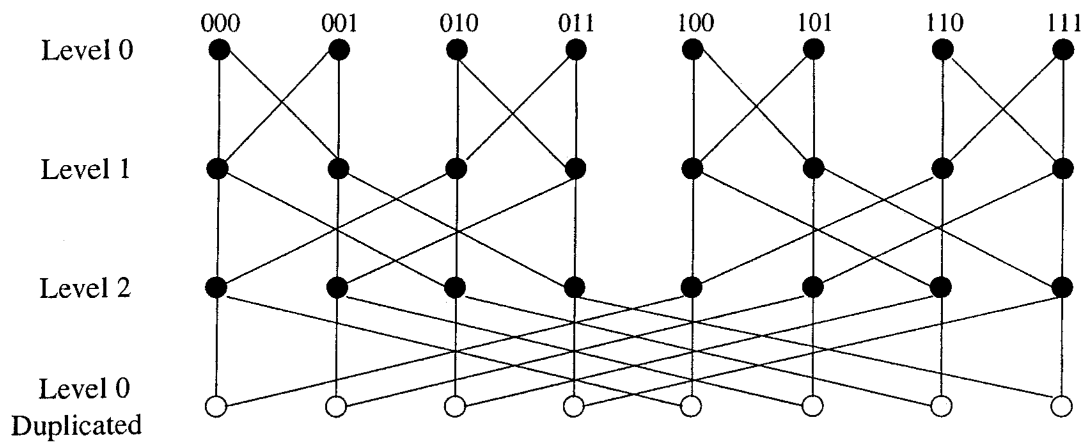

Figure 8: The BF_3 graphs

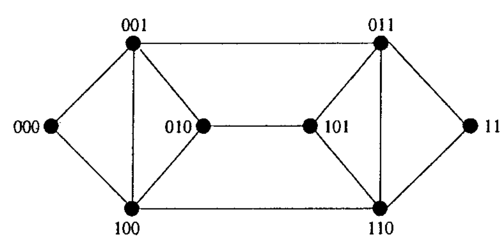

Figure 9: The $UB(2,3)$ graph

The following result is presented in [51]:

$$1.7417d \leq b_1(BF_d) \leq 2d - 1 \tag{5}$$

The upper and lower bounds on $g(BF_d)$ are presented in [46] and [48] respectively: $1.7417d \leq g(BF_d) \leq 5\lceil \frac{d}{2} \rceil$.

• *The de Bruijn graph UB(d, D)*: The de Bruijn digraph $B(d, D)$ with indegree and outdegree d and diameter D, is the digraph whose $N = d^D$ vertices are denoted by the words of length D on an alphabet of d letters [24]. There is a direct edge

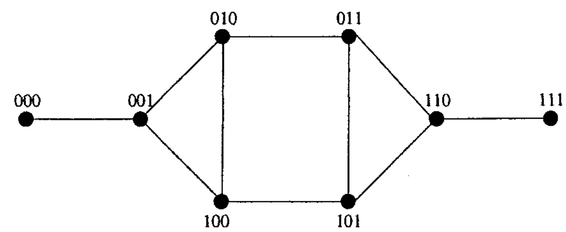

Figure 10: The SE_3 graphs

from each vertex $(x_0x_1 \cdots x_{D-1})$ to $(x_1 \cdots x_{D-1}\gamma)$, where γ could be any letter in the alphabet [24]. The de Bruijn graph $UB(d, D)$ is obtained by removing the edge orientation in $B(d, D)$ [24]. For $UB(2, D)$ (see Figure 9), the following result is proved in [51]: $1.3171D \leq b_1(UB(2, D))$. The following upper bound is shown in [8]: $b_1(UB(2, D)) \leq \frac{3}{2}(D + 1)$.

For gossip, $1.3171D \leq g(UB(2, D)) \leq 3D + 2$ [48].

• *The shuffle-exchange graph SE_d*: The d-dimensional shuffle-exchange graph has $n = 2^d$ vertices (see Figure 10) [24]. Each vertex corresponds to a unique d-bit binary number, and two vertices u and v are linked by an edge, if either u and v differ in precisely the last bit, or u is a left or right cyclic shift of v [60]. The 1-broadcasting time for SE_d is $2d - 1$ [47], which is equal to the diameter of SE_d.

The bounds on gossip time in SE_d is: $2d - 1 \leq g(SE_d) \leq 4d - 3$ [47].

• *The star graph S_n*: An n-Star Graph has $N = n!$ vertices, where these vertices are represented by all the permutations on n symbols [24]. Each vertex is linked to vertices generated by a set of generators g, which consists of $n - 1$ transpositions $\{g_2, g_3, g_4, \cdots, g_n\}$ where g_1 is the transposition that switches the ith element with the

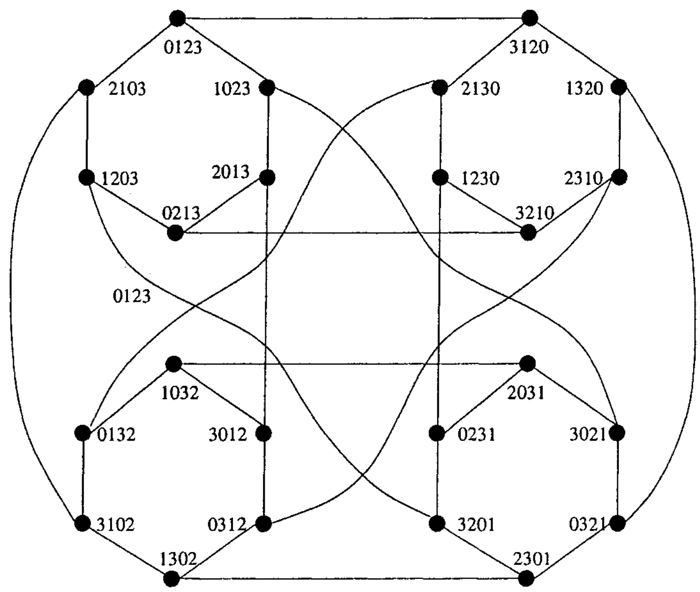

Figure 11: The star graph S_4

first, and leaves the remaining elements in their original positions (see Figure 11) [24]. In [1], it is proved that the diameter of the star graph is $\lfloor 3(n-1)/2 \rfloor$. The following results are shown in [9]:

$$\lceil log_2 N \rceil \leq b_1(S_n) \leq \lceil log_2 N \rceil + \lceil 7n/4 \rceil + \lceil log_2 n \rceil \tag{6}$$

$$\lceil log_2 N \rceil \leq g(S_n) \leq \lceil log_2 N \rceil + 2n \tag{7}$$

• *The generalized chordal rings $GCR_n(a, b, c)$*: The 1-broadcasting in the generalized chordal rings is discussed in [11]. Let n be an even positive integer, and a, b, c, three distinct odd positive integers smaller than n. The *generalized chordal ring* (GCR) $GCR_n(a, b, c)$ has the vertex set $V = V_0 \cup V_1$, where $V_0 = \{0, 2, 4, \cdots, n-2\}$ and $V_1 = \{1, 3, 5, \cdots, n-1\}$. Each vertex $i \in V_0$ is adjacent to the vertices $i + a$, $i + b$,

$i + c \in V_1$ (vertex addition is always modulo n). Equivalently, each vertex $j \in V_1$ is adjacent to the vertices $j - a$, $j - b$, $j - c \in V_0$. Every generalized chordal ring of diameter D admits a 1-broadcast (from any starting vertex) that informs all vertices in time at most $D + 2$ [11]. In other words, the 1-broadcast of a generalized chordal ring with diameter D could be D, D+1 or D+2.

The maximum number of vertices in a generalized chordal ring of diameter D is equal to $(3D^2 + 1)/2$ when D is odd, and is at most $3D^2/2$ and at least $3D^2/2 - D$ when D is even [80]. The generalized chordal rings with the greatest number of vertices is optimal. For the optimal generalized chordal rings, the 1-broadcast time is $D + 1$ when D is even, and is $D + 2$ when D is odd [11]. GCR_n (1, -1, $3D$) on $3D^2 + 1/2$ vertices has diameter D [69]. Therefore these graphs are optimal [11]. GCR_n(1, -1, $3D$+1) on $3D^2/2 - D$ vertices has diameter D [69]. There graphs are generally believed to be optimal, although the upper bound on the number of vertices cannot be attained [69].

- *The optimal double fixed step graphs $G_{2,D}$ and triple fixed step Graphs $G_{3,D}$*: The 1-broadcast times of $G_{2,D}$ and $G_{3,D}$ are presented in [61]. For a positive integer n and a set of positive, pairwise distinct integers $\{s_1, s_2, \cdots, s_k\}$, the *multiple fixed step graph* $G(n, s_1, s_2, \cdots, s_k)$ (also called a *distributed loop graph*) is a graph on vertices $\{0, 1, \cdots, n - 1\}$ such that for any vertex u there is an edge between u and vertex $u + s_i \pmod n$ for $1 \leq i \leq k$. In general, the problem of determining the diameter of multiple fixed step graphs is difficult. However, more is known for the cases $k = 2$ and $k = 3$ which have been called *double fixed step* and *triple fixed step* graphs [17] [81]. Any double fixed step graph of diameter D has at most $2D^2+2D+1$ vertices, while the graph $G(2D^2+2D+1, D, D+1)$ is of diameter D [80]. The graph $G(2D^2+2D+1, D, D+1)$ is called an *optimal double fixed step graph* and is denoted by $G_{2,D}$ in [61]. The graph $G(3D^3+3D+1, D, D+1, 2D+1)$ has been shown to have diameter D and it shares many properties the optimal double fixed step graphs [81]. So, this graph is denoted by $G_{3,D}$ in [61]. It is proved that $b_1(G_{2,D}) = D+2$ and $b_1(G_{3,D}) = D+3$ [61].

Chapter 2

Optimal k-broadcasting in Trees

This chapter presents two linear algorithms on optimal k-broadcasting in trees. Algorithm KBT determines $b_k(u, T)$ for the originator u in a given tree T, while algorithm KBC locates the k-broadcast center for a given tree. One by-product of algorithm KBT is the optimal k-broadcast scheme for the originator u.

2.1 k-broadcasting for a Given Originator

This section presents an algorithm called KBT, k-broadcast time, which is a linear algorithm to generate $b_k(u, T)$ and the optimal k-broadcast scheme for a given originator u in a tree T.

The problem of k-broadcasting in trees is discussed in [33], [55], [56], [71] and [76]. Given an informed vertex v and its p uninformed neighbors v_1, v_2, $\cdots$, v_p, $T(v_i)$ $(1 \leq i \leq p)$ stands for the subtree that is rooted at v_i and does not contain the originator v. Assume the p uninformed children are labeled such that $b_1(v_1, T(v_1)) \geq b_1(v_2, T(v_2)) \geq \cdots \geq b_1(v_p, T(v_p, v))$, then the optimal sequence of 1-broadcast calls is such that v initially calls v_1, then v_2, then v_3, etc.

Similarly, given the originator u and its p neighbors c_1, c_2, $\cdots$, c_p, where $b_k(c_i, T(c_i)) \geq b_k(c_{i+1}, T(c_{i+1}))$ $(1 \leq i < p)$, the optimal k-broadcast scheme for vertex u will be sending the message to c_1, c_2, $\cdots$, c_k at round 1, to c_{k+1}, c_{k+2}, $\cdots$, c_{2k} at round 2, in general to $c_{(i-1)k+1}$, $c_{(i-1)k+2}$, $\cdots$, c_{ik} at round i, for $1 \leq i \leq \lceil \frac{p}{k} \rceil$ (see Figure 12). By using this k-broadcast scheme, after c_i $(1 \leq i \leq p)$ is informed, all vertices in $T(c_i)$ can be informed in $max\{b_k(c_{(i-1)k+1}, T(c_{(i-1)k+1})) + i\}$ $(1 \leq i \leq \lceil \frac{p}{k} \rceil)$ rounds.

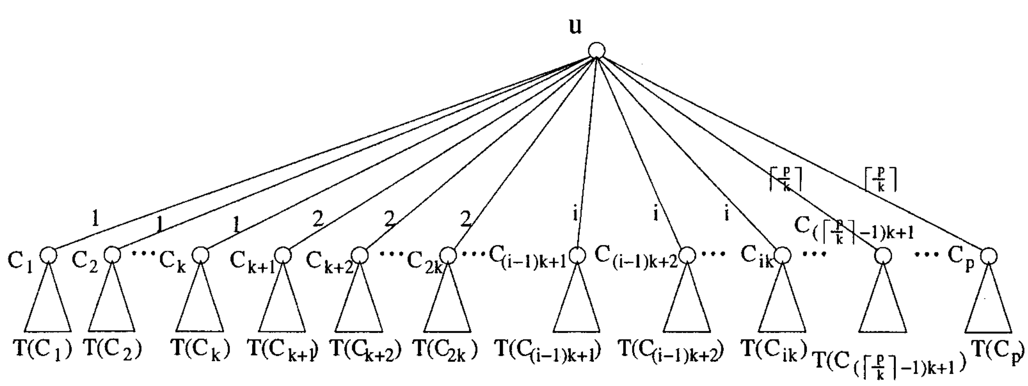

Figure 12: The optimal k-broadcasting

The algorithm KBT first performs BFS (breadth first search) from the originator u, during which all vertices are labeled with their distance to vertex u and all vertices are pushed into a *stack* in the order that they are visited. If $dist(p, u) = dist(q, u) + 1$ and vertex p is a neighbor of vertex q, then p is a child of q and q is the parent of p. After the BFS, KBT inductively calculates the *weights* of all vertices in T. Let $w(v)$

represent the weight of vertex v. If v has no children, then $w(v) = 0$. Otherwise, assume v has p children $c_1, c_2, \cdots, c_p$, and these children are ordered such that $w(c_j) \geq w(c_{j+1})$, for $1 \leq j < p$, then $w(v) = max\{w(c_{(i-1)k+1}) + i\}$, for $1 \leq i \leq \lceil \frac{p}{k} \rceil$. We will see that for each vertex v in T, $w(v) = b_k(v, T(v))$, and so $w(u) = b_k(u, T)$ where u is the originator. Given p integers, the procedure *OrderWeight* calculates $max\{c_{(i-1)k+1} + i\}$ $(1 \leq i \leq \lceil \frac{p}{k} \rceil)$ in $O(p)$ time, where $c_i \geq c_{i+1}$ $(1 \leq i < p)$. In KBT, given a vertex v and the weights of its p children, the procedure *OrderWeight* is used to calculate the weight of v in $O(p)$ time.

Algorithm KBT

Input: tree T, originator u, k

Output: $b_k(u, T)$

1. Stack *Vertex* $\leftarrow \emptyset$;

2. For each vertex v in T, $v.childrenset \leftarrow \emptyset$;

3. Perform BFS from u;

   3.1. During BFS, push all the vertices into *Vertex* in the order that they are visited;

   3.2. During BFS, for each vertex v, $v.dist$ = the distance between u and v;

4. **While** *Vertex* is not empty;

   4.1. v = *Vertex.pop()*;

   4.2. **if** $v.childrenset = \emptyset$, **then** $v.weight = 0$;

4.3. **else** $v.weight$ = Procedure $OrderWeight$(weights of all children of v, k);

4.4. **For** any neighbor w of v,

4.5. **if** $v.dist = w.dist + 1$, **then** $w.childrenset \leftarrow v$.

5 Output $u.weight$; //the originator u is the last vertex in Stack *Vertex*.

Procedure *OrderWeight*

Input: p integers, k.

Output: Assume the p integers $c_1, c_2, \cdots, c_p$ are ordered such that $c_1 \geq c_2 \geq \cdots \geq c_p$, output $max_{1 \leq i \leq \lceil \frac{p}{k} \rceil}\{c_{(i-1)k+1} + i\}$.

1 Let MAX = $max_{1 \leq i \leq p}\{c_i\}$;

2 Create $\lceil \frac{p}{k} \rceil$ buckets. These buckets are numbered from 0 to $\lceil \frac{p}{k} \rceil - 1$;

3 For each integer c, if $c = MAX - i$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$, then put c into bucket i;

4 Let the number of integers in the ith bucket be $NUM(i)$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$, $SUM(i) = \sum_{j=0}^{i} NUM(j)$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$;

5 Output $max_{0 \leq i < \lceil \frac{p}{k} \rceil}\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\}$;

The following discussions confirm the validity of the procedure OrderWeight and the algorithm KBT.

Lemma 2. $max_{0 \leq i < \lceil \frac{p}{k} \rceil}\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\} = max_{1 \leq i \leq \lceil \frac{p}{k} \rceil}\{c_{(i-1)k+1} + i\}$.

Proof. Since $c_{(i-1)k+1} \geq c_{(i-1)k+2} \geq \cdots \geq c_{(i-1)k+k}$, then $c_{(i-1)k+1} + \lceil \frac{(i-1)k+1}{k} \rceil \geq c_{(i-1)k+2} + \lceil \frac{(i-1)k+2}{k} \rceil \geq \cdots \geq c_{(i-1)k+k} + \lceil \frac{(i-1)k+k}{k} \rceil$. Subsequently, $max_{1 \leq i \leq \lceil \frac{p}{k} \rceil}\{c_{(i-1)k+1} + i\} = max_{1 \leq i \leq p}\{c_i + \lceil \frac{i}{k} \rceil\}$.

For an integer c_j in the ith buckets ($1 \leq i \leq \lceil \frac{p}{k} \rceil$ and $1 \leq j \leq p$), $c_j + \lceil \frac{j}{k} \rceil \leq c_j + \lceil \frac{SUM(i)}{k} \rceil = \lceil \frac{SUM(i)}{k} \rceil + MAX - i$. Thus, $\lceil \frac{SUM(i)}{k} \rceil + MAX - i$ equals the maximal $c_j + \lceil \frac{j}{k} \rceil$ for all vertices in the ith bucket. Therefore, $max_{0 \leq i < \lceil \frac{p}{k} \rceil}\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\}$ $= max_{1 \leq i \leq p}\{c_i + \lceil \frac{i}{k} \rceil\}$. □

Lemma 3. *For each vertex v in T, $w(v) = b_k(v, T(v))$.*

Proof. When v is a leaf, $T(v)$ is a single vertex tree, and $w(v) = b_k(v, T(v)) = 0$. If all the p children of vertex v, denoted by x_1, x_2, $\cdots$, x_p, are leaves, then $w(x_1) = w(x_2) = \cdots = w(x_p) = 0$. Therefore, $w(v) = max_{1 \leq i \leq \lceil \frac{p}{k} \rceil}\{w(x_{(i-1)k+1}) + i\} = \lceil \frac{p}{k} \rceil$ $= b_k(v, T(v))$. In general, assume that for any child c of v, $w(c) = b_k(c, T(c))$. Let c_1, c_2, $\cdots$, c_p stand for the children of v, where $w(c_j) \geq w(c_{j+1})$ ($1 \leq j < p$). These p children can be divided into $\lceil \frac{p}{k} \rceil$ groups, where vertices $c_{(i-1)k+1}$, $c_{(i-1)k+2}$, $\cdots$, c_{ik} ($1 \leq i \leq \lceil \frac{p}{k} \rceil$) belong to the ith group. In the optimal k-broadcasting in $T(v)$, during the ith round after v is informed, vertex v sends the message to the vertices in the ith group. Therefore, the time needed to inform all the vertices in $T(c_{(i-1)k+1})$, $T(c_{(i-1)k+2})$, $\cdots$, $T(c_{ik})$ is $b_k(c_{(i-1)k+1}, T(c_{(i-1)k+1})) + i = w(c_{(i-1)k+1}) + i)$ ($1 \leq i \leq \lceil \frac{p}{k} \rceil$). Thus, the time needed to inform all the descendants of v is: $b_k(v, T(v))$ $= max_{1 \leq i \leq \lceil \frac{p}{k} \rceil}\{b_k(c_{(i-1)k+1}, T(c_{(i-1)k+1})) + i\} = max_{1 \leq i \leq \lceil \frac{p}{k} \rceil}\{w(c_{(i-1)k+1}) + i\} = w(v)$. Consequently, for any vertex v in T, $w(v) = b_k(v, T(v))$. □

Based upon Lemma 3, we have the following theorem:

Theorem 1. *For the originator u in tree T, $w(u) = b_k(u,T)$, where $w(u)$ represents the weight of u assigned by KBT.*

The k-broadcast scheme for the originator u is defined as follows: if a vertex v is informed at time t, it sends the message to vertices $c_{(i-1)k+1}$, $c_{(i-1)k+2}$, $\cdots$, c_{ik} during the round $t+i$, $1 \leq i \leq \lceil \frac{p}{k} \rceil$, where $w(c_1) \geq w(c_2) \geq \cdots \geq w(c_p)$ for all children c_1, c_2, $\cdots$, c_p of v. This leads us to the following theorem:

Theorem 2. *The algorithm KBT generates an optimal k-broadcast scheme for a given originator u in a given tree T.*

The time complexity of BFS is $O(|E|)$, where E is the set of edges in T. The time complexity of BFS in T is $O(|E|) = O(|V|)$, where V is the set of vertices in T. By using the *OrderWeight* procedure, the time needed to calculate the *weight* of a vertex with degree d is $O(d)$. Let the degree of the ith vertex in T be d_i, for $1 \leq i \leq |V|$. Then, the time needed to calculate the *weights* of all the vertices is $\sum_{i=1}^{|V|} O(d_i)$. Since $\sum_{i=1}^{|V|} d_i = 2|E|$, the complexity of calculating *weight* is $O(|E|) = O(|V|)$. Therefore, the total time complexity of the algorithm KBT is $O(|V|)$.

Figure 13 illustrates the performance of algorithm KBT for 2-broadcasting in a tree. Figure 13 (a) presents the original tree, wherein vertex u is the originator. Figure 13 (b) presents the tree with labels on all the vertices, where the label of a vertex is its distance from the originator. In Figure 13 (c), the number assigned to a vertex is the *weight* of the vertex in KBT. We can see that $b_2(u,T) = w(u) = 4$.

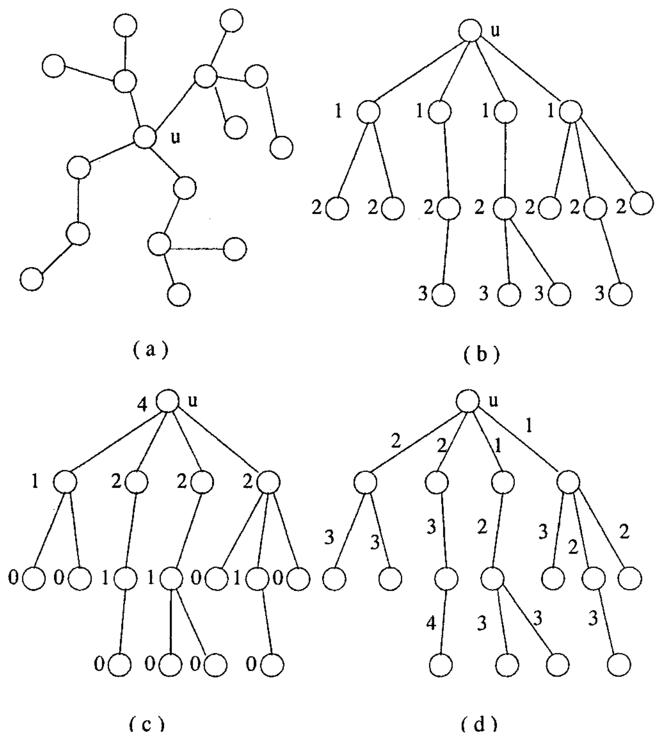

Figure 13: The performance of the algorithm KBT

Figure 13 (d) presents the 2-broadcast scheme generated by KBT, which is an optimal 2-broadcast scheme.

2.2 k-broadcast Center in Trees

$BC_k(G)$ stands for the k-broadcast center of $G = (V, E)$, which is defined as the set of vertices that have minimum k-broadcast time: $BC_k(G) = \{u \mid b_k(u) = min\{b_k(v, G),\ v \in V\},\ u \in V\}$. This section proves that the $BC_k(T)$ of an arbitrary tree T is a star graph. It also proves that the k-broadcast time of any vertex v in a tree T is the sum of the shortest distance from v to a vertex in $BC_k(T)$ and the

k-broadcast time of T. The algorithm in [76] determines the 1-broadcast center in a tree. This section derives an algorithm to calculate the k-broadcast center in a tree.

2.2.1 Theorems on k-broadcast Center

Lemma 4. *In any tree T, $BC_k(T)$ is a connected subgraph of T.*

Proof. Assume that the $BC_k(T)$ of tree T is not a connected graph. Then, there must exist two vertices u and v, so that both of them belong to $BC_k(T)$, and every vertex on the path between them does not belong to $BC_k(T)$. Assume that vertex a is the neighbor of vertex v on the path. Thus, $a \notin BC_k(T)$. Figure 14 illustrates such a situation. Let $dist(u,v) = d$ and $b_k(u,T) = b_k(v,T) = t$. t_v denotes the k-broadcast time of tree T_v (see Figure 14) originated at v, and t_a denotes the k-broadcast time of tree T_a (see Figure 14) originated at a. When considering the k-broadcasting originated at u, all vertices in T_v must be informed through v. Thus, $b_k(u,T) \geq dist(u,v) + b_k(v,T_v)$. Since $b(u,T) = t$, $b_k(v,T_v) = t_v$ and $dist(u,v) = d$, then $t_v \leq t - d$. Similarly, since $b(v,T) = t$ and $dist(v,a) = 1$, then $t_a \leq t - 1$. Then there is a k-broadcast scheme from originator a so that $b(a,T) = t$. In this scheme, vertex a sends the message to v first, and then finishes the k-broadcasting in T_a in $t-1$ time units. Meanwhile, vertex v can finish the k-broadcasting in T_v in $t-d$ time units. This contradicts the assumption that $a \notin BC_k(T)$. Therefore, in any tree T, $BC_k(T)$ is a connected subgraph of T. □

Lemma 5. *The diameter of $BC_k(T)$ is less than or equal to 2.*

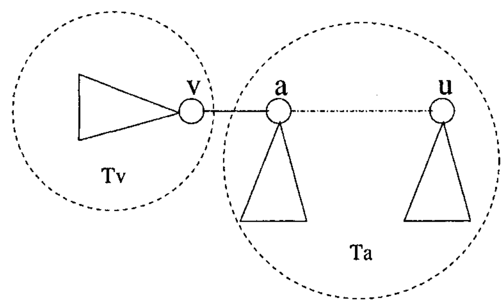

Figure 14: Figure for the proof of lemma 4

Proof. Assume there is a $BC_k(T)$ with a diameter greater than 2. There is a path with a length of three and all the four vertices on the path belong to $BC_k(T)$. In Figure 15, vertices a, b, c and d belong to $BC_k(T)$. Let $b_k(a,T) = b_k(b,T) = b_k(c,T) = b_k(d,T) = t$. t_c denotes the k-broadcast time of T_c (see Figure 15) originated at c, and t_b denotes the k-broadcast time of T_b (see Figure 15) originated at b. Because vertex d belongs to $BC_k(T)$ and $dist(b,d) = 2$, then $t_b \leq t-2$. Because vertex a belongs to $BC_k(T)$ and $dist(a,c) = 2$, then $t_c \leq t-2$. Then, there is a k-broadcast scheme for vertex c so that the k-broadcast time of c in T is less than t. In this scheme, vertex c first sends the message to vertex b, then vertex c finishes the k-broadcast in T_c in $t-2$ time units. Meanwhile, vertex b finishes the k-broadcast in T_b in $t-2$ time units. Thus, $b_k(c,T) \leq t-1$. This directly contradicts $b_k(c,T) = t$. Therefore, the diameter of $BC_k(T)$ is less than or equal to 2. □

From Lemma 4 and Lemma 5, we can derive the following theorem:

Theorem 3. *The $BC_k(T)$ of an arbitrary tree T is a star.*

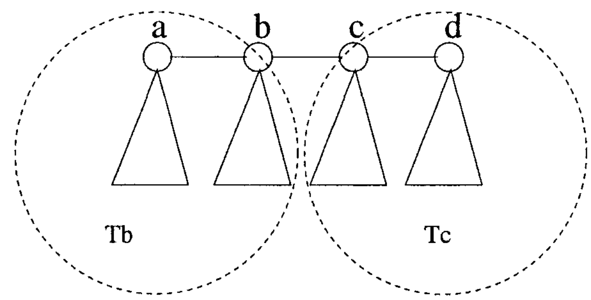

Figure 15: Figure for the proof of lemma 5

In 1-broadcasting, there are at least two vertices in $BC_1(T)$ in any tree T. However, in k-broadcast ($k \geq 2$), $BC_k(T)$ can be a single vertex. Considering the 2-broadcasting in the tree shown in Figure 16, only vertex u belongs to $BC_2(T)$. The 2-broadcast time of u is 3, and the 2-broadcast time of any other vertex in the tree is at least 4. In fact, the tree in Figure 16 is a 3-nomial tree of dimension 3. The *b-nomial* tree T_b^m of dimension m has b^m vertices. T_b^0 is a single vertex. For $m \geq 1$, the tree T_b^m is obtained from b copies of T_b^{m-1} by connecting the roots of $b-1$ copies of T_b^{m-1} to the root u of the remaining copy of T_b^{m-1}. This vertex u is the root of T_b^m. When considering the k-broadcasting from the root of T_{k+1}^m, a $(k+1)$-nomial tree on $(k+1)^m$ vertices, clearly, the root of T_{k+1}^m can k-broadcast to $(k+1)^m$ vertices (including itself) in m time units. Moreover, the root is the only vertex that belongs to $BC_k(T_{k+1}^m)$.

Theorem 4. *Given a vertex v that does not belong to $BC_k(T)$, and the shortest distance from v to a vertex u in $BC_k(T)$ is $dist(u, v)$, $b_k(v, T) = b_k(u, T) + dist(u, v)$.*

Proof. In Figure 17, vertex u belongs to $BC_k(T)$, but vertex v and x do not. Vertex

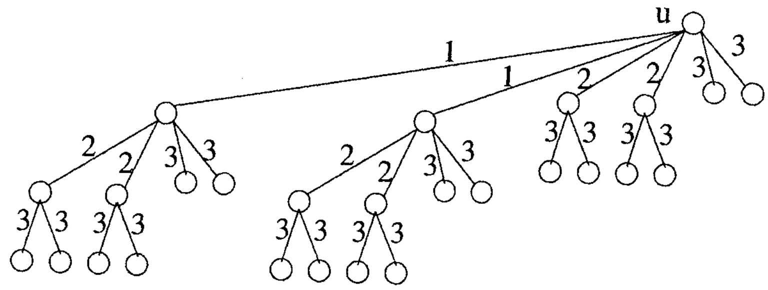

Figure 16: T_3^3: Only vertex $u \in BC_2(T_3^3)$

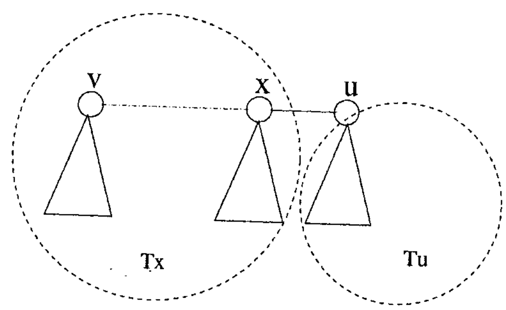

Figure 17: Figure for the proof of theorem 4

x is a neighbor of vertex u, and x is on the shortest path between u and v. Any vertex on this path does not belong to $BC_k(T)$, except for vertex u.

Clearly, when vertex v is the originator, it can first send the message to vertex u, then u can finish the k-broadcasting in tree T in $b_k(u,T)$ time units. Therefore, $b_k(v,T) \leq b_k(u,T) + dist(u,v)$.

Now we must prove that $b_k(v,T) \geq b_k(u,T) + dist(u,v)$. Let $b_k(u,T_u) = t_u$ and $b_k(x,T_x) = t_x$, where T_u and T_x are subtrees of T that are rooted at x and u respectively (see Figure 17). Assume $b_k(v,T) < b_k(u,T) + dist(u,v)$. Let $b_k(v,T) =$

$b_k(u,T) + dist(u,v) - p$ where $p \geq 1$, then $t_u \leq b_k(u,T) - p$. Since u belongs to $BC_k(T)$ and $dist(x,u) = 1$, then $t_x \leq b_k(u,T) - 1$. Thus, there is a k-broadcast scheme for vertex x that completes the k-broadcasting in $b_k(u,T)$ time units. In this scheme, vertex x first sends the message to u. Next, the k-broadcast in T_u and T_x can be completed in $b_k(u,T) - p$ time units and $b_k(u,T) - 1$ time units respectively. So $x \in BC_k(T)$. This is a contradiction. Therefore, $b_k(v,T) \geq b_k(u,T) + dist(u,v)$.

Then, we can draw the conclusion that $b_k(v,T) = b_k(u,T) + dist(u,v)$. □

2.2.2 An Algorithm to Determine the k-broadcast Center

The algorithm KBC (k-broadcast center) determines the k-broadcast center of a given tree T. One of its by-products is the k-broadcast time of each vertex in T. Let T' stand for the subtree of T obtained by removing all the leaves of T. For a vertex $u \in T$, $u.label$ refers to the label of vertex u. The algorithm KBC first labels all leaves of T with 0. Then, it labels each leaf of T' as follows: given a leaf u of T' and its p labeled neighbors from T c_1, c_2, $\cdots$, c_p, where $c_1.label \geq c_2.label \geq \cdots \geq c_p.label$, then $u.label = max\{c_{(i-1)k+1}.label + i\}$, for $1 \leq i \leq \lceil \frac{p}{k} \rceil$. Given a vertex u and its unsorted p labeled neighbors, the algorithm KBC use the *Orderweight* procedure, presented in Section 2.1, to calculate $u.label$ in $O(p)$ time. After this, KBC removes a leaf of T' with the minimal label. Let such a leaf be v, and s be the neighbor of v in T'. If s is now a leaf of T', then KBC labels s. The algorithm KBC keeps removing the leaf of T' with the minimal label until there is only one vertex in T'. Let the last vertex be w, and c_1, c_2, $\cdots$, c_p are its p labeled neighbors. Then the algorithm

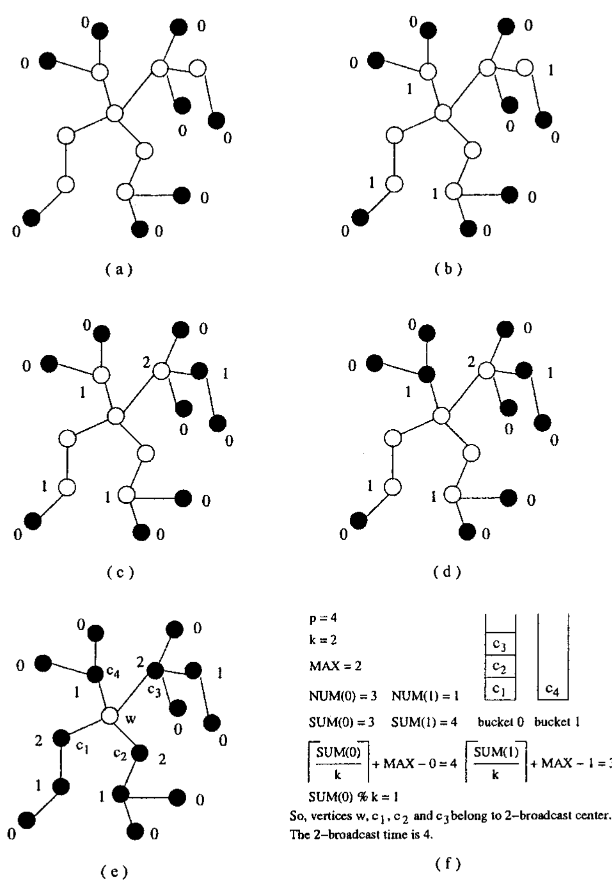

Figure 18: The performance of algorithm KBC for 2-broadcast center

KBC creates $\lceil \frac{p}{k} \rceil$ buckets, which are numbered from 0 to $\lceil \frac{p}{k} \rceil - 1$. Let MAX = max $\{ c_i.label \}$, for $1 \leq i \leq p$. For each labeled neighbor c of w, if $c.label = MAX - i$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$, then put c into bucket i. Let the number of elements in the ith bucket be $NUM(i)$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$. The number of elements in the buckets 0, 1, 2, ..., $i-1$ and i is denoted by $SUM(i)$. Thus, $SUM(i) = \sum_{j=0}^{i} NUM(j)$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$. Then $w.label = max\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\}$, for $0 \leq i < \lceil \frac{p}{k} \rceil$. Let x be the minimal integer between 0 and $\lceil \frac{p}{k} \rceil - 1$, such that $\lceil \frac{SUM(x)}{k} \rceil + MAX - x = max\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\}$, for $0 \leq i < \lceil \frac{p}{k} \rceil$. If $SUM(x)\ mod\ k = 1$, then vertex w and all vertices in buckets $0, 1, \cdots, x$ belong to $BC_k(T)$. Otherwise when $SUM(x)\ mod\ k \neq 1$, only vertex w belongs to $BC_k(T)$. The label of w is the k-broadcast time of w in tree T, and by Theorem 4, we can then calculate the k-broadcast time of each vertex in T.

Algorithm KBC

Input: tree T, k $(k \geq 2)$.

Output: $BC_k(T)$ and $b_k(T)$.

1. $T' \leftarrow T$;

2. For each leaf c of T

   2.1. $c.label = 0$;

   2.2. $T' \leftarrow T' - c$;

3. For each leaf u of T', $u.label = max\{c_{(i-1)k+1}.label + i\}$, for $1 \leq i \leq \lceil \frac{p}{k} \rceil$, where c_1, c_2, $\cdots$, c_p stand for the p labeled vertices adjacent to u and c_1, c_2, $\cdots$, c_p are ordered such that $c_1.label \geq c_2.label \geq \cdots \geq c_p.label$; // Calculated by Procedure OrderWeight

4. While($|V(T')| \geq 2$) ($V(T')$ stands for the set of vertices in T')

   4.1. Given $v \in V(T')$ such that $v.label = min\{u.label | u \in V(T')\}$;

   4.2. $T' \leftarrow T' - v$;

   4.3. Let s be the vertex adjacent to v in T'. If s is now a leaf of T', then $s.label = max\{c_{(i-1)k+1}.label + i\}$, for $1 \leq i \leq \lceil \frac{p}{k} \rceil$, where c_1, c_2, $\cdots$, c_p

stand for the p labeled vertices adjacent to s, and $c_1, c_2, \cdots, c_p$ are ordered such that $c_1.label \geq c_2.label \geq \cdots \geq c_p.label$; // Calculated by Procedure OrderWeight

5 Let w be the only vertex in T', and $c_1, c_2, \cdots, c_p$ are its p labeled neighbors, MAX = max { $c_i.label$ }, for $1 \leq i \leq p$;

6 Create $\lceil \frac{p}{k} \rceil$ buckets, which are numbered from 0 to $\lceil \frac{p}{k} \rceil - 1$;

7 For each labeled neighbor c of w, if $c.label = MAX - i$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$, then put c into bucket i;

8 Let the number of elements in the ith bucket be $NUM(i)$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$. The number of elements in the first ith buckets is denoted by $SUM(i)$. $SUM(i) = \sum_{j=0}^{i} NUM(j)$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$;

9 $w.label = max\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\}$, for $0 \leq i < \lceil \frac{p}{k} \rceil$;

10 Let x be the minimal integer between 0 and $\lceil \frac{p}{k} \rceil - 1$, such that $\lceil \frac{SUM(x)}{k} \rceil + MAX - x = max\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\}$, for $0 \leq i < \lceil \frac{p}{k} \rceil$. If $SUM(x)$ *mod* $k = 1$, then vertex w and all vertices in the buckets $0, 1, \cdots, x$ belong to $BC_k(T)$. Otherwise, only vertex w belongs to $BC_k(T)$.

Figure 18 illustrates the performance of KBC algorithm in determining the 2-broadcast center of a given tree. In Figure 18 (a), all vertices with black backgrounds are leaves of tree T, while all vertices with white backgrounds belong to T'. In Figure 18 (b), all leaves of T' are labeled with 0. Then, in Figure 18 (c), one of the leaves of T' with minimal label is removed from T', and its neighbor in T' is labeled. In Figure 18 (d), again, one of the leaves of T' with minimal label is removed from T'. However, its neighbor in T' is not labeled since this neighbor is not a leaf of T'. This process continues until T' contains only one vertex. Let this single vertex in T' be w and its four labeled neighbors be c_1, c_2, c_3 and c_4, where $c_1.label \geq c_2.label \geq c_3.label \geq c_4.label$. Then, MAX $= c_1.label = 2$. Let $p = 4$ stand for the number of labeled neighbors of w. Then, the algorithm KBC creates $\lceil \frac{p}{k} \rceil = 2$ buckets, and puts c_1, c_2 and c_3 in bucket 0 and c_4 in bucket 1. Since $\lceil \frac{SUM(0)}{k} \rceil + MAX - 0 = max\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\} = 4$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$ and $SUM(0)$ mod $2 = 1$, then vertices w, c_1, c_2 and c_3 belong to $BC_2(T)$ and $b_2(T) = 4$.

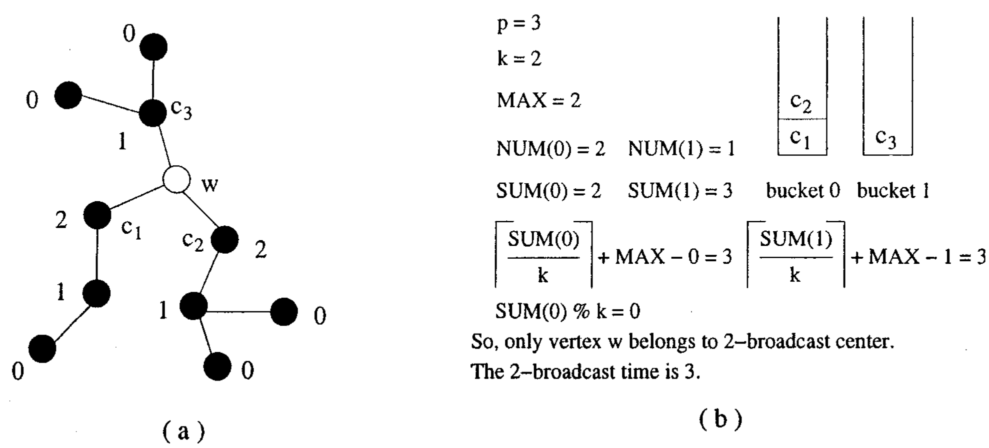

Figure 19: A case with only one vertex in 2-broadcast center

Figure 19 illustrates an example where only one vertex belongs to 2-broadcast center. After applying KBC on the tree in Figure 19 (a), w is the last vertex in T'. Vertex w has three labeled neighbors c_1, c_2 and c_3, where $c_1.label \geq c_2.label \geq$

$c_3.label$. Then, the algorithm creates $\lceil \frac{p}{k} \rceil = 2$ buckets, and puts c_1 and c_2 in bucket 0 and c_3 in bucket 1. Since $\lceil \frac{SUM(0)}{k} \rceil + MAX - 0 = max\{\lceil \frac{SUM(i)}{k} \rceil + MAX - i\} = 3$ $(0 \leq i < \lceil \frac{p}{k} \rceil)$ and $SUM(0)\ mod\ 2 = 0$, then only vertex w belongs to $BC_2(T)$ and $b_2(T) = 3$.

The following discussion justifies the validity of the KBC algorithm. If a vertex s is labeled in step 4.3., and its label is calculated based on the labels of vertices c_1, c_2, $\cdots$, c_p, then we say s is the parent of c_1, c_2, $\cdots$, c_p, and c_1, c_2, $\cdots$, c_p are the children of s. A child of vertex u is its descendant. Any child of the descendants of u is also a descendant of u. The descendant tree of u, which is denoted by T_u, is the tree rooted at u and composed by u and all its descendants. Clearly, $u.label = b_k(u, T_u)$.

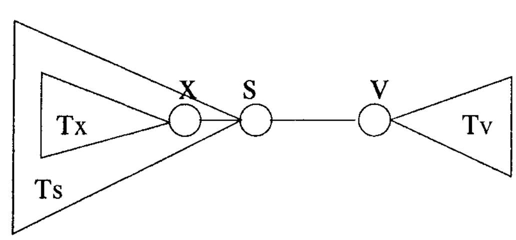

Figure 20: The illustration of the proof of lemma 6

Lemma 6. *The last vertex w in T' is in $BC_k(T)$.*

Proof. First, we need to prove that after removing a vertex from T' in step 4.3., T' still includes some vertices in $BC_k(T)$. Let v be the removed vertex in step 4.3. and s be the vertex adjacent to v in T', it suffices to show that $b_k(v, T) \geq b_k(s, T)$. If $V(T') = \{s, v\}$ before step 4.3., then by the algorithm $v.label \leq s.label$. So, $b_k(v, T_v) \leq b_k(s, T_s)$, $b_k(v, T) \geq 1 + b_k(s, T_s)$ and $b_k(s, T) \leq 1 + b_k(s, T_s)$. Therefore,

$b_k(v,T) \geq b_k(s,T)$. If there is another leaf x in T' before step 4.3., then by the algorithm $x.label \geq v.label$. So, $b_k(x,T_x) \geq b_k(v,T_v)$. Let the tree $T - T_v$ be T_s and T_s is rooted at s (see Figure 20). Since T_x is a subtree of T_s, $b_k(v,T_v) \leq b_k(x,T_x) \leq b_k(s,T_s)$. So, $b_k(v,T) \geq 1+b_k(s,T_s)$ and $b_k(s,T) \leq 1+b_k(s,T_s)$. Therefore, $b_k(v,T) \geq b_k(s,T)$. Thus, for any vertex x removed from T' in KBC, $b_k(x,T) \geq b_k(w,T)$, and $w \in BC_k(T)$. □

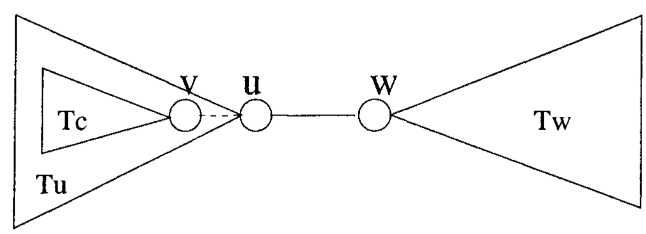

Figure 21: The illustration of the proof of lemma 7

Lemma 7. *Let w be the last vertex in T', then a vertex in $BC_k(T)$ could only be either w or a vertex adjacent to w.*

Proof. Let vertex u be a neighbor of vertex w and vertex v be a descendant of u (see Figure 21). By the algorithm, $b_k(w,T_w) \geq b_k(u,T_u) > b_k(v,T_v)$. So, $b_k(v,T) \geq b(w,T_w) + dist(v,w) \geq b(w,T_w + 2 > b(w,T_w) + 1 \geq b_k(w,T)$. So, if $dist(v,w) \geq 2$, then $v \notin BC_k$. This statement and Theorem 3 conclude that only w or its neighbors could belong to $BC_k(T)$. □

The last question is which neighbor of w is in $BC_k(T)$? Figure 22 shows vertex u and its p labeled neighbors where $c_1.label \geq c_2.label \geq \cdots \geq c_p.label$. Thus,

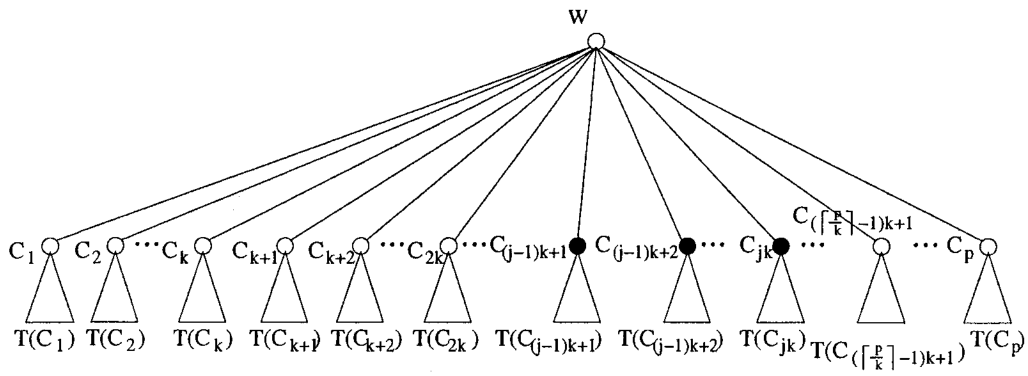

Figure 22: Vertex w and its labeled neighbors

$b_k(w,T) = w.label = max\{c_{(i-1)k+1}.label + i\}$, for $1 \leq i \leq \lceil \frac{p}{k} \rceil$. For any labeled neighbors c_q $(1 \leq q \leq p)$ of w, if $b_k(c_q,T) = b_k(w,T)$, then $c_q \in BC_k(T)$. $T - T_{c_q}$ stands for the subtree of T obtained by removing T_{c_q} from T.

Lemma 8. *Let w be the last vertex in T' and c_q $(1 \leq q \leq p)$ is a neighbor of w, $c_q \in BC_k(T)$ if and only if $b_k(w,T) > b_k(w,T - T_{c_q})$.*

Proof. By KBC, $b_k(c_q,T_{c_q}) \leq b_k(w,T - T_{c_q})$, then $b_k(c_q,T) = 1 + b_k(w,T - T_{c_q}) > b_k(w,T - T_{c_q})$. Thus, $b_k(w,T) > b_k(w,T - T_{c_q}) \Rightarrow b_k(w,T) \geq b_k(w,T - T_{c_q}) + 1 \Rightarrow b_k(w,T) \geq b_k(c_q,T)$. Since $w \in BC_k(T)$, $b_k(w,T) \leq b_k(c_q,T)$. Thus, $b_k(w,T) = b_k(c_q,T) \Rightarrow c_q \in BC_k(T)$. On the other hand, $c_q \in BC_k(T) \Rightarrow b_k(w,T) = b_k(c_q,T) = 1 + b_k(w,T - T_{c_q}) > b_k(w,T - T_{c_q})$. □

Let j be the minimal integer between 0 and $\lceil \frac{p}{k} \rceil - 1$, such that $c_{(j-1)k+1}.label + j = max\{c_{(i-1)k+1}.label + i\}$, for $1 \leq i \leq \lceil \frac{p}{k} \rceil$. The k vertices $c_{(j-1)k+1}$, $c_{(j-1)k+2}$, $\cdots$, c_{jk} have black backgrounds in Figure 22. For any $x > (j-1)k + 1$, $b_k(w,T - T_{c_x}) =$

$c_{(j-1)k+1}.label + j = b_k(w,T)$. So, all vertices c_x where $x > (j-1)k+1$ are not in $BC_k(T)$. Now let us examine the vertices c_x where $x \leq (j-1)k+1$. If $c_{(j-1)k+1}.label = c_{(j-1)k+2}.label$, then $b_k(w,T-T_{c_x}) = c_{(j-1)k+2}.label+j = c_{(j-1)k+1}.label+j = b_k(w,T)$. If $c_{(j-1)k+1}.label > c_{(j-1)k+2}.label$, then for any vertex c_x where $x \leq (j-1)k+1$, $b_k(w,T-T_{c_x}) = max\{c_{(j-1)k+2}.label+j, c_{(j-2)k+2}.label+j-1,\} < c_{(j-1)k+1}.label+j$ $= b_k(w,T)$. By Lemma 8, $c_x \in BC_k(T)$. Therefore, if $c_{(j-1)k+1}.label > c_{(j-1)k+2}.label$, vertex $c_x \in BC_k(T)$ where $x \leq (j-1)k+1$. In step 10 of algorithm KBC, when $SUM(x)\ mod\ k = 1$, vertices $c_1, c_2, \cdots, c_{(j-1)k+1}$ are in buckets $0, 1, \cdots, x$ and vertex $c_{(j-1)k+1}$ is in bucket $x+1$. Thus, $c_{(j-1)k+1}.label > c_{(j-1)k+2}.label$. Therefore, all vertices in buckets $0, 1, \cdots, x$ belong to $BC_k(T)$. This leads us to the following theorem:

Theorem 5. *Given a tree T and k, the algorithm KBC generates $BC_k(T)$.*

By Theorem 4 and Theorem 5, it is easy to calculate $b_k(u,T)$ for any given vertex u in a tree T.

Assigning labels to vertices dominates the time complexity of algorithm KBC. Given a vertex u and its unsorted p labeled neighbors, the $OrderWeight$ procedure calculates $u.label$ in $O(p)$. Let the degree of the ith vertex in $T = (V,E)$ be d_i, for $1 \leq i \leq |V|$. Then, the time needed to calculate the labels of all the vertices is $\sum_{i=1}^{|V|} O(d_i)$. Since $\sum_{i=1}^{|V|} d_i = 2|E|$, the complexity of calculating labels is $O(|E|) = O(|V|)$. Therefore, the time complexity of KBC algorithm is $O(|V|)$.

Chapter 3

An Efficient Heuristic for k-Broadcasting in Networks

The problem of finding an optimal broadcast scheme or determining the broadcast time of an arbitrary network is NP-complete [76]. This chapter introduces a new heuristic for k-broadcasting, which is called Tree Based Algorithm (TBA). This heuristic generates optimal time k-broadcast schemes in rings, trees and in grid graphs when the originator is a corner vertex. In a two-dimensional torus graph T_2, it gives an upper bound $b_1(T_2) + 3$ for $k = 1$ and $b_2(T_2) + 1$ for $k = 2$. When $k \geq 3$, it also generates optimal k-broadcast time in the torus graph. Presently, the heuristic from [3] is the best heuristic for 1-broadcasting in practice. With a few exceptions, TBA generates the same 1-broadcast scheme as the heuristic from [3] on several commonly used interconnection networks, such as the *de Bruijn*, the *Shuffle Exchange*, the *Butterfly* graphs and the *Cube-Connected Cycle.* However, TBA outperforms the heuristic in [3] on three graph models from a network simulator ns-2. In a given graph $G = (V, E)$, the time complexity of one round of TBA is $O(|E|)$, while the complexity of one round without the process of matching of the algorithm from [3] is $O(|V|^2 \cdot |E|)$.

3.1 Previous Heuristics

Most of the previous heuristics for k-broadcasting have been for $k = 1$ [3], [16], [22], [25], [54], [72], [75]. Some of these algorithms give theoretical upper bounds on the 1-broadcast time. Given $G = (V, E)$ and the originator u, the heuristic in [54] returns 1-broadcast protocol whose performance is at most $b_1(u, G) + O(\sqrt{|V|})$ rounds. Theoretically, the best upper bound is presented in [16]. The approximation algorithm in [16] generates a broadcast protocol with $O(\frac{log(|V|)}{loglog(|V|)})b_1(G)$ rounds. This thesis concentrates on heuristics that perform well in practice. The heuristic in [3], which is called the Round_Heuristic, is the best existing heuristic for 1-broadcasting in practice. In fact, the Round_Heuristic is designed for the gossip problem, where each vertex has a message that it needs to send to all other vertices. Each gossip scheme also provides a 1-broadcast scheme for each vertex in a graph. Thus, the 1-broadcasting scheme is only a by-product of the Round_Heuristic.

In each round of 1-broadcasting, the Round_Heuristic gives a weight to each edge in the given graph. Then, a *maximum weighted matching* is calculated based on the weights of the edges. The matched edges are active in the current round, which

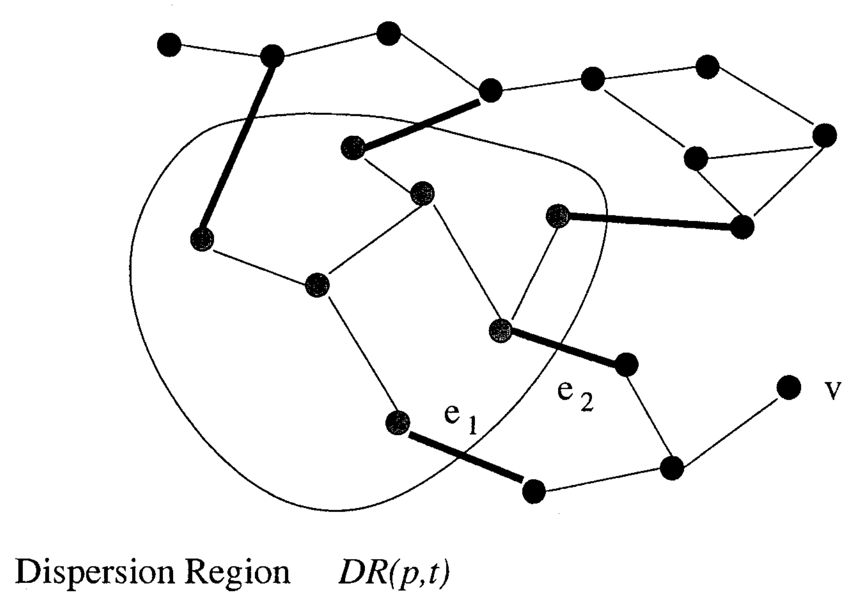

Figure 23: The definitions in Round-Heuristic

means that messages are passed through these matched edges in this round.

Two approaches are used in the Round_Heuristic to set the weights. One is the *Potential Approach*, wherein the weight of an edge (v, w) is set to equal its *potential*, defined as the number of messages known by either v or w, but not by both of them. In broadcasting, the weight could only be 0 or 1. Although this approach is simple, requires little storage and is very fast, its performance is worse than the second approach: the Breadth-First-Search (BFS) approach.

Several definitions are needed to introduce the BFS approach. The dispersion region $DR(p, t)$ of a message p refers to the set of vertices that know p at the beginning of round t (this is a connected subgraph). For a vertex v, $dist_v(p, t)$ denotes the shortest distance in the graph from v to a vertex $w \in DR(p,t)$. The set of border-crossing edges $bce(p,t)$ is defined as $bce(p,t) = \{(v,w) \in E \mid v \in DR(p,t) \textit{ and } w \notin DR(p,t)\}$. For a vertex $v \notin DR(p,t)$, $bce_v(p,t)$ consists of all edges in $bce(p,t)$ that lie on a shortest path from $DR(p,t)$ to v. Figure 23 [3] illustrates these definitions. In this figure, the edges of $bce(p,t)$ are drawn in bold. $dist_v(p,t) = 3$ and $bce_v(p,t) = \{e_1, e_2\}$. When considering an edge $e \in bce(p,t)$, how useful is e for the rapid dissemination of p? Message p should be routed on the shortest paths from $DR(p,t)$ to all other vertices. The more shortest paths that e lies on, the likelier it is that the dissemination of the message would be faster. Also, the larger $dist_v(p,t)$ is, the more priority should be given to forwarding p towards v. These criteria motivate the use of two parameters $Dist\_Exp$ and Num_Exp in calculating the weight. In round t, every vertex $v \notin DR(p,t)$ attributes the weight to every edge $e \in bce_v(p,t)$ as follows:

$$weight(v,p,t) = \frac{dist_v(p,t)^{Dist\_Exp}}{\mid bce_v(p,t) \mid^{Num\_Exp}} \tag{8}$$

In [3], a modified BFS algorithm is used to calculate the weight. Because it is a BFS algorithm, the vertices are considered in an order of increasing $dist_v(p,t)$. For vertices v with $dist_v(p,t) = 1$, $bce_v(p,t)$ consists of all adjacent edges that connect v to a vertex in $DR(p,t)$. For larger $dist_v(p,t)$, the algorithm computes the *union* of the sets $bce_{w_i}(p,t)$, for all vertices w_j adjacent to v with $dist_{w_i}(p,t) = dist_{w_i}(p,t) - 1$. Thus, the calculation of $bce_v(p,t)$ can easily be incorporated into the BFS search. For a vertex v, $bce_v(p,t)$ is the union of a maximum of $|V|$ sets with a maximum of $|E|$ elements each. This computation takes $O(|V| \cdot |E|)$ time. The $bce_v(p,t)$ is computed for all uninformed vertices. Consequently, calculating the weights takes $O(|V|^2 \cdot |E|)$

in total. Calculating a maximum weighted matching is viewed as an external routine in [3]. Therefore, no specific algorithm for matching is introduced. The total time complexity without matching of the Round_Heuristic is $O(R \cdot |V|^2 \cdot |E|)$, in which R represents the number of rounds of 1-broadcasting.

There are no theoretical bounds on the performance of Round_Heuristic. However, most of the test results presented in [3] for the CCC_k graph, the *Shuffle Exchange* graph, the *Butterfly* graph and the *DeBruijn* graphs are equal to the optimal 1-broadcast times. The performance of Round_Heuristic heavily depends on the choice of the values of the two parameters. *Dist_Exp* is the parameter of particular importance. It determines the influence of the distance between vertices and dispersion regions. The values in the range from 0.25 to 60 are used.

3.2 The Tree Based Algorithm (TBA)

This section presents the new heuristic for k-broadcasting in arbitrary graphs. The heuristic always generates the optimal k-broadcast time in an arbitrary tree originated at any vertex. Thus, it is called the Tree Based Algorithm (TBA).

3.2.1 TBA and its Complexity

In order to formally present TBA, we first give several definitions.

Definition 1. *bright border: The bright border $bb(t)$ is composed of those informed vertices that have uninformed neighbors at round t.*

Let $D(v, t)$ stand for the shortest distance from uninformed vertex v to $bb(t)$ at round t.

Definition 2. *child and parent: Given an uninformed vertex u and its uninformed neighbor v, if $D(u, t) = D(v, t) + 1$, then u is a child of v, and v is the parent of u.*

Definition 3. *descendant: a child of vertex u is its descendant. Any of the children of a descendant of u is also a descendant of u.*

Fig. 24 illustrates these definitions. In this example, vertex a is the originator. After three rounds, vertices in the shadowed area are still uninformed. The informed vertices with shadowed backgrounds belong to $bb(4)$. The distance between $bb(4)$ and the uninformed vertices with black backgrounds is one. The distance between $bb(4)$ and the uninformed vertices n, o and p is two. The distance between $bb(4)$ and the uninformed vertex q is three. So, vertices o and p are children of vertex j, and vertex q is a child of vertices o and p. We can also say that vertices o, p and q are descendants of vertex j.

The basic idea of TBA is to find a matching between the set of informed and uninformed vertices in each round, and then distribute the message between them. To achieve this, in each round, we first perform a modified BFS (breadth first search) from $bb(t)$ towards uninformed vertices. During this process, we label any uninformed vertex v with $D(v, t)$. Thus, the parents and children relationship among the uninformed vertices can be defined by these distances.

The weight of a vertex in TBA is based on the strategy of the optimal k-broadcasting

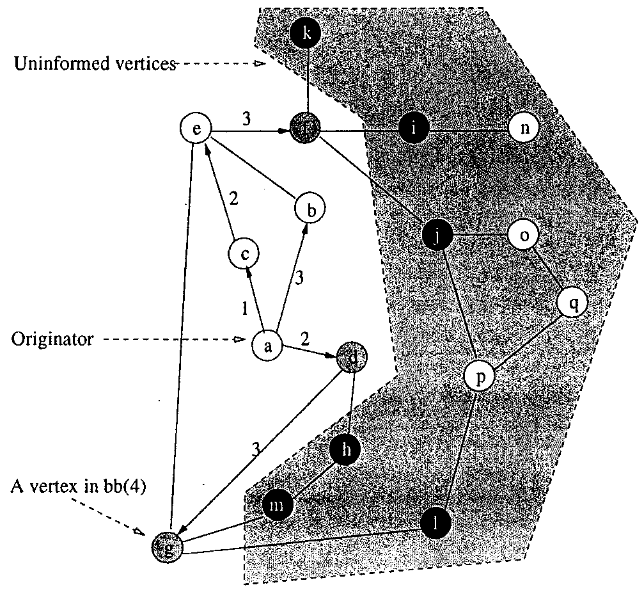

Figure 24: Definitions in TBA

in trees. Let $w(u,t)$ stand for the weight of vertex u in round t. If u has no children, then $w(u,t) = 0$. If u has p children v_1, v_2, ..., v_p, and $w(v_1,t) \geq w(v_2,t) \geq \cdots \geq w(v_p,t)$, then $w(u,t) = max\{w(c_{(i-1)k+1}, t) + i\}$ $(1 \leq i \leq \lceil \frac{p}{k} \rceil)$. After that, we find a matching between the set of informed and uninformed vertices, wherein each informed vertex has up to k uninformed mates. We use a heuristic with time complexity $O(|E|)$ to find the matching. The heuristic tries to bring the number of pairs of vertices to a maximum; given this, it tries to maximize the weights of matched vertices. Finally, every matched informed vertex sends the message to its mates.

In TBA, procedures *Calculate_weight* and *Calculate_match* determine weights of all uninformed vertices and the matching in each round respectively. Given the weights of all k children of a vertex v, procedure *Weight* returns the weight of v

in time $O(k)$. This procedure is similar to the procedure $OrderWeight$ in Chapter 2 except that it is intended for both integers and fractional numbers. In the refinement of TBA, the weights could be fractional numbers. The procedure *Calculate_weight* starts with assigning weights to the vertices that have no children. Then it assigns weights to all uninformed vertices recursively by calling the procedure *Weight*. Procedure *Weight* takes $O(d)$ time to calculate the weight of a vertex with degree d. Thus, the time needed to calculate the weights of all the vertices is $\sum_{i=1}^{n} O(d_i)$. Since $\sum_{i=1}^{n} d_i = 2|E|$, the time complexity of the procedure *Calculate_weight* is $O(|E|)$. The procedure *Calculate_match* approximately computes a maximum weighted matching. All the vertices in $bb(t)$ are saved in a group of linked lists. The operations used in the procedure *Calculate_match* are similar to *build*(), *deletemin*() and *decreasekey*() in a priority queue. Generally, these operations take $O(|V|log|V| + |E|)$ time. However, the priorities are bounded by the maximum degree. We use a linked list for each priority class, where each class has the same number of uninformed vertices. This reduces the time complexity to $O(|V| + |E|) = O(|E|)$. Therefore, in each round, the time complexity of TBA is $O(|E|)$ in total. The pseudocode of the heuristic is presented below. The refinement of TBA is also mentioned in the procedure *Calculate_weight* and *Weight*. The refinement will be introduced later in detail.

Heuristic *TBA (Tree Based Algorithm)*

Input: graph $G = (V, E)$ and originator u.

Output: broadcast scheme and $b(u, G)$.

1. round = 0; /* set broadcast time 0 */

2. *bb*(*round*) $\leftarrow$ all informed vertices with uninformed neighbors; /* in round 0, only the originator is on the bright border */

3. *Remote* $\leftarrow \emptyset$; /* stack used to calculate the weight of each uninformed vertex */

4. *Uninformed* $\leftarrow |V|$ - 1; /* number of uninformed vertices */

5. **While** Uninformed != 0

   5.1. round ++;

   5.2. Perform variant BFS from *bb*(*round*) to uninformed vertices, and mark any uninformed vertex v with $D(v, round)$;

   5.3. During the process of variant BFS, push vertices in *bb*(*round*) and all uninformed vertices into stack *Remote*;

   5.4. Procedure Calculate_weight;

   5.5. Procedure Calculate_match;

   5.6. *bb*(*round*) $\leftarrow \emptyset$;

   5.7. *bb*(*round*) $\leftarrow$ all informed vertices with uninformed neighbors;

Procedure *Calculate_weight*

1. **While** *Remote* is not empty

   1.1. v = *Remote*.pop();

   1.2. **if** $v.childrenset = \emptyset$;

1.3. **then** $v.weight = 0$;

1.3. /* Refinement version */ ***then*** $v.weight = 1$;

1.4. **else** $v.weight$ = Procedure Weight($v.childrenset$);

1.5. **For** all uninformed neighbors w of v;

1.5.1 **if** $D(w, round) = D(v, round) - 1$;

1.5.2 **then** w.childrenset $\leftarrow v$;

Procedure *Weight*

Input: *v.childrenset*

Output: the weight of vertex v

0. /* Only in refinement version */ ***for*** $w \in v.childrenset$ $w.weight = \frac{(w.weight) \cdot p}{q}$, *where* q *is the number of parents of* w *and* p *is a parameter. /* for each vertex* w*, this calculation only be performed once in each round although* w *could have more than one parent.*/*

1. Create $\lceil \frac{p}{k} \rceil$ *Bucket*, where $p = | v.childrenset |$ */

2. $MAX(v) = max\{w.weight \mid w \in v.childrenset\}$;

3. **for** $w \in v.childrenset$

   3.1. **if** $MAX(v) - i \geq w.weight > MAX(v) - i - 1, 0 \leq i < \lceil \frac{p}{k} \rceil$;

   3.2. **then** $Bucket[i] \leftarrow w$;

4. **for** $0 \leq i < \lceil \frac{p}{k} \rceil$

4.1. $SUM(i) = \sum_{j=0}^{i} \mid Bucket(i) \mid$;

4.2. $MIN(i) = min\{w.weight \mid w \in Bucket(i)\}$;

5. **return** $max\{\lceil \frac{SUM(i)}{k} \rceil + MIN(i) \mid 0 \leq i < \lceil \frac{p}{k} \rceil\}$;

Procedure *Calculate_match*

1. list *match*[*degree*]; /* create *degree* lists. *degree* stands for the maximum degree of all vertices in $G = (V, E)$ */

2. **for** all vertices w in $bb(round)$

   2.1. *neighbor* = the number of uninformed neighbors of w;

   2.2. *match*[*neighbor*-1].add(w);

3. **for** $0 \leq i \leq degree - 1$

   3.1. *match*[i].setcurr();/* set the current pointer in each list point to the first element */

4. **While** not all *current* points in lists of *match*[*degree*] are NULL; and let the first list where *current* is not NULL be *match*[i]

   4.1. w = *match*[i].getnext())!= NULL /* get the current element, and assign it to w; *current* points to the next element. */

   4.2. Let $x = k$

   4.3. **While** $x \neq 0$

4.3.1. v = one of the uninformed neighbors of w with maximum weights;

4.3.2. Output the broadcast scheme: w sends the message to vertex v in the current round;

4.3.3. mark v informed and *Uninformed* = *Uninformed* - 1 ;

4.3.4. **for** all neighbors p of vertex v such that p belongs to a list *match*[j]

4.3.4.1 **if** j=0 **then** remove p from *match*[j];

4.3.4.2 **if** $j > 0$ **then** move p from *match*[j] to *match*[j-1]; /* if p was located before the current pointer in *match*[j], then p is also located before the current pointer in *match*[j-1]; and if p was located behind the current pointer in *match*[j], then p is also located behind the current pointer in *match*[j-1] */

4.3.5. $x = x - 1$;

Refinement

In TBA, a vertex could be a descendant of multiple vertices. Thus, the effect of this vertex on the process of broadcasting is overestimated. Figure 25 shows such an example. The graph in Figure 25(a) is the original graph. Vertex a is the originator. The graph in Figure 25(c) illustrates the 1-broadcast scheme generated by TBA. The weights of each uninformed vertex in round 2 are presented in Figure 25(b). The vertices with shadowed backgrounds are informed vertices. In the second round, the weights of vertices f and c are equal to 1 because vertex g is a child of both f and c. However, vertex g receives the message either from f or from c, but not from both.

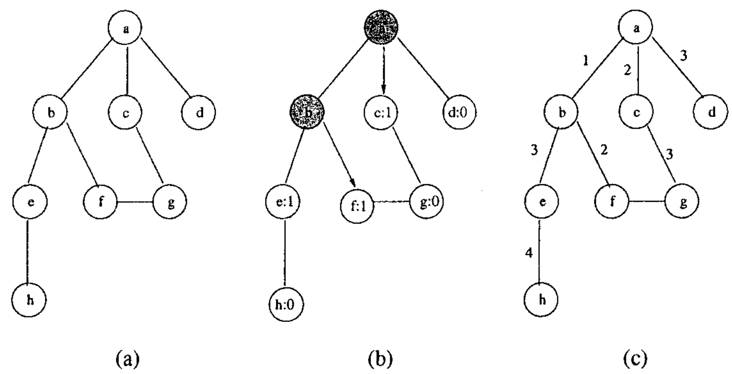

Figure 25: The performance of TBA

Therefore the effect of vertex g is overestimated. In this example, vertex e and vertex f have the same weight. As a result vertex b could send the message to vertex f in the second round, although sending to vertex e would be a better choice. This motivates the following refinement: the weight of a child is divided by the number of its parents. In k-broadcasting, if a vertex u has no children, then $w(u,t) = 1$. If u has x children v_1, v_2, ..., v_x, where $w(v_1,t) \geq w(v_2,t) \geq \cdots \geq w(v_x,t)$, then $w(u,t) = max\{\frac{w(c_{(i-1)k+i},t)\cdot p}{q} + i, 1 \leq i \leq \lceil\frac{x}{k}\rceil\}$. Here q stands for the number of parents of c_i, and p stands for a parameter. For the parameter p, we used integers from 1 to 6. Note that the time complexity of the refinement is the same as that of the original heuristic.

The graph in Figure 26(a) presents the weights of each uninformed vertex in round 2 by using the refinement. The graph in Figure 26(b) shows the broadcast scheme generated by the refinement. This is the optimal broadcast scheme from originator a.

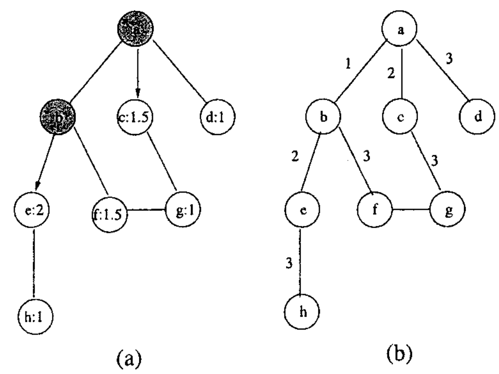

Figure 26: The performance of the refinement

3.2.2 Theoretical Results

It is easy to see that TBA generates the optimal k-broadcast scheme on several simple topologies, such as *cycle* and *tree*. An $m \times n$ grid graph $G_{m,n}$ is the Cartesian product of path graphs on m and n vertices, while the $m \times n$ torus graph $Torus(m, n)$ is the Cartesian product of cycle graphs on m and n vertices. In grid and torus graphs, the vertical paths or cycles are columns, and the horizontal paths or cycles are rows. The columns are numbered from 0 to $n-1$, while the rows are numbered from 0 to $m-1$. A vertex on the intersection of row i and column j is denoted by (i, j). This section will prove that TBA generates an optimal 1-broadcast scheme on the *grid* graph when the originator is a corner vertex. More importantly, this section will prove that a 1-broadcast scheme in a grid graph is an optimal 1-broadcast scheme if the originator is a corner vertex and a vertex is not idle unless this vertex has no uninformed neighbors. In torus graphs, the upper bound of the 1-broadcast time generated by TBA is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2 \leq b_1(Torus(m, n)) + 3$, while the upper bound of the 2-broadcast time generated by TBA is $\lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + 1 = b_2(Torus(m,n)) + 1$. When $k \geq 3$, TBA generates an optimal time k-broadcast scheme in $Torus(m,n)$. In this section, $b_k(A(u,G))$ stands for the k-broadcast time of graph G generated by TBA when the originator is vertex u, and $b_k(A(G))$ stands for the k-broadcast time of graph G generated by TBA.

The Grid Graph

The following results are presented in [19]: let v be a vertex in $G_{m,n}$, then $b_1(v, G_{m,n}) = m + n - 2$ when v is a corner vertex. When v is a side vertex, then $b_1(v, G_{m,n}) =$ the maximum distance from v to a corner vertex plus 1 if there are two corner vertices at the maximum distance, and $b_1(v, G_{m,n}) =$ the maximum distance from v to a corner vertex if there is one corner vertex at the maximum distance. If v is an interior vertex at position (i, j), then $b_1(v, G_{m,n}) =$ the maximum distance from v to a corner vertex plus 1 if $i = \frac{m-1}{2}$ or $j = \frac{n-1}{2}$, plus 2 if $i = \frac{m-1}{2}$ and $j = \frac{n-1}{2}$, and $b_1(v, G_{m,n}) =$ the maximum distance from v to a corner vertex otherwise.

Given the originator u and a 1-broadcast scheme S in a graph G, $b_1(S(u,G))$ stands for the 1-broadcast time of u in graph G by using the 1-broadcast scheme S. This section will prove that for a 1-broadcast scheme S where a vertex is not idle unless it has no uninformed neighbors, $b_1(S((0,0), G_{m,n})) = b_1((0,0), G_{m,n}) = m + n - 2$.

To present the theorem, we need the following definitions:

(1) *border*: A path of the minimal length which contains all the informed vertices that have uninformed neighbors, and all the vertices on this path are informed vertices.

(2) *outside neighbors* of vertex (i, j): $(i + 1, j)$ and $(i, j + 1)$.

(3) *convex border*: A border is convex if there are no two vertices (i, j) and (p, q) on the border such that $i > p$ and $j > q$.

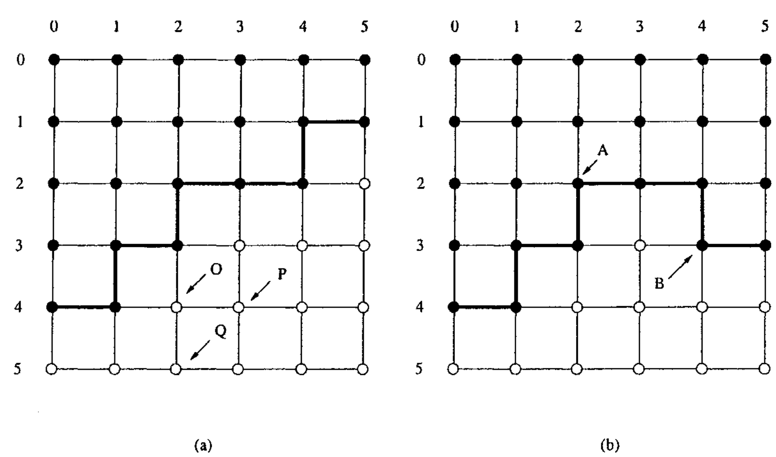

Figure 27: Definitions in the grid graph.

Fig. 27 illustrates the above definitions. The vertices with black backgrounds are informed vertices, and the vertices with white backgrounds are uninformed vertices. The vertices connected by bold edges compose the *border*. Vertices P and Q are the outside neighbors of vertex O. The border in Fig. 27(a) is convex, while the border in Fig. 27(b) is not convex, since vertices $A = (2, 2)$ and $B = (3, 4)$ are on the border and $2 < 3$, $2 < 4$. First, we will prove some auxiliary lemmas.

Lemma 9. *Given a vertex (i,j) on a convex border, any other vertex (p,q), where $0 \leq p \leq i$ and $0 \leq q \leq j$, is informed.*

Proof. Assume that there exist uninformed vertices (p,q), where $0 \leq p \leq i$ and $0 \leq q \leq j$. Consider any such vertex (x,y) that has the shortest distance from $(0,0)$. This means that both $(x-1,y)$ and $(x,y-1)$ are informed. Then both $(x,y-1)$ and $(x-1,y)$ are on the convex border. If $x < i$ and $y \leq j$, then this is a contradiction, since $x < i$ and $y-1 < j$. If $x \leq i$ and $y < j$, then this is also a contradiction, since $x-1 < i$ and $y < j$. □

Lemma 10. *The border is convex after each round of a 1-broadcast scheme originated at vertex $(0,0)$ if vertices are active in this scheme as long as they have uninformed neighbors.*

Proof. We will prove this lemma by induction on the number of rounds. At the beginning, $(0,0)$ is the only informed vertex. In the first round, $(0,0)$ informs either $(0,1)$ or $(1,0)$, which generates a convex border.

Assume that after round t, the border generated by any 1-broadcast scheme S is convex. We should prove that the border will be convex after round $t+1$. Assume that the border is not convex after round $t+1$. Then, there exist vertices (i,j) and (p,q) on the border, where $i < p$ and $j < q$. It is easy to see that (p,q) was not on the border after round t, since either (i,j) or one of $(i-1,j)$ and $(i,j-1)$ were on the convex border after round t. Thus, vertex (p,q) received the message at time $t+1$ either from $(p-1,q)$ or $(p,q-1)$. So, at least one of $(p-1,q)$ and $(p,q-1)$ was on the convex border after round t. Consider the case that $(p-1, q)$ was on the convex border after round t. By Lemma 9, after round t, vertex (i, j) was informed (since $i \leq p-1$ and $j < q$) and it had at most one uninformed neighbor, $(i+1, j)$. So, after round $t+1$, $(i+1, j)$ is informed, and (i, j) cannot be on the border since it has no uninformed neighbors. This contradicts the assumption of the lemma. Similarly, we can get a contradiction for the case that $(p, q-1)$ was on the convex border after round t. □

Theorem 6. $b_1(S((0,0), G_{m,n})) = m + n - 2$.

Proof. From Lemma 9, any vertex on a convex border can only send the message to its outside neighbors. From Lemma 9 and Lemma 10, the longest distance between the vertices on the border and the originator increases by 1 at each round. The vertex $(m-1, n-1)$ is informed in round $m+n-2$ since it is the only vertex in the grid that has distance $m+n-2$ from $(0,0)$. From Lemma 9, after vertex $(m-1, n-1)$ is informed, then the broadcasting is complete. Thus, $b_1(S((0,0), G_{m,n})) = m + n - 2$. □

When $k \geq 2$ and the originator is on a corner, an informed vertex has at most 2 uninformed neighbors in each round. Therefore, the k-broadcast time is the diameter of the grid graph, which is $m+n-2$. So, we have:

Theorem 7. $b_k(S((0,0), G_{m,n})) = m + n - 2$.

The above theorem states that the k-broadcast time of any k-broadcast scheme from originator $(0,0)$ is equal to the diameter of the grid. The k-broadcast time of

$G_{20,30}$		$G_{50,30}$		$G_{15,25}$		$G_{20,25}$	
O	R	O	R	O	R	O	R
0,0	48	0,0	78	0,0	38	0,0	43
3,2	43	9,6	63	3,5	30	3,2	38
5,3	40	12,7	59	6,8	24	5,8	30
9,5	34	12,14	52	7,10	22	8,4	31
10,7	32	15,20	54	7,12	21	10,12	23
11,9	31	15,25	59	9,15	24	15,10	29
10,15	25	20,25	54	11,16	27	15,16	31
15,10	34	25,15	40	12,20	32	18,20	38
15,20	35	30,18	48	12,22	34	18,24	42
19,28	47	45,28	73	14,22	36	12,24	36

Table 1: Test results of 1-broadcasting in $G_{m,n}$

TBA follows directly.

Corollary 1. $b_k(A((0,0), G_{m,n})) = m + n - 2$.

It is natural to state that $b_k(A((x,y), G_{m,n})) \leq m+n-2$ for any originator (x,y), since the worst case of k-broadcasting in grid graphs happens when the originator is on a corner. All the test results of TBA confirm the above statement (see Table 1). In this table, the originator is listed in the columns labeled O, and the 1-broadcast times are listed in the column labeled R. Moreover, TBA always generates the theoretical minimum 1-broadcast time (see [19]) from all originators. However I am unable to prove the above statement mathematically.

The Torus Graph

TBA generates an almost optimal time k-broadcast scheme for the torus. The optimal 1-broadcast time of $Torus(m,n)$ is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 1$ when both m and n are odd,

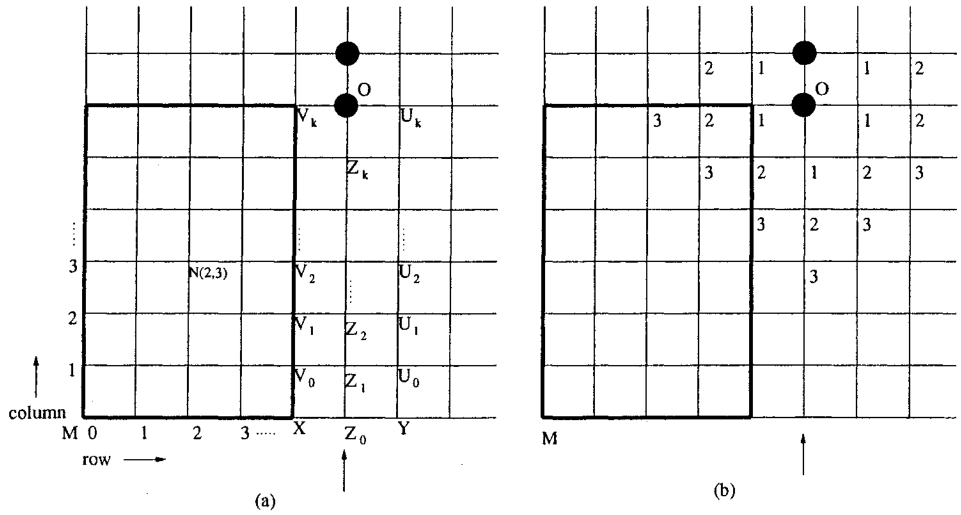

Figure 28: 1-broadcasting in $Torus(m,\ n)$

and is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil$ otherwise [24]. When $k \geq 2$, we can achieve the optimal k-broadcast time in $Torus(m, n)$ by using the following scheme: first, any informed vertex sends the message to its uninformed column neighbors (if it has such neighbors). After all vertices in the originator's column are informed, each informed vertex sends the message to its uninformed row neighbors (if it has such neighbors). This scheme clearly gives $b_k(Torus(m, n)) = \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor$, which is the diameter of $Torus(m, n)$. In this section, we will show that $b_1(A(Torus(m, n))) \leq \lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2 \leq b_1(Torus(m, n)) + 3$, $b_2(A(Torus(m, n))) \leq \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + 1 = b_2(Torus(m, n)) + 1$ and $b_k(A(Torus(m, n))) \leq \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + 1 = b_k(Torus(m, n))$ when $k \geq 3$.

Let us first look at the case that $k = 1$. Without loss of generality, assume that in round one of 1-broadcasting the originator sends the message to a neighbor on the same column. Fig. 28 illustrates $Torus(m, n)$ after the first round. Fig. 28(b) shows the distance between vertices in the torus and the originator vertex O. The vertices with solid background are informed vertices, and vertex M is the uninformed vertex that has the longest distance from vertex O. In 1-broadcasting, each vertex in the area defined by thick line has two children except vertices in row 0 or in column 0. The vertices in the column indicated by the arrow have three children except vertex Z_0 (which has two children). The two children of (i,j) are $(i-1,j)$ and $(j-1,i)$. Vertex $(0,j)$ has one child $(0,j-1)$ for $j>0$. Vertex $(i,0)$ has one child $(i-1,0)$ for $i>0$. Vertex $(0,0)$ does not have any children because it has the longest distance from vertex O. The weight of a vertex $N=(i,j)$ is denoted by $w(N)$ or $w(i,j)$.

Before proving the main theorem of 1-broadcasting, we first present some auxiliary lemmas.

Lemma 11. *In the area defined by thick lines, $w(i,j) = i + j + min\{i,j\}$.*

Proof. Lemma 11 can be proved by induction. The statement is correct for vertex $(0,0)$ since $w(0,0) = 0+0+min\{0,0\} = 0$. For all the vertices that are on row 0, assume that $w(0,j) = 0+j+min\{0,j\} = j$, then $w(0,j+1) = w(0,j)+1 = j+1 = 0+(j+1)+min\{0,j+1\}$. For the vertices that are on column 0, the proof is similar. Assume that the statement is correct for all descendants of (i,j) ($i \neq 0$ and $j \neq 0$). Vertex (i,j) has two children $(i-1,j)$ and $(i,j-1)$. If $i > j$, then $min\{i-1,j\} = j$ and $min\{i,j-1\} = j-1$. So, $w(i-1,j) = i-1+j+min\{i-1,j\}$ $= i+2j-1$ and $w(i,j-1) = i+j-1+min\{i,j-1\} = i+2j-2$. Then, $w(i,j) = w(i-1,j)+1 = i+2j = i+j+min\{i,j\}$. If $i < j$, then $min\{i-1,j\} = i-1$ and $min\{i,j-1\} = i$. So, $w(i,j-1) = 2i+j-1$ and $w(i-1,j) = 2i+j-2$. Then,

$w(i,j) = w(i,j-1)+1 = 2i+j = i+j+min\{i,j\}$. If $i = j$, then $w(i,j-1) = i+2j-2 = w(i-1,j)$. So, $w(i,j) = w(i-1,j)+2 = i+2j = i+j+min\{i,j\}$. □

Lemma 12. $w(Z_0) \geq w(V_0) = w(U_0)$.

Proof. Let $Z_0 = (0,p)$. By Lemma 11, $w(X) = p-1$. Similarly, $w(Y) = p-1$. Since X and Y are two children of Z_0, then $w(Z_0) = p+1$. By Lemma 11, $w(V_0) = w(p-1,1) = p+min\{p-1,1\}$. $w(Z_0)-w(V_0) = 1-min\{p-1,1\}$. When $p-1 \geq 1$, then $w(Z_0)-w(V_0) = 0$, and when $p-1 < 1$, then $w(Z_0)-w(V_0) > 0$. Therefore, $w(Z_0) \geq w(V_0)$. The proof of $w(V_0) = w(U_0)$ is simple. □

Lemma 13. $w(Z_i) > w(V_i) = w(U_i)$, *for* $i = 1,2,\cdots,k$.

Proof. TBA assigns the same weights to vertices V_i and U_i. So, $w(V_i) = w(U_i)$, for $i = 1,2,\cdots,k$.

By Lemma 12, $w(Z_0) \geq w(V_0) = w(U_0)$. Z_1 has three children: Z_0, V_0 and U_0. By the definition of the weight, $w(Z_1) \geq w(V_0)+3$. Let $V_0 = (p,q)$, then $w(Z_1) \geq w(V_0)+3 = p+q+min\{p,q\}+3$. $w(V_1) = w(p,q+1) = p+q+1+min\{p,q+1\}$. $w(Z_1)-w(V_1) \geq min\{p,q\}+2-min\{p,q+1\}$. When $p \leq q$, $w(Z_1)-w(V_1) \geq 2$. When $p > q$, $w(Z_1)-w(V_1) \geq 1$. So, $w(Z_1) > w(V_1) = w(U_1)$. Assuming $w(Z_i) > w(V_i) = w(U_i)$ and $V_i = (p,q)$, we have $w(Z_{i+1}) \geq w(V_i)+3 = p+q+min\{p,q\}+3$. $w(V_{i+1}) = w(p,q+1) = p+q+1+min\{p,q+1\}$. So, $w(Z_{i+1})-w(V_{i+1}) \geq min\{p,q\}+2-min\{p,q+1\} > 0$. Thus, $w(Z_{i+1}) > w(V_{i+1}) = w(U_{i+1})$. Therefore, $w(Z_i) > w(V_i) = w(U_i)$, for $1 \leq i \leq k$,. □

Theorem 8. $b_1(A(Torus(m,\ n))) \leq \lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2$.

Proof. By Lemma 13, $w(Z_i) > w(V_i) = w(U_i)$, for $1 \leq i \leq k$. Thus, vertex Z_i receives the message before vertices V_i and U_i for any $i = k, k-1, \cdots, 1$. Since Z_0 is the furthest vertex from the originator on the same column, then Z_1 receives the message at round $\lceil \frac{m}{2} \rceil - 1$. After this, it is possible that Z_1 sends to V_0 or U_0 first, because $w(Z_0) \geq w(V_0) = w(U_0)$. In the worst case, Z_0 first sends to V_0, then U_0, and finally to Z_0. This takes 3 rounds. After this, vertices Z_0, Z_1, $\cdots$, Z_k and all the other vertices on the same column are informed. It takes at most $\lceil \frac{n}{2} \rceil$ rounds more to finish the 1-broadcasting using horizontal edges. Thus, $b_1(A(Torus(m,n))) \leq \lceil \frac{m}{2} \rceil - 1 + 3 + \lceil \frac{n}{2} \rceil = \lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2$. □

Theorem 9. $b_2(A(Torus(m,\ n))) \leq \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + 1 = b_2((Torus(m,\ n)) + 1$.

Proof. : In Figure 29, vertex O has three uninformed vertices, and M is the furthest uninformed vertex from vertex O in the *torus* graph. The numbers represent the weight of the vertices by using the new algorithm in 2-broadcasting. $w(u)$ denotes the weight of a vertex u. From Figure 29 and because of the symmetry, we can see that $w(Z_0) = w(V_0) = w(U_0)$. From the definition of weight, we know that $w(Z_1) = w(Z_0) + 2$ and $w(V_1) = w(V_0) + 1$. Therefore, $w(Z_1) > w(V_1) = w(U_1)$. Similarly, we can see that $w(Z_i) > w(V_i) = w(U_i)$ when $i \geq 1$.

Figure 30 (a) and (b) illustrate the two possibilities after the first round of 2-broadcasting in $Torus(m,n)$. Because $w(Z_i) > w(V_i) = w(U_i)$ when $i \geq 1$, vertex O will inform vertex Z_i in the second round, and vertex Z_i will inform vertex Z_{i-1} in the third round. This continues until vertex Z_1 is informed. Note that $w(Z'_i) > w(Z'_{i-1}) > w(V'_{i-1})$, so the same scheme is processed on the side of vertex O'. In the

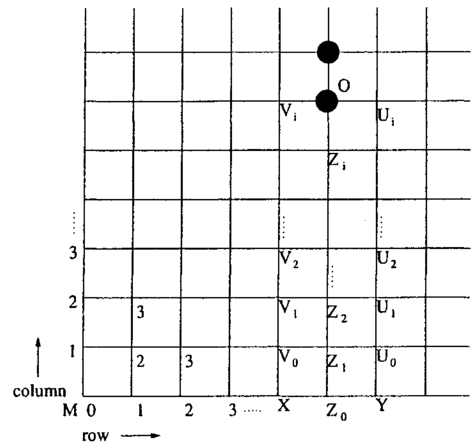

Figure 29: The weights in $Torus(m,\ n)$ in 2-broadcasting

case illustrated by Figure 30 (a), it takes $\lceil \frac{m}{2} \rceil - 1$ rounds until vertex Z_1 is informed, while in the case of Figure 30 (b), it takes $\lfloor \frac{m}{2} \rfloor - 1$ rounds. So, it takes at most $\lceil \frac{m}{2} \rceil - 1 \leq \lfloor \frac{m}{2} \rfloor$ rounds until vertex Z_1 is informed.

Depending on whether m is odd or even and which case illustrated in Figure 30 happens, Z_0 and Z_0' could either be the same vertex or not. When they are the same vertices, there are two possibilities in the two rounds after vertex Z_1 is informed (see Figure 31). When, Z_0 and Z_0' are not the same vertex, there are three possibilities in the two rounds after vertex Z_1 is informed (see Figure 32). During the matching, TBA first gives mates to the vertex with fewer uninformed neighbors. In Figure 32 (c), at the beginning of the second round after Z_1 is informed, vertex Z_0 has more uninformed neighbors than Z_1' has. So, vertex Z_1' is matched before vertex Z_0. Therefore, vertex Z_1' sends the message to vertex Z_0' in the second round after Z_1 is informed, although Z_0' is also an uninformed neighbor of vertex Z_0. Thus, in two rounds after Z_1 is

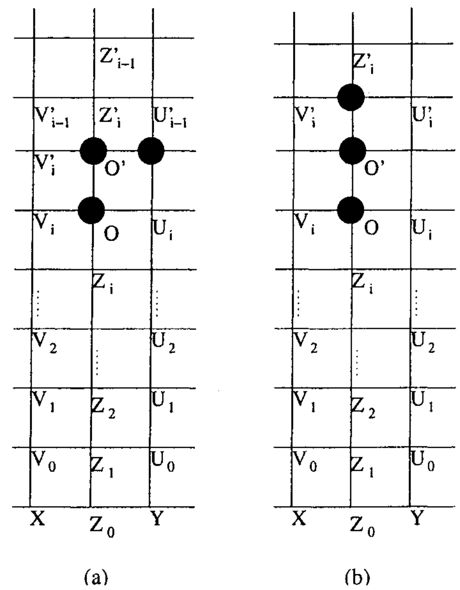

Figure 30: The first step of 2-broadcasting in the torus graph

informed, the vertices on all the three columns shown in both Figure 31 and Figure 32 are informed. After this, it takes $\lfloor \frac{n}{2} \rfloor - 1$ rounds to inform other columns. Therefore, the total rounds needed to finish 2-broadcasting in $Torus(m, n)$ by using the new algorithm is at most $\lfloor \frac{m}{2} \rfloor + 2 + \lfloor \frac{n}{2} \rfloor - 1 = \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + 1$ rounds, which is one round more than the optimal 2-broadcast time. □

Theorem 10. *When $k \geq 3$, $b_k(A(Torus(m,\ n))) \leq \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor = b_k((Torus(m,\ n))$.*

Proof. : Figure 33 illustrates the first round of k-broadcasting ($k \geq 3$) in a torus graph. Assuming that u_1 and v_1 are on the same column, both u_2 and v_2 will be informed in the second round because both u_1 and v_1 have only 3 uninformed neighbors. Therefore, the vertices on this column will be informed in $\lfloor \frac{m}{2} \rfloor$ rounds, and then other vertices will be informed in $\lfloor \frac{n}{2} \rfloor$ rounds. This scheme is the same as the

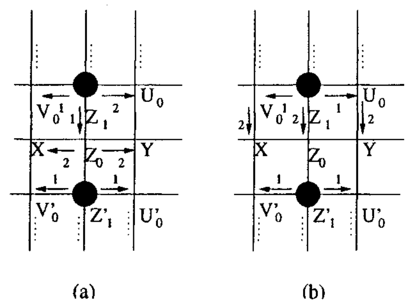

Figure 31: Two possibilities when Z_0 and Z'_0 are the same vertex

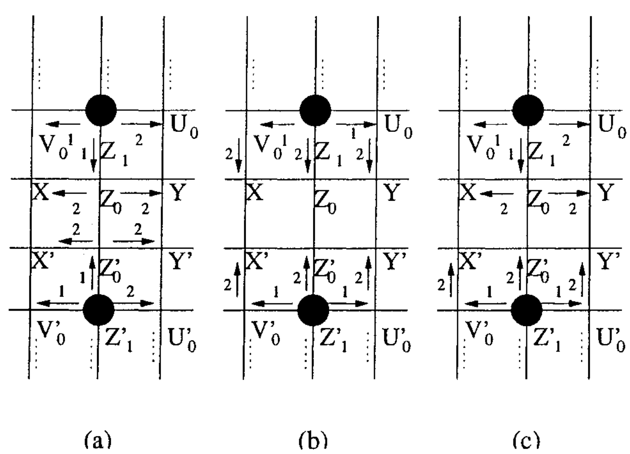

Figure 32: Three possibilities when Z_0 and Z'_0 are not the same vertex

optimal k-broadcasting scheme in $Torus(m, n)$. □

3.2.3 Experimental Results

This section presents the test results of TBA for 1-broadcasting in several commonly used topologies and three graph models.

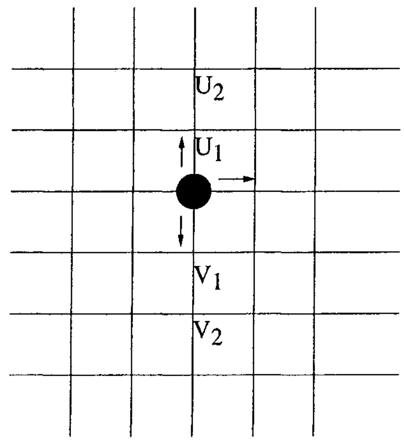

Figure 33: the k-broadcasting in Torus graph

Test Results in Commonly Used Topologies

In this section and the following tables, S_d, BF_d, H_d, SE_d, $UB(2,d)$ and CCC_d abbreviate the *Star* graph, *Butterfly* graph, *HyperCube* graph, *Shuffle Exchange* graph, *deBruijn* graph and *Cube-Connected Cycle* of dimension d respectively. GCR_n, $G_{2,D}$ and $G_{3,D}$ stand for the generalized Chordal rings, the optimal double fixed step graph and the triple fixed step graph $G(3D^2+3D+1, D, D+1, 2D+1)$ with diameter D. *Low* and *Up* stand for the best known theoretical lower and upper bounds, respectively. *TB* stands for the minimum test result of TBA. *Opt* is the optimal broadcast time of a graph. These bounds and optimal broadcast times are presented in [9], [11], [14], [24], [51], [61] and [63]. As in [3], TBA was tested on $UB(2,d)$, SE_d, BF_d and CCC_d. TBA was tested for $d \leq 20$ in $UB(2,d)$ and SE_d, and for $d \leq 16$ in BF_d and CCC_d, while in [3], the authors have values for $d \leq 14$. All the results for $d \leq 14$ are the same as in [3], except for 4 cases: in 2 cases TBA gives a better

	CCC_d			BF_d		
d	**Low**	**Up**	**TB**	**Low**	**Up**	**TB**
3	6	7	6	5	5	5
4	9	9	9	7	7	7
5	11	12	11	8	9	9
6	13	14	13	10	11	10
7	16	17	16	11	13	12
8	18	19	18	13	15	14
9	21	22	21	15	17	16
10	23	24	23	16	19	18^-
11	26	27	26	18	21	19
12	28	29	28	19	23	21*
13	31	32	31	21	25	23
14	33	34	33	23	27	25^-
15	36	37	36	24	29	27
16	38	39	39	26	31	29

Table 2: Test results in CCC_d and BF_d

result (denoted by *) and in 2 cases the results from [3] are better (denoted by $^-$). In all the four cases the difference is one round. In fact, TBA generates new upper bounds on 1-broadcast time for CCC_d when $d = 15$, for BF_d when $15 \leq d \leq 16$, and for $UB(2,d)$ and SE_d when $15 \leq d \leq 20$. In addition, TBA was tested on H_d and S_d graph, providing again new upper bounds for S_d. TBA generates optimal 1-broadcast time in GCR_n(1, -1, $3D$), GCR_n(1, -1, $3D$+1) and $G_{3,D}$, but it spends one round more than the optimal broadcast scheme for 1-broadcasting in $G_{2,D}$ when D is greater than 20.

	S_d						
d	**Low**	**Up**	**TB**	**d**	**Low**	**Up**	**TB**
3	3	3	3	7	13	16	14
4	5	6	5	8	16	21	16
5	7	9	8	9	19	22	20
6	9	13	11				

Table 3: Test results in S_d

	H_d		$\boldsymbol{UB}(2,d)$			$\boldsymbol{SE}_d$	
d	**Opt**	**TB**	**Low**	**Up**	**TB**	**Opt**	**TB**
5	5	5	7	9	6*	9	9
6	6	6	8	11	8	11	11
7	7	7	10	12	9	13	13
8	8	9	11	14	11	15	15
9	9	10	12	15	12	17	17
10	10	11	14	17	14	19	19
11	11	12	15	18	15	21	21
12	12	13	16	20	17	23	24
13	13	14	18	21	18	25	26
14	14	15	19	23	20	27	28
15	15	16	20	24	21	29	30
16	16	17	22	26	23	31	32
17	17	18	23	27	25	33	34
18	18	19	24	29	26	35	36
19	19	20	26	30	28	37	38
20	20	21	27	32	29	39	40

Table 4: Test results in H_d, $UB(2,d)$ and SE_d

GCR_n (1, -1, $3D$)			GCR_n(1, -1, $3D$+1)		
D	**Opt**	**TB**	**D**	**Opt**	**TB**
11	13	13	10	11	11
15	17	17	12	13	13
17	19	19	16	17	17
29	31	31	18	19	19
39	41	41	20	21	21
49	51	51	40	41	41
59	61	61	50	51	51
69	71	71	60	61	61
79	81	81	80	81	81
89	91	91	90	91	91
99	101	101	100	101	101

Table 5: Test results in GCR_n

$G_{2,D}$			$G_{3,D}$		
D	**Opt**	**TB**	**D**	**Opt**	**TB**
10	12	12	10	13	13
20	22	22	20	23	23
30	32	33	30	33	33
40	42	43	40	43	43
50	52	53	50	53	53
60	62	63	60	63	63
70	72	73	70	73	73
80	82	83	80	83	83
90	92	93	90	93	93
100	102	103	100	103	103

Table 6: Test results in $G_{2,D}$ and $G_{3,D}$

Tiers: 1105 vertices					
Edges	RH	TB	Edges	RH	TB
1106	24	24	1324	23	21
1110	24	23	1326	23	21
1214	22	21	1331	20	20
1216	22	21	1447	22	21
1220	22	21	1449	21	20

Table 7: Test Results in Tiers Model: 1105 vertices

Test Results in Three Graph Models

The ns-2 is a widely used simulator for networking research, which creates topologies by using several models. In order to compare TBA with the algorithm from [3], three different network design models from ns-2 are considered: GT-ITM *Pure Random* [82], GT-ITM *Transit-Stub* (TS) [82] and *Tiers* [15].

The *Tiers* model is designed to generate test networks for routing algorithms. The model produces graphs corresponding to the data communication networks such as **IP** network and **ATM** network [15]. GT-ITM Transit-Stub is a well-known model for the Internet. The Internet can be viewed as a set of *routing domains.* A domain is a group of hosts on the Internet. We can consider a domain to be an independent network, where all vertices in a domain share routing information. Just like the real Internet, interconnected domains compose the graphs generated by GT-ITM Transit-Stub [82]. GT-ITM PureRandom is a standard random graph model. Considering each pair of vertices, an edge is added between them with probability p. Many models are variations of this model. This model is often used in studying networking problems, although it does not correspond to real networks [82].

Tiers: 2210 vertices					
Edges	**RH**	**TB**	**Edges**	**RH**	**TB**
2209	28	27	3028	31	29
2234	26	25	3209	30	29
2409	32	31	3225	26	24
2427	25	24	3409	32	32
2609	33	32	3428	27	26
2628	26	26	3609	30	29
2809	29	29	3627	30	29
2833	27	27	3809	28	28
3009	32	31	4207	27	26

Table 8: Test Results in Tiers Model: 2210 vertices

The tables in this section represent some of the test results of TBA and the algorithm from [3] in the above three models. The results of the algorithm from [3] and TBA are presented in column RH and TB, respectively. In total, TBA was tested on about 200 different graphs generated by the three models for $155 \leq |V| \leq 4400$. In only one case (shown by * in Table 10), TBA gave a 1-broadcast time that was one more than the 1-broadcast time obtained by using the algorithm from [3]. In all other cases, TBA generated either the same 1-broadcast times as in [3] or better. In the Pure Random model we got a 12% improvement. In the Transit-Stub model TBA gave better 1-broadcast times in more than 40% of the cases. TBA worked better under the Tiers model, as it gave smaller 1-broadcast times in about 60% of the cases.

Pure Random							
Vertices	**Edges**	**RH**	**TB**	**Vertices**	**Edges**	**RH**	**TB**
200	346	10	10	500	1725	10	10
200	475	9	8	500	1830	10	9
200	595	8	8	750	2099	11	11
300	684	10	10	750	2236	11	10
300	756	10	9				

Table 9: Test Results in GT-ITM Pure Random Model

TS: 600 vertices					
Edges	**RH**	**TB**	**Edges**	**RH**	**TB**
1169	14	13	1222	15	14
1190	14	14	1231	14	13
1200	16	15	1232	14	13
1206	14	14	1247*	13	14
1219	15	14	1280	14	13

Table 10: Test Results in TS Model: 600 vertices

TS: 1056 vertices					
Edges	**RH**	**TB**	**Edges**	**RH**	**TB**
2115	17	16	2176	17	16
2121	17	17	2177	18	17
2134	17	16	2185	16	16
2142	16	15	2187	16	15
2147	16	15	2204	16	15
2149	16	15	2219	17	16
2151	15	15	2220	15	15
2167	17	16	2230	16	15
2169	17	17	2255	15	14

Table 11: Test Results in TS Model: 1056 vertices

3.3 Derived Heuristic for Gossip

Given an arbitrary graph G and an integer p, the problem that whether there exist a gossip scheme in G with a gossip time less or equal to p is NP-complete [12]. Several heuristics for gossiping have been presented in [3] and [25]. Among them, the algorithm in [3] is the best existing heuristic in practice.

This section presents a heuristic for gossip in arbitrary graphs. This heuristic is derived from TBA, so we call it the Tree Based Algorithm for Gossip, or TBAG. The input of TBAG is a graph $G = (V, E)$ and its output is a gossip scheme for graph G. In each round t of TBAG, a message s has an informed area (vertices holding s), an uninformed area (vertices not holding s), a bright border (vertices are holding s and having uninformed neighbors) and a dark border (a set of vertices are not holding s and have informed neighbors). For a message s, TBAG performs a variant of BFS from the bright border towards the uninformed border and labels each visited vertex u with the shortest distance from u to the bright border of s. Then, TBAG calculates the weights of all uninformed vertices of the message s by the Procedure Calculate_Weight in TBA. Given an edge (u, v), the weight of (u, v) of message s is the weight of v if u is in the bright border of s, and v is in the dark border of s, and is zero otherwise. In total, the final weight of an edge is the sum of its weights of all the $|V|$ messages. Then, TBAG finds the Maximum-Weighted Matching for graph G based on the weights of all edges. Finally, messages are exchanged between the two vertices in each matched vertex pair. In the simulation of TBAG, the matching is performed by the program written by Ed Rothberg, who implemented H. Gabow's N-cubed weighted matching algorithms [26].

The pseudocode of TBAG is presented below.

Heuristic *TBAG (Tree Based Algorithm for Gossip)*

Input: graph $G = (V, E)$, a vertex u in G is holding message u.

Output: gossip scheme and gossip time t of G.

1. $t = 0$ and $w(u, v) = 0$ for any edge $(u, v) \in E$, where $w(u, v)$ stands for the weight of any edge $(u,\ v)$;

2. **For** $i = 1$ to $|V|$, *Uninformed[i]* $\leftarrow$ $|V|$ - 1; /* number of uninformed vertices of all messages */

3. **While** exist i such that *Uninformed[i]* != 0

   3.1. t++;

   3.2. **For** each message s of the $|V|$ messages

      3.2.1. $bb(t, s) \leftarrow$ all informed vertices of message s with uninformed neighbors, where $bb(t, s)$ stands for the set of vertices on the bright border of message s in round t.

      3.2.2. *Remote* $\leftarrow \emptyset$; /* stack used to calculate the weight of each uninformed vertex */

      3.2.3. Perform variant BFS from $bb(t, s)$ to uninformed vertices of message s, and label any uninformed vertex v with $D(v, s, t)$, where $D(v, s, t)$ stands for the shortest distance from vertex v to $bb(t, s)$;

3.2.4. During the process of variant BFS, push vertices in $bb(t, s)$ and all uninformed vertices into stack *Remote*;

3.2.5. Calculate $w(u, s, t)$ for each vertex u in stack *Remote* by using Procedure Calculate_weight (defined in Section 3.2), where $w(u, s, t)$ stands for the weight of vertex u of message s in round t;

3.2.6. For any vertex u in $bb(t, s)$, if $(u,\ v)$ is an edge in G and v is uninformed of message s, then $w(u, v) = w(u, v) + w(v, s, t)$.

3.3 Calculate the Maximum-Weighted Matching;

3.4 Messages are exchanged between the two vertices in any matched vertex pair. When message i is sent to a vertex, *Uninformed[i]* = *Uninformed[i]* - 1.

The tables in this section represent some of the test results of TBAG and the algorithm from [3]. In these tables, $|V|$ and $|E|$ denote the number of vertices and the number of edges in the tested graphs respectively. The results of the algorithm from [3] and TBA are presented in column RH and TB, respectively. Generally speaking, TBAG performs worse than the Roung_Heuristic in several commonly used topologies, such as $UB(2, d)$, BF_d, SE_d and CCC_d, but TBAG performs better in the graphs generated by three network design models: $Tiers$, $GT - ITM\ TS$ and $GT - ITM$ *Random*. The most significant advantage of TBAG is its low time complexity. The time complexity of TBAG is $O(|V||E|)$ in each round without matching, while the time complexity of Round_Heuristic [3] is $O(|V|^3|E|)$ in each round without matching.

	$\boldsymbol{UB}(2,d)$		BF_d		$\boldsymbol{SE}_d$		CCC_d	
d	**RH**	**TB**	**RH**	**TB**	**RH**	**TB**	**RH**	**TB**
3	4	4	5	5	5	5	7	8
4	6	6	7	7	7	7	9	9
5	8	8	10	10	10	10	13	13
6	10	10	12	13	12	13	14	15
7	12	12	15	16	15	16	19	20
8	14	14	17	18	17	18	19	22
9	16	17	20	21	20	21		
10	18	19			23	25		

Table 12: Gossip times in $UB(2,d)$, BF_d, SE_d and CCC_d

$\|V\|$	$\|E\|$	**TB**	**RH**
20	27	11	13
40	51	25	27
50	53	24	26
57	80	24	26
160	164	36	39
180	190	41	44
235	239	38	45
360	373	47	51
490	495	55	53
610	618	65	63
770	863	64	66

Table 13: Gossip times in *Tiers*

$\|V\|$=**100**			$\|V\|$=**200**		
$\|E\|$	**TB**	**RH**	$\|E\|$	**TB**	**RH**
187	20	24	368	25	37
177	19	22	335	29	30
168	21	23	357	25	35
166	21	26	355	25	33
185	19	27	361	29	31
176	17	26	340	26	31
179	20	27	345	26	33
191	17	24	354	26	40
189	22	25	353	27	38
192	22	30	357	26	33

Table 14: Gossip times in *GTITM-TS*

$\|V\|$=100			$\|V\|$=300			$\|V\|$=400		
$\|E\|$	**TB**	**RH**	$\|E\|$	**TB**	**RH**	$\|E\|$	**TB**	**RH**
148	16	21	926	13	14	779	19	23
145	16	19	943	14	14	866	17	21
158	14	16	1125	12	13	902	16	17
177	16	19	1267	12	12	1319	13	14
187	13	15	1350	12	12	1405	13	14
204	13	15	1475	11	11	1645	12	13
211	12	13	1569	11	11	2049	12	12
213	12	15	1698	11	11	2389	12	12
225	12	13	1788	11	11	2488	11	11
235	12	13	1989	11	11	2881	11	12

Table 15: Gossip times in *GTITM-Random*

Chapter 4

Minimum k-Broadcast Graphs

Previous chapters presented efficient k-broadcasting in a given graph. This chapter will focus on how to construct efficient graphs or network topologies that have small k-broadcast time. A *k-broadcast graph* G is a graph on n vertices where $b_k(G) = \lceil log_{k+1}n \rceil$. Evidently, a complete graph K_n is a k-broadcast graph, since $b_k(K_n) = \lceil log_{k+1}n \rceil$. However, K_n is not minimal in terms of the number of edges. We can remove edges from K_n and obtain a graph with k-broadcast time $\lceil log_{k+1}n \rceil$. $B_k(n)$ stands for the minimum possible number of edges in a k-broadcast graph on n vertices. A k-broadcast graph on n vertices with $B_k(n)$ edges is a *minimum k-broadcast graph* or *k-mbg*. This chapter first presents previous results on *k-mbg*'s and then presents several new *k-mbg*'s.

4.1 Previous Results

Up until now, no general method exists to determine $B_k(n)$ for an arbitrary value of n. Moreover, the previous studies have suggested that the k-*mbg*'s are extremely difficult to construct. When n is small, k-*mbg*'s can be found by exhaustive case analysis. This technique is no longer effective when n is large, due to a rapid increase of the number of possible graphs [20]. In most cases, the previous k-*mbg*'s are found by first defining a lower bound l on $B_k(n)$ and then looking for a k-broadcast graph on n vertices with l edges.

Most of the previous work in this area has been for $k = 1$. The result $B_1(2^p) = p \cdot 2^{p-1}$ was shown in [20]. In order to inform 2^p vertices in p rounds, each vertex in a 1-*mbg* on 2^p vertices must have degree at least p . Thus, such a 1-*mbg* must have at least $\frac{1}{2}(p \cdot 2^p) = p \cdot 2^{p-1}$ edges, so, $B_1(2^p) \geq p \cdot 2^{k-1}$. Then, we need to construct 1-broadcast graphs with 2^p vertices and $p \cdot 2^{p-1}$ edges. In the construction presented in [20], we first take two copies of a minimum 1-broadcast graphs on 2^{p-1} vertices and then add an edge between any two corresponding vertices of the two graphs. This process eventually reduces to the graph on one vertex. In fact, the results of such a construction are hypercubes (see the introduction on hypercube in Chapter 1). The recursive circulant graphs [70] and the Knödel graphs [52] for $n = 2^p$ are also 1-*mbg*'s on 2^p vertices.

$B_1(2^p - 2) = (p-1)(2^{p-1} - 1)$ was presented in [13] and [50]. Any vertex in a 1-*mbg* on $2^p - 2$ vertices must have degree at least $p-1$. Otherwise, when 1-broadcasting is originated by a vertex u with degree less than $p-1$, u could inform at most $2^{p-1} + 2^{p-2} + \cdots + 2^2 + 1 = 2^p - 3 < 2^p - 2$ vertices in p rounds. Thus, $B_1(2^p - 2) \geq \frac{(p-1)(2^{p-1}-1)}{2}$. Then, it suffices to construct 1-broadcast graphs on $2^p - 2$ vertices and $\frac{(p-1)(2^{p-1}-1)}{2}$ edges. The modified Knödel graphs are 1-broadcast graphs on $2^p - 2$ vertices and $\frac{(p-1)(2^{p-1}-1)}{2}$ edges [50]. Let H_p stand for the modified Knödel graph on $2^p - 2$ vertices, and these vertices in H_p are denoted by $v_0, v_1, \cdots, v_{2^p-3}$. An edge exists between vertex v_i and v_j iff $i + j = (2^r - 1) \ mod \ (2^p - 2)$, where $1 \leq r \leq p - 1$. In order to inform H_p in p rounds, an informed vertex v_i sends the message to vertex v_j in round r, where $i+j = (2^r - 1) \ mod \ (2^p - 2)$, for $1 \leq r \leq p-1$ and $i + j = 1 \ mod \ (2^p - 2)$ for $r = p$.

Aside from the cases that $n = 2^p$ and $n = 2^p - 2$, the result of $B_1(n)$'s is only known for some small values of n. Table 16 summarizes the previously known values of $B_1(n)$.

For $k \geq 1$, $B_k((k+1)^p) = \frac{1}{2}kp(k+1)^p$ $(k \geq 1)$ was presented in [28] and [58]. A p-dimensional k-hypercube graph is a k-*mbg* on $(k+1)^p$ vertices, where each vertex corresponds to a p-bit string on $k+1$ alphabets and two vertices are linked with an edge iff their strings differ by precisely one bit. In a p-dimensional k-hypercube graph, the k-broadcasting can be performed in $p = \lceil log_{k+1} n \rceil$ rounds by using the following scheme: in round i, each informed vertex sends the message to its k neighbors that differ in the ith bit. Two more general results are presented in [58]. For $n \leq k+1$, $B_k(n) = \frac{1}{2}n(n-1)$ and a minimum k-broadcast graph is the complete graph on n

n	$B_1(n)$	Ref.	n	$B_1(n)$	Ref.
1	0	[20]	20	26	[65]
2	1	[20]	21	28	[65]
3	2	[20]	22	31	[65]
4	4	[20]	26	42	[74]
5	5	[20]	27	44	[74]
6	6	[20]	28	48	[74]
7	8	[20]	29	52	[74]
8	12	[20]	30	60	[7]
9	10	[20]	31	65	[7]
10	12	[20]	32	80	[20]
11	13	[20]	58	121	[74]
12	15	[20]	59	124	[74]
13	18	[20]	60	130	[74]
14	21	[20]	61	136	[74]
15	24	[20]	62	155	[13] [50]
16	32	[20]	63	162	[30] [57]
17	22	[68]	127	389	[79]
18	23	[7] [78]	2^p	$p2^{p-1}$	[20]
19	25	[7] [78]	$2^p - 2$	$(p-1)(2^{p-1}-1)$	[13] [50]

Table 16: $B_1(n)$'s and References

n	$B_2(n)$	$B_3(n)$	$B_4(n)$	n	$B_2(n)$	$B_3(n)$	$B_7(n)$
1	0 [58]	0 [58]	0 [58]	9	18 [58]		
2	1 [58]	1 [58]	1 [58]	10	12 [58]	15 [34]	
3	3 [58]	3 [58]	3 [58]	11	13 [58]	18 [34]	
4	3 [58]	6 [58]	6 [58]	12	15 [58]		
5	5 [58]	4 [58]	10 [58]	24	48 [34]		
6	7 [58]	7 [58]	5 [58]	50			175 [34]
7	10 [58]	9 [58]	9 [58]				
8	12 [58]	11 [58]	11 [58]				

Table 17: $B_k(n)$'s and References

vertices [58]. $B_k(k+2) = k+1$ and a minimum k-broadcast graph on $k+2$ vertices is the star with $k+1$ edges around a central vertex [58]. Minimum k-broadcast graphs for all n in the range $k+3 \leq n \leq 2k+3$ were presented in [53].

Except these general results, values of $B_k(n)$ for some particular values of k and n were presented in [58] and [34]. Table 17 summarizes these results.

4.2 New 1-*mbg*'s

This section addresses mbg's on $2^p - 1$ vertices, where $p+1$ is a prime number. When $p+1$ is a prime number, $p+1$ devides $2^p - 1$ (Fermat's little theorem). The lower bound on $B_1(2^p - 1)$ is $\frac{p^2(2^p-1)}{2(p+1)}$ [57]. For any vertex u in a 1-*mbg*'s on $2^p - 1$ vertices, $d(u) \geq p-1$, where $d(u)$ stands for the degree of vertex u. Moreover, for a vertex u where $d(u) = p-1$, there must exist a neighbor v of u such that $d(v) \geq p$ [7] [30] [57]. The 1-*mbg*'s on $2^p - 1$ vertices for $p = 4$ and $p = 6$ have two types of vertices: vertices of degree p and $p-1$. The number of vertices of degree p is $\frac{2^p-1}{p+1}$ and the number of

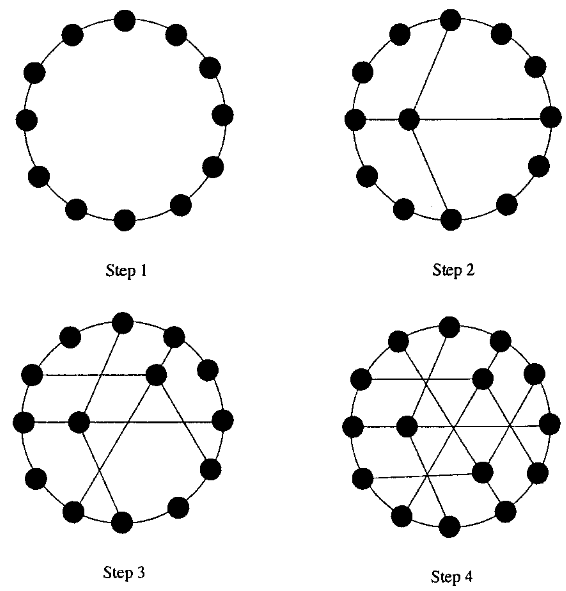

Figure 34: The construction of 1-*mbg* on 15 vertices

vertices of degree $p-1$ is $\frac{p(2^p-1)}{p+1}$. All the vertices of degree $p-1$ form a Hamiltonian cycle, or a ring, of length $\frac{p(2^p-1)}{p+1}$. All the vertices of degree p are connected to the vertices on the ring alternatively. This means that these 1-*mbg*'s can be composed by the combination of a ring on $\frac{p(2^p-1)}{p+1}$ vertices and $\frac{2^p-1}{p+1}$ copies of star graphs. All leaves of these star graphs are on the ring. Since $p+1$ devides 2^p-1, a vertex with degree p, which is the center of a star, has p neighbors of degree $p-1$; while a vertex with degree $p-1$ has exactly one neighbor of degree p. Up until now, each vertex on the ring has three incident edges. Therefore, chords are added to allow them to have degree $p-1$, when $p>4$. Figure 34 illustrates the construction of a 1-*mbg* on 15 vertices. This 1-*mbg* includes a ring on 12 vertices and 3 stars on 4 vertices.

Following these observations, we can define the ring-star graph, which are candidates for k-mbg's on $2^p - 1$ vertices, where $p + 1$ is a prime number and $p \geq 4$. Let $R(n)$ stand for a ring-star graph on n vertices, and the vertices in $R(n)$ are numbered by $0, 1, \cdots, 2^p - 2$. Vertices $0, 1, \cdots, \frac{p(2^p-1)}{p+1} - 1$ are *ring* vertices since they are located on a ring. All other vertices are *switch* vertices. There are two types of edges in $R(n)$. The edges among the ring vertices are chords, and the edges between switch vertices and ring vertices are called switch edges. For each ring vertex v, its incident chords are $\{(v, (v \pm 2^{2i}) \ mod \ \frac{p(2^p-1)}{p+1}), 0 \leq i \leq \frac{p-4}{2}\}$. For each switch vertex v, its incident switch edges are $\{(v, v \ mod \ \frac{p(2^p-1)}{p+1} + i\frac{2^p-1}{p+1}), 0 \leq i \leq p-1\}$.

The number of edges in $R(2^p - 1)$ is $\frac{p^2(2^p-1)}{2(p+1)}$, which is equal to the lower bound on $B_1(2^p - 1)$. Thus, if $b_1(R(2^p - 1)) = \lceil log_2(2^p - 1) \rceil = p$, then $R(2^p - 1)$ is a 1-mbg. Because of the symmetry of the ring-star graph $R(2^p - 1)$, it suffices to show the 1-broadcast scheme of vertex 0 or vertex $\frac{p(2^p-1)}{p+1}$. $R(15)$ and $R(63)$ are 1-mbg's on 15 vertices [20] and 63 vertices [57]. This section will show that $R(1023)$ and $R(4095)$ are also 1-mbg's. However, it is important to mention that systematic description of the 1-broadcast schemes for all ring-star graphs have still not been found.

1-*mbg* on 1023 vertices

Among the 1023 vertices in $R(2^{10} - 1)$, vertices 0,1, ..., 929 ($1093 - \frac{1023}{11} = 930$ vertices in total) have degrees 9, and vertices 930, 931, ..., 1022 ($\frac{1023}{11} = 93$ vertices in total) have degrees 10. The chords of $R(2^{10} - 1)$ are: $\{(v, (v + 1) \ mod \ 930), (v, (v + 4) \ mod \ 930), (v, (v + 16) \ mod \ 930), (v, (v + 64) \ mod \ 930), (v, (v - 1) \ mod \ 930), (v, (v -$

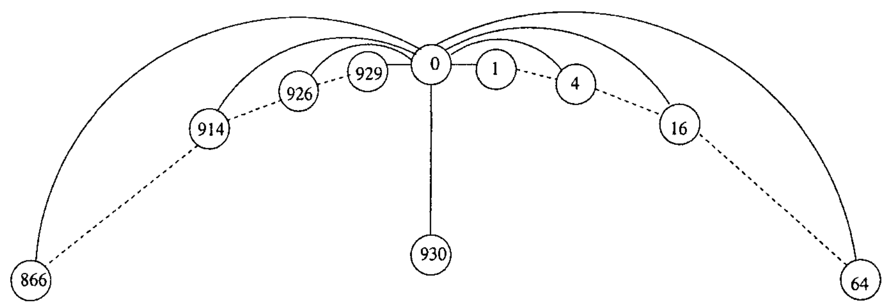

Figure 35: Vertex 0 in $R(1023)$

4) mod 930), $(v,(v-16)$ mod 930), $(v,(v-64)$ mod 930)}, where $0 \leq v \leq 929\}$. The switch edges of $R(2^{10}-1)$ are: $\{(v, v \ mod \ 930+93i)\}$, where $0 \leq i \leq 9$ and $930 \leq v \leq 1022$.

A vertex v where $0 \leq v \leq 929$ has nine incident edges: $(v,(v+1) \, mod \, 930), (v,(v+4) \, mod \, 930), (v,(v+16) \, mod \, 930), (v,(v+64) \, mod \, 930), (v,(v-1) \, mod \, 930), (v,(v-4) \, mod \, 930), (v,(v-16) \, mod \, 930), (v,(v-64) \, mod \, 930)$ and $(v, v \ mod \ 93+930)$.

A vertex v where $930 \leq v \leq 1022$ has ten incident edges: $(v, v-930)$, $(v, v-930+93)$, $(v, v-930+93\times 2)$, $(v, v-930+93\times 3)$, $(v, v-930+93\times 4)$, $(v, v-930+93\times 5)$, $(v, v-930+93\times 6)$, $(v, v-930+93\times 7)$, $(v, v-930+93\times 8)$ and $(v, v-930+93\times 9)$.

It would be difficult to present here the whole $R(1023)$, due to its large number of vertices and edges. So, Figure 35 and Figure 36 only illustrate vertex 0, vertex 930 and their incident edges. In these figures, solid lines represent edges and dashed lines stand for paths between two vertices in the ring.

$B_1(1023) = B_1(2^{10}-1) \geq \frac{10^2(2^{10}-1)}{2(10+1)} = 4650$ [57]. Since there are 930 vertices

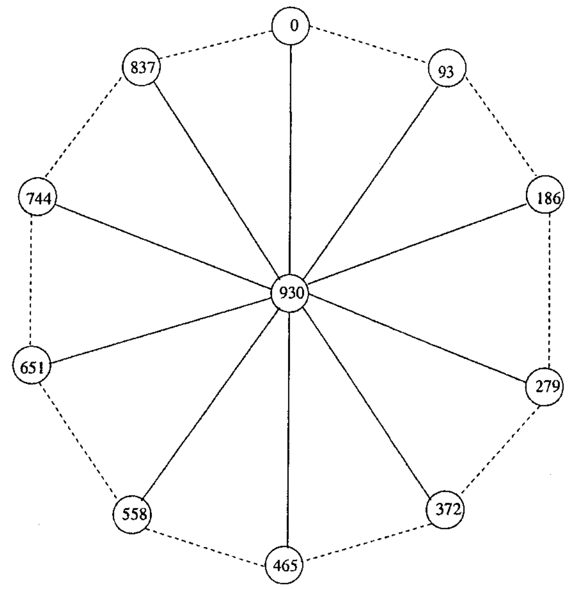

Figure 36: Vertex 930 in $R(1023)$

with degrees 9 and 93 vertices with degrees 10 in $R(2^{10}-1)$, the number of edges in $R(2^{10}-1)$ is also 4650 in total. Appendix A presents the 1-broadcast scheme of vertex 0, which generates $b_1(0, R(1023)) = \lceil log_2(1023) \rceil = 10$. This 1-broadcast scheme is found by performing the heuristic in [3], the Round_Heuristic, on $R(1023)$. Since 0, 1, $\cdots$, 929 are symmetric, it suffices to show the 1-broadcast scheme of vertex 0 or vertex 930, where in the first round, vertex 0 sends the message to vertex 930 or vice versa. Thus, we have the following theorem:

Theorem 11. $B_1(1023) = 4650$.

1-*mbg* on 4095 vertices

Among the 4095 vertices in $R(2^{12}-1)$, vertices 0,1, ..., 3779 ($4095 - \frac{4095}{13} = 3780$ vertices in total) have degrees 11, and vertices 3780, 3781, ..., 4094 ($\frac{4095}{13} = 315$ vertices in total) have degrees 12. The chords of $R(2^{12}-1)$ are: $\{(v,(v+1)\,mod\,3780),(v,(v+4)mod3780),(v,(v+16)mod3780),(v,(v+64)mod3780),(v,(v+256)mod3780),(v,(v-1)\,mod\,3780),(v,(v-4)\,mod\,3780),(v,(v-16)\,mod\,3780),(v,(v-64)\,mod\,3780),(v,(v-256)\ mod\ 3780)$, where $0 \le v \le 3779$. The switch edges of $R(2^{12}-1)$ are: $\{(v, v\ mod\ 3780 + 315i)\}$, where $0 \le i \le 11$ and $3780 \le v \le 4094$.

A vertex v where $0 \le n \le 3779$ has eleven incident edges: $(v,(v+1)mod3780),(v,(v+4)mod3780),(v,(v+16)mod3780),(v,(v+64)mod3780),(v,(v+256)mod3780),(v,(v-1)\,mod\,3780),(v,(v-4)\,mod\,3780),(v,(v-16)\,mod\,3780),(v,(v-64)\,mod\,3780),(v,(v-256)\ mod\ 3780)$, and $(v, v\ mod\ 315 + 3780)$.

A vertex v where $3780 \le n \le 4094$ has twelve incident edges: $(v,\ v-3780)$, $(v,\ v-3780+315)$, $(v,\ v-3780+315\times 2)$, $(v,\ v-3780+315\times 3)$, $(v,\ v-3780+315\times 4)$, $(v,\ v-3780+315\times 5)$, $(v,\ v-3780+315\times 6)$, $(v,\ v-3780+315\times 7)$, $(v,\ v-3780+315\times 8)$, $(v,\ v-3780+315\times 9)$, $(v,\ v-3780+315\times 10)$ and $(v,\ v-3780+315\times 11)$.

Figure 37 and Figure 38 illustrate vertex 0 and vertex 3780 in $R(4095)$ and their incident edges. In these figures, solid lines represent edges, and dashed lines stand for paths between two vertices in the ring.

$B_1(4095) = B_1(2^{12}-1) \ge \frac{12^2(2^{12}-1)}{2(12+1)} = 22680$ [57]. Since there are 3780 vertices with degrees 11 and 315 vertices with degrees 12 in $R(2^{12}-1)$, the number of edges in $R(2^{12}-1)$ is also 22680 in total. Appendix B presents the 1-broadcast scheme

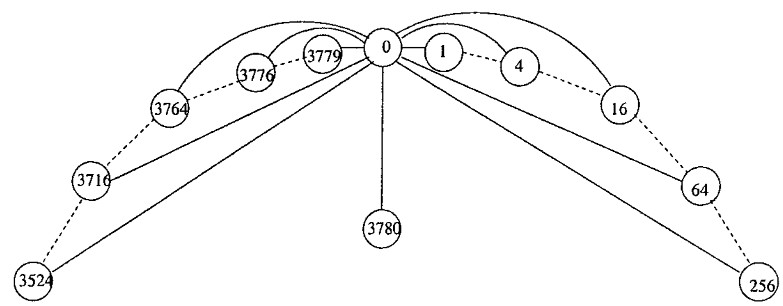

Figure 37: Vertex 0 in $R(4095)$

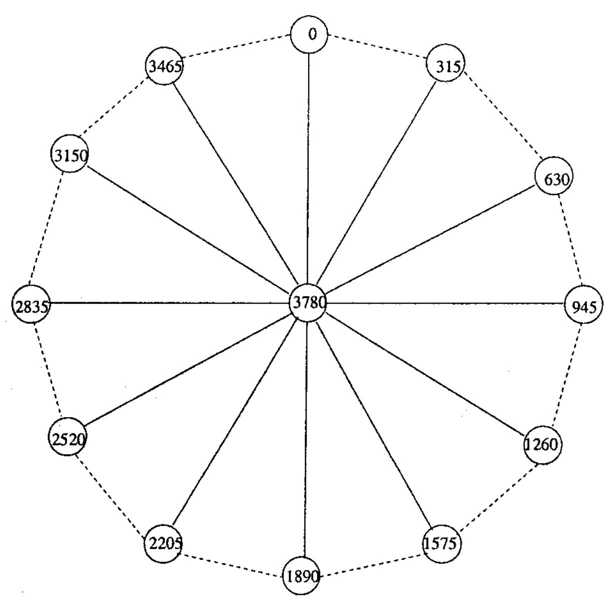

Figure 38: Vertex 3780 in $R(4095)$

of vertex 0, which generates $b_1(0, R(4095)) = \lceil log_2(4095) \rceil = 12$. This 1-broadcast scheme is found by performing the hueristic in [3], the Round_Heuristic, on $R(4095)$. Since 0, 1, · · ·, 3779 are symmetric, it suffices to show the 1-broadcast scheme of vertex 0 or vertex 3780, where in the first round, vertex 0 and vertex 3780 exchange the message. Thus, we have the following theorem:

Theorem 12. $B_1(4095) = 22680$.

4.3 A New 2-*mbg* on 10 Vertices

This section presents a new 2-*mbg* on 10 vertices. Since $B_2(10) = 12$ [58], it therefore suffices to present a graph G on 10 vertices and 12 edges, such that $b_2(G) = \lceil log_3 10 \rceil = 3$. Such a graph and the 2-broadcast schemes of all distinct vertices are illustrated in Figure 40, where the originators are presented with black backgrounds. The arrows in Figure 40 demonstrate the direction in which the messages are sent, while the numbers beside the arrows indicate the rounds in which the messages are sent. This 2-mbg on ten vertices, wherein no vertices have degree 4, is obviously not isomorphic to the one presented in [58], wherein two vertices have degree 4 (see Figure 39).

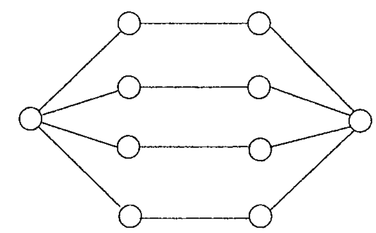

Figure 39: The 2-*mbg* on 10 vertices in [58]

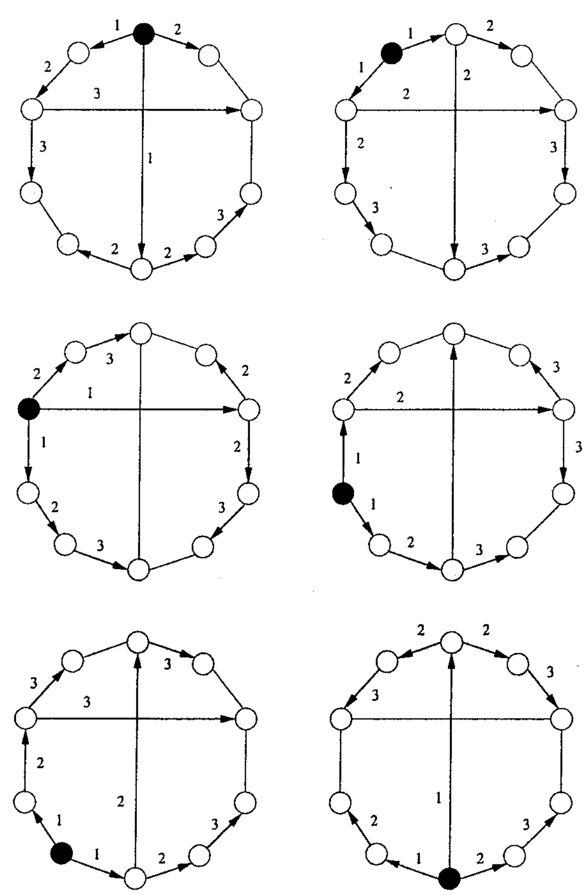

Figure 40: A new 2-*mbg* and its 2-broadcast schemes

Chapter 5

On the k-broadcast Function

Previous studies suggest that k-*mbg*'s are very difficult to find. Therefore, many papers present sparse k-broadcast graphs with a small number of edges, which provide upper bounds on $B_k(n)$ [10] [18] [28] [35] [34] [59]. On the other hand, several papers provide lower bounds on $B_k(n)$ [28] [53] [34]. When a lower bound and an upper bound match at a particular value, we get a new k-*mbg*. This chapter presents an improved lower bound on $B_k(n)$. Since $B_k((k+1)^p) = \frac{1}{2}kp(k+1)^p$ ($k \geq 1$) was presented in [28] and [58], in this chapter, only the case that $n \neq (k+1)^p$ will be discussed. This chapter also presents a small result on the monotonicity of the k-broadcast function.

5.1 Previous Lower Bounds on $B_k(n)$

Consider the $(k+1)$-ary representation of an integer $n-1$: $n-1 = (\gamma_{m-1}\gamma_{m-2}...\gamma_0)_{k+1}$, where $0 \leq \gamma_i \leq k$ for $i = 0, 1, ..., m-1$ and $\gamma_{m-1} \neq 0$. Let p be the index of the leftmost digit which is not equal to k. Then, $n-1 = k(k+1)^{m-1} + \cdots + k(k+1)^{p+1} + \gamma_p(k+1)^p + \gamma_{p-1}(k+1)^{p-1} + \cdots + \gamma_0$. Given n and k, $\beta = 0$ if $p = 0$ or if $\gamma_0 = \gamma_1 = ... = \gamma_{p-1} = 0$. Otherwise, $\beta = \gamma_p + 1$. Thus, $\beta 00...0$ (p digits) $\geq \gamma_p\gamma_{p-1}...\gamma_0$. The authors of [28] and [53] state that $B_k(n) \geq \frac{nk}{2}(m-p-1)$. This lower bound has been improved in [34].

Lemma 14. *[34] In order to inform at least n vertices in $\lceil \log_{k+1} n \rceil$ rounds, a vertex must send the message to at least $k(m-p-1)+\beta$ neighbors during the k-broadcasting.*

This follows that the degree of each vertex in a *k-mbg* is at least $k(m-p-1)+\beta$, which provides the best lower bound on $B_k(n)$ in the present: $B_k(n) \geq \frac{nk}{2}(m-p-1) + \frac{n}{2}\beta$.

5.2 Improved Lower Bound on $B_k(n)$

Most of the previous lower bounds on $B_k(n)$ are obtained by examining the minimum possible degree of the originator in a *k-mbg*. For example, the best lower bound on the degree of a vertex in a *k-mbg* is $k(m-p-1)+\beta$, where m, p, and β are described as in Section 5.1. However, studies on the degrees of a given originator, and that of all its neighbors, will improve the lower bound in [34]. Let $L_k(n)$ stand for the $(k+1)$-ary representation of the integer n. Given $L_k(n)$, $D(n)$ stands for $k(m-p-1)+\beta$. In the first round, the originator u in a *k-mbg* can send the message to up to k of its neighbors. Figure 41 illustrates the originator u and its informed neighbors after round 1. After this, u and its x $(1 \le x \le k)$ informed neighbors will continue to perform the k-broadcasting independently. Thus, vertex u, or at least one of its neighbors, must inform a minimum of $\lceil \frac{n}{k+1} \rceil$ vertices in $\lceil log_{k+1} n \rceil - 1 = \lceil log_{k+1} \lceil \frac{n}{k+1} \rceil \rceil$ rounds. As illustrated in Figure 41, vertex v is a neighbor of vertex u, and $T(v)$ is a tree on vertex v and all the vertices that are informed through vertex v after round 1. $T(u)$ is a tree composed by u and all the vertices that are informed through vertex u after round 1. Assume there are at least $\lceil \frac{n}{k+1} \rceil$ vertices in tree $T(v)$. Then, we can apply Lemma 14 on vertex v and all $\lceil \frac{n}{k+1} \rceil$ vertices. The later discussion will show that, in most of the cases, vertex v has to inform at least $D(n)$ neighbors after round 1. Since there is an edge between vertex u and vertex v, $d(v) \ge D(n) + 1$, where $d(v)$ is the degree of vertex v. In the case that $T(u)$ consists of at least $\lceil \frac{n}{k+1} \rceil$ vertices, again, we apply Lemma 14 on vertex u and $\lceil \frac{n}{k+1} \rceil$ vertices. In most of the cases, vertex u must send the message to its $D(n)$ distinct neighbors after round 1. Thus, $d(u) \ge D(n)+1$, since vertex u has sent the message to at least one of its neighbors in round 1. Therefore, in most of the cases, a vertex u or one of its neighbors must have a degree of at least $D(n)+1$ in a *k-mbg* on n vertices. Before formally presenting the main theorem, several auxiliary lemmas need to be proved.

Lemma 15. *Given* $(\gamma_{m-1}\gamma_{m-2}...\gamma_0)_{k+1}$ *is the* $(k+1)$*-ary representation of the integer* $n-1$*, the* $(k+1)$*-ary representation of the integer* $\lceil \frac{n}{k+1} \rceil - 1$ *is* $(\gamma_{m-1}\gamma_{m-2}...\gamma_1)_{k+1}$.

Proof. When $\gamma_0 < k$, $n = (\gamma_{m-1}\gamma_{m-2}\cdots\gamma_1(\gamma_0+1))_{k+1}$. Since $(\gamma_0+1) > 0$, $\lceil \frac{n}{k+1} \rceil = (\gamma_{m-1}\gamma_{m-2}\cdots\gamma_1)_{k+1} + 1$. Therefore, $\lceil \frac{n}{k+1} \rceil - 1 = (\gamma_{m-1}\gamma_{m-2}\cdots\gamma_1)_{k+1}$. When $\gamma_0 = k$, let γ_p be the rightmost digit which is not equal to k. Then, $n = (\gamma_{m-1}\gamma_{m-2}...(\gamma_p +$

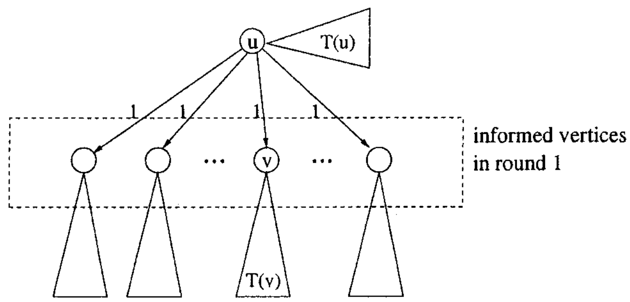

Figure 41: The originator u and its informed neighbors after round 1

$1)0\cdots 0)_{k+1}$ (p 0's after γ_p+1). Thus, $\lceil \frac{n}{k+1} \rceil = (\gamma_{m-1}\gamma_{m-2}...(\gamma_p+1)0\cdots 0)_{k+1}$ ($p-1$ 0's after γ_p+1). Then, $\lceil \frac{n}{k+1} \rceil - 1 = (\gamma_{m-1}\gamma_{m-2}...\gamma_p k\cdots k)_{k+1}$ ($p-1$ k's after γ_p) $= (\gamma_{m-1}\gamma_{m-2}...\gamma_1)_{k+1}$. □

Lemma 16. *For any vertex u in a k-mbg on n vertices, where n is not a power of $k+1$, the vertex u itself or one of its neighbors must have a degree of at least $k(m-p-1)+\beta+1$ except in three cases: (1) $n-1 = kk\cdots k\gamma_0$, (2) $n-1 = k\cdots k\gamma_1\gamma_0$ where $\gamma_0 \neq 0$, and (3) $n-1 = k\cdots k\gamma_x 0\cdots 0\gamma_0$ where $\gamma_0 \neq 0$.*

Proof. Consider a vertex u in a k-*mbg* on n vertices, where $n-1$ is not equal to any of the above three exceptions. By Lemma 14, the degree of vertex u is at least $k(m-p-1)+\beta$. Vertex u can send the message to at most k vertices in the first round, after which there are at most $k+1$ informed vertices. Thus, there exists vertex v that needs to inform $\lceil \frac{n}{k+1} \rceil$ neighbors after the first round, i.e., in $\lceil log_{k+1} n \rceil - 1 = \lceil log_{k+1} \lceil \frac{n}{k+1} \rceil \rceil$ rounds. By Lemma 15, the length of $L_k(\lceil \frac{n}{k+1} \rceil - 1)$ is $m-1$, and $p-1$ is the index of the leftmost digit which is not equal to k. Except for the three cases mentioned in this lemma, it is easy to see that the value of β for $L_k(\lceil \frac{n}{k+1} \rceil - 1) = (\gamma_{m-1}\gamma_{m-2}...\gamma_1)_{k+1}$

is equal to the value of β for $L_k(n-1) = (\gamma_{m-1}\gamma_{m-2}...\gamma_1\gamma_0)_{k+1}$. Thus, by Lemma 14, v must send the message to at least $k((m-1)-(p-1)-1)+\beta = k(m-p-1)+\beta$ distinct neighbors after round 1. Since one of the incident edges of v has been used in round 1, $d(v) \geq k(m-p-1)+\beta+1$. □

By Lemma 16, in a *k-mbg* on n vertices, each vertex with degree $D(n)$ must have a neighbor with degree at least $D(n)+1$ except the three previously mentioned cases. The next question is how many edges are there in such a graph? Let us first discuss the case that the degree of each vertex is either $D(n)$ or $D(n)+1$. In order to minimize the number of vertices with degree $D(n)+1$, a graph should consist of a set of stars, where each star includes $D(n)+2$ vertices, and the leaves of those stars are connected later in a way that the degrees of these leaves in the final graph are $D(n)$. Thus, the minimum possible number of vertices with degree $D(n)+1$ is $\frac{n}{D(n)+2}$, and the minimum possible number of edges in such a graph is $\frac{D(n)(n-\frac{n}{D(n)+2})+\frac{n}{D(n)+2}(D(n)+1)}{2} = \frac{n(D(n)+1)^2}{2(D(n)+2)}$. The following theorem shows that such a graph has a minimum possible number of edges for a graph whose all vertices have degrees greater than or equal to $D(n)$.

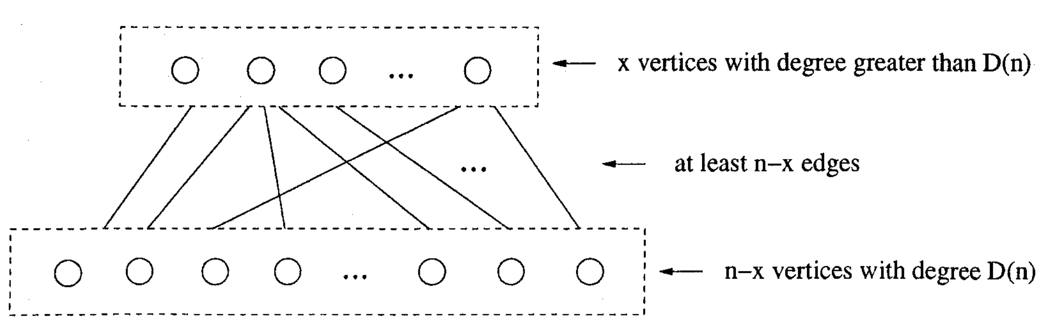

Figure 42: The graph that consists of vertices with a degree of at least $D(n)$

Theorem 13. *Given* $D(n) = k(m-p-1)+\beta$, $B_k(n) \geq \frac{n(D(n)+1)^2}{2(D(n)+2)}$ *except three cases: (1)* $n-1 = kk\cdots k\gamma_0$, *(2)* $n-1 = k\cdots k\gamma_1\gamma_0$ *where* $\gamma_0 \neq 0$, *and (3)* $n-1 = k\cdots k\gamma_x 0\cdots 0\gamma_0$ *where* $\gamma_0 \neq 0$.

Proof. The discussion in this proof is based on the assumption that $n-1$ is not one of the three exceptions. Assume that there are x vertices with degrees greater than or equal to $D(n)+1$ in a *k-mbg* on n vertices.

When $x \geq \frac{n}{D(n)+2}$, $B_k(n) \geq \frac{(n-x)D(n)+x(D(n)+1)}{2} = \frac{nD(n)+x}{2} \geq \frac{nD(n)+\frac{n}{D(n)+2}}{2} = \frac{n(D(n)+1)^2}{2(D(n)+2)}$.

When $x < \frac{n}{D(n)+2}$, there are $n-x$ vertices with degree $D(n)$, and each of the $n-x$ vertices has at least one neighbor with degree greater than $D(n)$. Thus, we can consider the *k-mbg* a graph on two disjointed sets of vertices: the set of vertices with degree $D(n)$ and the set of vertices with degree greater than $D(n)$. There must be at least $n-x$ edges between the two sets of vertices (see Figure 42). Therefore, in total, there are at least $n-x$ incident edges of the x vertices with degree greater than $D(n)$. Thus, $B_k(n) \geq \frac{(n-x)+(n-x)D(n)}{2} = \frac{(n-x)(D(n)+1)}{2}$. Since $x < \frac{n}{D(n)+2}$, $B_k(n) \geq \frac{(n-x)(D(n)+1)}{2} \geq \frac{(n-\frac{n}{D(n)+2})(D(n)+1)}{2} = \frac{n(D(n)+1)^2}{2(D(n)+2)}$. □

This new lower bound improves on the lower bound in [34] by $\frac{n}{2(D(n)+2)}$ whenever the $(k+1)$-ary representation of $n-1$ is not one of the three exceptions.

5.3 A Note on the Monotonicity of the k-broadcast Function

It is a long-standing conjecture that $B_k(n)$ is non-decreasing for n in the range $(k+1)^{m-1}+1 \leq n \leq (k+1)^m$. Theorems 14, 15 and 16 are presented in [32] to prove the monotonicity of $B_k(n)$ in the range $(k+1)^{m-1}+1 \leq n \leq (k+1)^{m-1}+(k+1)^{m-3}-1$. The t-relaxed k-broadcasting refers to the k-broadcasting on n vertices in $\lceil log_{k+1}n \rceil + t$ rounds for some $t \geq 1$. $B_k^t(n)$ is the minimum number of edges in any t-relaxed k-broadcast graph on n vertices.

Theorem 14. *[32] If $n \leq (k+1)^{m-1}+(k+1)^{m-3}$, then $B_k(n) < 2n$.*

Theorem 15. *[32] If for all n, $a \leq n \leq b-1$, where $\lceil log_{k+1}a \rceil = \lceil log_{k+1}b \rceil$, $B_k^t(n) < 2n$, then $B_k^t(n) \leq B_k^t(n+1)$ where $t \geq 0$.*

Theorem 16. *[32] For any $k \geq 1$ and $(k+1)^{m-1}+1 \leq n \leq (k+1)^{m-1}+(k+1)^{m-3}-1$, $B_k(n) \leq B_k(n+1)$.*

Based on these theorems from [32], we have the following theorem.

Theorem 17. *Given $k \geq 1$ and $(k+1)^{m-1}+(k+1)^{m-3} \leq n \leq (k+1)^m$, $B_k(n) \geq B_k((k+1)^{m-1}+(k+1)^{m-3}-1)$.*

Proof. Assume there exists an integer n such that $B_k(n) < B_k((k+1)^{m-1}+(k+1)^{m-3}-1)$ and $(k+1)^{m-1}+(k+1)^{m-3} \leq n \leq (k+1)^m$. By Theorem 14, $B_k((k+1)^{m-1}+(k+1)^{m-3}-1) < 2((k+1)^{m-1}+(k+1)^{m-3}-1)$. Then, $B_k(n) \leq B_k((k+1)^{m-1}+(k+1)^{m-3}-1) < 2((k+1)^{m-1}+(k+1)^{m-3}-1) < 2n$.

Let G be a minimum k-broadcast graph on n vertices and $B_k(n)$ edges. Based on the idea presented in the proof of Theorem 15, we can construct a k-broadcast graph $G' = (V', E')$ on $n-1$ vertices and at most $B_k(n)$ edges as follows.

Since $B_k(n) < 2n$, there is a vertex $u \in G$ such that the degree of u is 3 or less.

When the degree of u is 1, remove the vertex u and its incident edge. The resulting graph G' is a k-broadcast graph on $n-1$ vertices and $B_k(n) - 1$ edges.

When the degree of u is 2, let neighbors of u be v and w. By removing the vertex u with its incident edges and adding the edge (v, w), if it was not already in G, we obtain a k-broadcast graph G' on $n-1$ vertices and at most $B_k(n) - 1$ edges. The k-broadcast scheme in G' is presented in the proof of Theorem 15 in [32]: "Here, the k-broadcast scheme for any originator in G' is easily obtained from the corresponding scheme in G as follows. Without loss of generality, in the scheme for G vertex u is informed by v at time τ. This call can be deleted in the scheme for G'. If u subsequently calls w at some time $\tau + x$ in the scheme for G, replace this call with a call from v to w at time τ. "

When the degree of u is 3, let the neighbors of u be v_1, v_2 and v_3. Remove the vertex u and its incident edges and add the edges (v_1, v_2), (v_1, v_3) and (v_2, v_3) if they were not already in G. The obtained graph G' is a k-broadcast graph on $n-1$ vertices and at most $B_k(n)$ edges. Again, the k-broadcast scheme in G' is presented in the proof of Theorem 15 in [32]: "To k-broadcast from any originator w of G' consider a minimum time k-broadcast scheme S from w in graph G. Without loss of generality, suppose that in S vertex u receives the message from v_1 at time τ and the it calls vertices v_2 and v_3 at times $\tau + x$ and $\tau + y$, respectively where $x \leq y$. (The simpler situation in which u calls fewer of its neighbors is easily handled.) To k-broadcast from vertex w in graph G' we use the scheme S with the following changes: at time τ vertex v_1 calls vertex v_2 (in place of u) and at time $\tau + x$ vertex v_2 calls vertex v_3."

Thus, we can always construct a k-broadcast graph $G' = (V', E')$ on $n-1$ vertices and at most $B_k(n)$ edges. Consequently, $B_k(n-1) \leq |E'| \leq B_k(n)$. Then, by the assumption $B_k(n) < B_k((k+1)^{m-1} + (k+1)^{m-3} - 1)$, $B_k(n-1) \leq B_k(n) < B_k((k+1)^{m-1} + (k+1)^{m-3} - 1) < 2((k+1)^{m-1} + (k+1)^{m-3} - 1) \leq 2(n-1)$ when $n - 1 \geq (k+1)^{m-1} + (k+1)^{m-3} - 1$. Similarly, we can construct a k-broadcast graph on $n-2$ vertices and at most $B_k(n-1)$ edges. Thus, $B_k(n-2) \leq B_k(n-1)$. Eventually, we get $B_k(n-p) \leq B_k(n-p+1) \leq \cdots \leq B_k(n-1) \leq B_k(n)$ where $n - p = (k+1)^{m-1} + (k+1)^{m-3} - 1$. However, $B_k((k+1)^{m-1} + (k+1)^{m-3} - 1) \leq B_k(n)$ contradicts the assumption that $B_k(n) < B_k((k+1)^{m-1} + (k+1)^{m-3} - 1)$. Therefore, $B_k(n) \geq B_k((k+1)^{m-1} + (k+1)^{m-3} - 1)$ for $(k+1)^{m-1} + (k+1)^{m-3} \leq n \leq (k+1)^m$. □

Chapter 6

Conclusions and Future Work

This thesis presents linear algorithms that determine the k-broadcast time and the k-broadcast center in a tree. However, there is no polynomial algorithms (unless $P = NP$) to determine k-broadcast time or optimal k-broadcast schemes in arbitrary graphs, since these problems are NP-complete. More than twenty years have passed since the first heuristic for 1-broadcasting was presented in [75]. Today, the design of efficient heuristics for k-broadcasting is still puzzling many researchers, and I believe it will continue to do so in the near future.

The TBA, which is the heuristic for k-broadcasting in this thesis, is the most efficient heuristic in practice. The most important advantage of TBA is its small time complexity, which is $O(|E|)$ in one round on a graph $G = (V, E)$. The TBA heuristic outperforms previous heuristics on graphs from three graph generators. Moreover, TBA generates almost optimal k-broadcast schemes in grid and torus graphs. One by-product of these theoretical results is a general statement on 1-broadcasting in grid graphs: any 1-broadcast scheme generates optimal 1-broadcast time in a grid graph when the originator is (0, 0). I have used a heuristic to calculate the matching in TBA. Therefore, the performance of TBA may be improved by using the maximum weighted matching process. However, this will obviously increase its time complexity.

There is no general upper bound on the performance of TBA in arbitrary graphs, which means that TBA cannot guarantee an effective performance unless it is tested on a particular topology. This problem could be worse when TBA is working on a dynamically changing network topology. Actually, the following phenomenon generally exists: the authors who presented an efficient heuristic in practice normally could not provide general bounds for the heuristic, such as the heuristic in [3] and the heuristic in this thesis; while heuristics with general upper bounds can be defeated easily in practice. The solution to this dilemma may lie in the combination of two or more heuristics. Some heuristics with general bounds, such as the heuristics presented in [16] and [54], include pure random processes, which might be one of the reasons for their relatively poor performance in practice. Therefore, we can use TBA to substitute for random processes in these heuristics, which will improve their performance in practice without changing the general bounds.

This thesis presents several new *k-mbg*'s for some particular values. More importantly, it defines the ring-star graphs, which are candidates for *k-mbg*'s on $2^p - 1$ vertices, where $p+1$ is a prime number. Hypercubes and the modified Knödel graphs are two major families of *k-mbg*'s. The ring-star graphs could possibly be the third, if its systematic k-broadcast scheme can be found. However, I am still unaware of how to find such a scheme, nor can I guarantee its existence.

The studies on the upper and lower bounds on $B_k(n)$ play important roles in looking for new *k-mbg*'s. This thesis improves the lower bound by considering the minimum possible degree not only of the originator, but also of its neighbors. Since we have studied the originator and its neighbors, perhaps we may continue to study the neighbors of its neighbors, until all vertices in a graph are exhausted. However, the further we go from the originator, the more conditions exist, which dramatically increases the complexity of the analysis.

Bibliography

[1] S. Akers, D. Harel and B. Krishnamurthy, The star graph: an attractive alternative to the n-cube, *Proceedings of the International Conference on Parallel Processing (CPP 1987)*, PA, 1987, pp. 393-400.

[2] W. Aiello, F. Chung and L. Lu, Random evolution in massive graphs, *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001)*, 2001, pp. 510-519.

[3] R. Beier, J. F. Sibeyn, A powerful heuristic for telephone gossiping, *The 7th International Colloquium on Structural Information & Communication Complexity (SIROCCO 2000)*, L'Aquila, Italy, 2000, pp. 17-36.

[4] J.-C. Bermond and P. Fraigniaud, Broadcasting and gossiping in de Bruijn networks, *SIAM J. Comput.*, 23, 1994, pp. 212-225.

[5] J. -C. Bermond, H. A. Harutyunyan, A. L. Liestman and S. Perennes, A Note on the Dimensionality of Modified Knödel Graphs, *Int. J. Found. Comp. Sci.*, 8, 1997, pp. 109-117.

[6] J.-C. Bermond, P. Hell, A. L. Liestman and J. G. Peters, Broadcasting in bounded degree graphs, *SIAM J. Discr. Math.*, 5, 1992, pp. 10-24.

[7] J.-C. Bermond, P. Hell, A. L. Liestman and J. G. Peters, Sparse broadcast graphs, *Discrete Appl. Math.*, 36, 1992, pp. 97-130.

[8] J.-C. Bermond and C. Peyrat, Broadcasting in de bruijn networks, *Proceeding 19th Southeastern Conference on Combinatorics, Graph Theory, and Computing, FL Congressus Numerantium 66*, 1988, pp. 283-292.

[9] P. Berthomé, A. Ferreira and S. Perennes, *Tech. Rept.* LIP, ENS-Lyon, France, 1992.

[10] S. C. Chau and A. L. Liestman, Constructing minimal broadcast networks, *J. Comb. Inf. & Sys. Sci.*, 10, 1985, pp. 110-122.

[11] F. Comellas and P. Hell, Broadcasting in Generalized Chordal Rings, *Networks*, 42(3), 2003, pp. 123-134.

[12] G. Cybenko, D. W. Krumme and K. N. Venkataraman, Gossiping in minimum time, *SIAM J. Comput.*, 21(1), 1992, pp. 111-139.

[13] M. J. Dinneen, M. R. Fellows and V. Faber, Algebraic constructions of efficient broadcast networks, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes 9, Lecture Notes in Computer Science*, 539, Springer, Berlin, 1991, pp. 152-158.

[14] S. Djelloul, Etudes de certains réseaux d'interconnexion: structures et communications, PhD Thesis, Université Paris-Sud, Orsay, 1992.

[15] M. B. Doar, A Better Model for Generating Test Networks, *IEEE GLOBECOM'96*, London, UK, 1996.

[16] M. Elkin and G. Kortsarz, Sublogarithmic approximation for telephone multicast: path out of jungle, *Symposium on Discrete Algorithms*, Baltimore, Maryland, 2003, pp. 76-85.

[17] J. Fàbrega and M. Zaragozá, Fault tolerant routings in double fixed-step networks, *Discrete Applied Mathematics*, 78, 1997, pp. 61-74.

[18] A. Farley, Minimal broadcast networks, *Networks*, 9, 1979, pp. 313-332.

[19] A. Farley and S. Hedetniemi, Broadcasting in grid graphs, *Proc. Eighteenth Southeastern Conference on Combinatorics, Graph Theory and Computing.* Utilitas Mathematica, Winnipeg, 1978, pp. 275-288.

[20] A. Farley, S. Hedetniemi, S. Mitchell and A. Proskurowski, Minimum broadcast graphs, *Discr. Math.*. 25, 1979, pp. 189-193.

[21] A. Farley and A. Proskurowski, Gossiping in grid graphs, *J.Combin.Inform. Systems Sci.*, 5, 1980, pp. 161-172.

[22] U. Feige, D. Peleg, P. Raghavan and E. Upfal, Randomized broadcast in networks, *SIGAL International Symposium on Algorithms*, 1990, pp. 128-137.

[23] R. Feldmann and W. Unger, The Cube-Connected Cycles Network is a Subgraph of the Butterfly Network, *Parallel Processing Letters*, Vol. 2, No. 1, 1992, pp. 13-19.

[24] P. Fraigniaud and E. Lazard, Methods and problems of communication in usual networks, *Discrete Appl. Math.*, 53, 1994, pp. 79-133.

[25] P. Fraigniaud and S. Vial, Approximation Algorithms for Broadcasting and Gossiping, *Journal of Parallel and Distributed Computing*, 43(1), 1997, pp. 47-55.

[26] H. Gabow, Implementation of Algorithms for Maximum Matching on Nonbipartite graphs, *Ph.D. thesis*, Stanford University, 1973.

[27] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. F. Freeman, San Francisco, 1979.

[28] M. Grigni and D. Peleg, Tight bounds on minimum broadcast networks, *SIAM J Discr Math*, 4 (1991), pp. 207-222.

[29] A. Harary and A. J. Schwenk, The communication problem on graphs and digraphs, *J. Franklin Inst.*, 297, 1974, pp. 491-495.

[30] H. S. Harutyunyan, Minimum broadcast networks, *Fourth International Colloquium on Coding Theory*, 1991, pp. 36-40.

[31] H. A. Harutyunyan, Optimal Broadcasting in Digraphs, *Congressus Numerantium*, vol. 148, 2001, pp. 113-128.

[32] H. Harutyunyan and A. L. Liestman, On the Monotonicity of the Broadcast Function, *Discr. Math.*, 262, 2003, pp. 149-157.

[33] H. A. Harutyunyan and A. L. Liestman, k-Broadcasting in trees, *Networks*, 38(3), 2001, pp. 163-168.

[34] H. A. Harutyunyan and A. L. Liestman, Improved upper and lower bounds for k-broadcasting, *Networks*, 37(2), 2001, pp. 94-101.

[35] H. A. Harutyunyan and A. L. Liestman, More broadcast graphs, *Discrete Applied Mathematics*, 98, 1999, pp. 81-102.

[36] H. A. Harutyunyan and B. Shao, On minimum k-broadcast graphs and k-boradcast function, to be submitted.

[37] H. A. Harutyunyan and B. Shao, Algorithms for k-broadcasting and k-broadcast center, to be submitted.

[38] H. A. Harutyunyan and B. Shao, Heuristics for broadcasting and gossiping, submitted.

[39] H. A. Harutyunyan and B. Shao, An efficient heuristic for broadcasting in networks, *Journal of Parallel and Distributed Computing*, in press.

[40] H. A. Harutyunyan and B. Shao, Optimal k-broadcast in Trees, *Congressus Numernatium*, 160, 2003, pp. 117-127.

[41] H. A. Harutyunyan and B. Shao, A Heuristic for k-broadcasting in arbitrary networks, *Seventh IEEE Conference on Graph Theory Application (IV 2003)*, London, England, 2003, pp. 287-293.

[42] H. A. Harutyunyan and B. Shao, An experimental result for broadcast time, *The 21st IASTED International Multi-Conference on Applied Informatics*, Innsbruck, Austria, 2003, pp. 441-445.

[43] H. A. Harutyunyan and B. Shao, A heuristic for broadcasting, *IASTED International Conference on Communications and Computer Networks (CCN 2002)*, Cambridge, USA, 2002, pp. 360-365.

[44] S.M Hedetniemi, S.T. Hedetniemi and A.L. Liestman, A survey of gossiping and broadcasting in communication networks, *Networks*, 18, 1988, pp. 319-349.

[45] P. Hell and A. L. Liestman, Broadcasting in one dimension, *Discr. Appl. Math*, 21, 1988, pp. 101-111.

[46] J. Hromkovic, C.D. Jeschke and B. Monien, Optimal algorithms for dissemination of information in some interconnection networks (extended abstract), *Proc. MFCS'90, Lecture Notes in Computer Science 452*, Springer Verlag, 1990, pp. 337-346.

[47] J. Hromkovic, C.D. Jeschke and B. Monien, Optimal algorithms for dissemination of information in some interconnection networks, *Algorithmica*, Vol. 10, No. 1, 1993, pp. 24-40.

[48] J. Hromkovic, R. Klasing, B. Monien and R. Peine, Dissemination of information in interconnection networks, *Combinatorial Network Theory*, D.-Z. Du, D.F. Hsu(eds.), Kluwer Academic Publishers, 1996, pp. 125-212.

[49] L. Khachatrian and H. S. Harutyunyan, Minimal broadcast trees, *XIV All Union School of Computing Networks*, Minsk, 1989, pp. 36-40.

[50] L. H. Khachatrian and H. S. Harutyunyan, Construction of new classes of minimal broadcast networks, *Proceedings 3rd International Colloquium on Coding Theory*, Armenia, 1990, pp. 69-77.

[51] R. Klasing, B. Monien, R. Peine and E. A. Stöhr, Broadcasting in butterfly and deBruijn networks, *Discrete Appl. Math.*, 53, 1994, pp. 183-197.

[52] W. Knödel, New gossips and telephones, *Discrete Math*, 13, 1975, pp. 95.

[53] J.-C. König and E. Lazard, Minimum k-broadcast graphs, *Discr Appl Math*, 53, 1994, pp. 199-209.

[54] G. Kortsarz and D. Peleg, Approximation algorithms for minimum time broadcast, *SIAM J. Discrete Math.*, 8, 1995, pp. 401-427.

[55] R. Labahn, The telephone problem for trees, *Elektron. Informationsverarb. u. Kybernet.*, 22, 1986, pp. 475-485.

[56] R. Labahn, Extremal broadcasting problems, *Discr. Appl. Math.*, 23, 1989, pp. 139-155.

[57] R. Labahn, A minimum broadcast graph on 63 vertices, *Discrete Applied Mathematics*, 53, 1994, pp. 247-250.

[58] E. Lazard, Broadcasting in DMA-bound bounded degree graphs, *Discr. Appl. Math.*, 37/38, 1992, pp. 387-400.

[59] S. Lee and J. A. Ventura, An algorithm for constructing minimal c-broadcast networks, *Networks*, 38(1), 2001, pp. 6-21.

[60] T. Leighton, *Introduction to parallel algorithms and architectures: array-trees-hypercubes*, Morgan-Kaufmann Publishers, San Mateo, California, 1992.

[61] A. L. Liestman, J. Opatrny and M. Zaragozá, Network Properties of Double and Triple Fixed Step Graphs, *International Journal of Foundations of Computer Science*, 9(1), 1998, pp. 57-76.

[62] A. L. Liestman and J. G. Peters. Minimum broadcast digraphs. *Discr. Appl. Math*, 37/38, 1992, pp. 401-419. reprinted in *Topics in Discrete Mathematics*, Volume 5: Interconnection Networks, J.-C. Bermond (Ed.), 1992, pp.401-419.

[63] A. L. Liestman, J. G. Peters, Broadcast networks of bounded degree, *SIAM Journal on Discrete Maths*, 1(4), 1988, pp. 531-540.

[64] M. Mahéo and J.-F. Saclé, Note on the problem of gossiping in multidimensional grids, *Discrete Appl. Math.*, 53, 1994, pp. 287-290.

[65] M. Mahéo and J.-F. Saclé, Some minimum broadcast graphs, *Discrete Appl. Math.*, 53, 1994, pp. 275-285.

[66] D.S. Meliksetian and C.Y.R. Chen, Communication aspects of the cube-connected cycles, *Proceedings of the International Conference on Parallel Processing (ICPP)*, 1990, pp. I579-I580.

[67] V.E. Mendia and D. Sarkar, Optimal broadcasting on the star graph, *IEEE Trans. Parallel Distrib. System*, 3, 1992, pp. 389-396.

[68] S. Mitchell and S. Hedetniemi, A census of minimum broadcast graphs, *J. Combin. Inform. System Sci.*, 5, 1980, pp. 141-151.

[69] P. Morillo, F. Comellas, and M. A. Fiol, The optimization of chordal ring networks, *Commnication technology*, Q. Yasheng and W. Xiuying (Editors), World Scientific, Singapore, 1987, ISBN 9971-50-349-9, pp. 295-299.

[70] J.H. Park and K.Y. Chwa. Recursive circulant: a new topology for multicomputers networks, (extended abstract), *Proc. Int. Symp. Parallel Architectures Algorithms and Networks (ISPAN'94)*, Kanazawa, Japan, 1994, pp. 73-80.

[71] A. Proskurowski, Minimum broadcast trees, *IEEE Trans on Comput.*, 30, 1981, pp. 363-366.

[72] R. Ravi, Rapid rumor ramification: approximating the minimum broadcast time, *35th Symposium on Foundation of Computer Science (FOCS'94)*, 1994, pp. 202-213.

[73] D. Richards and A. L. Liestman, Generalizations of broadcasting and gossiping, *Networks*, 18, 1988, pp. 125-138.

[74] J.-F. Saclé, Lower bounds for the size in four families of minimum broadcast graphs, *Discrete Math.*, 150, 1996, pp. 359-369.

[75] P. Scheuermann and G. Wu, Heuristic algorithms for broadcasting in point-to-point computer network, *IEEE Transactions on Computers*, C-33(9), 1984, pp. 804-811.

[76] P.J. Slater, E.J. Cockayne and S.T. Heditniemi, Information dissemination in trees, *SIAM J.Comput.*, vol. 10, no. 4, 1981, pp. 692-701.

[77] B. Shao, A Heuristic for broadcasting in arbitrary networks, *Master Thesis of Computer Science*, Concordia University, 2003.

[78] J. Xiao and X. Wang, A research on minimum broadcast graphs, *Chin. J. Comput.*, 11, 1988, pp. 99-105.

[79] X. Xu, Broadcast networks on $2^p - 1$ nodes and minimum broadcast network on 127 nodes, *Master Thesis of Computer Science*, Concordia University, 2003.

[80] J. L. A. Yebra, M. A. Fiol, P. Morillo, and I. Alegre, The diameter of undirected graphs associated to plane tessellations, *Ars Combinatoria 20B*, 20B, 1985, pp. 159-172.

[81] M. Zaragozá, Redes de Interconexión: Contribución al Estudio de su Vulnerabilidad, *PhD thesis*, Departament de Mathemàtica Aplicada i Telemàtica, Universitat Politecnica de Catalunya, Barcelona, Spain, 1994.

[82] E. W. Zegura, K. Calvert, and S. Bhattacharjee, How to model an internetwork, *IEEE INFOCOM*, San Francisco, CA, 1996.

Appendix A

The 1-Broadcast Scheme of 1-*mbg* on 1023 Vertices

The 1-broadcast scheme of 1-*mbg* on 1023 vertices that originated at vertex 0 is presented below.

Round 1: $0 \to 930$;

Round 2: $0 \to 64$; $930 \to 465$;

Round 3: $0 \to 866$; $64 \to 994$; $465 \to 401$; $930 \to 744$;

Round 4: $0 \to 929$; $64 \to 128$; $401 \to 337$; $465 \to 529$; $744 \to 680$; $866 \to 802$; $930 \to 279$; $994 \to 250$;

Round 5: $0 \to 1$; $64 \to 65$; $128 \to 965$; $250 \to 234$; $279 \to 295$; $337 \to 988$; $401 \to 397$; $465 \to 466$; $529 \to 593$; $680 \to 616$; $744 \to 760$; $802 \to 786$; $866 \to 862$; $929 \to 925$; $930 \to 558$; $994 \to 622$;

Round 6: $0 \to 4$; $1 \to 5$; $64 \to 68$; $65 \to 69$; $128 \to 192$; $234 \to 238$; $250 \to 254$;

279 → 280; 295 → 359; 337 → 336; 397 → 396; 401 → 400; 465 → 464; 466 → 470; 529 → 525; 558 → 559; 593 → 657; 616 → 552; 622 → 638; 680 → 679; 744 → 748; 760 → 824; 786 → 972; 802 → 801; 862 → 955; 866 → 882; 925 → 1018; 929 → 928; 930 → 651; 965 → 500; 988 → 151; 994 → 157;

Round 7: 0 → 914; 1 → 915; 4 → 8; 5 → 919; 64 → 60; 65 → 66; 68 → 84; 69 → 999; 128 → 124; 151 → 167; 157 → 161; 192 → 208; 234 → 233; 238 → 242; 250 → 266; 254 → 258; 279 → 263; 280 → 281; 295 → 294; 336 → 335; 337 → 353; 359 → 423; 396 → 412; 397 → 393; 400 → 958; 401 → 405; 464 → 448; 465 → 449; 466 → 450; 470 → 454; 500 → 501; 525 → 521; 529 → 513; 552 → 536; 558 → 562; 559 → 543; 593 → 594; 616 → 612; 622 → 606; 638 → 702; 651 → 667; 657 → 673; 679 → 678; 680 → 664; 744 → 728; 748 → 747; 760 → 761; 786 → 787; 801 → 797; 802 → 806; 824 → 840; 862 → 846; 866 → 850; 882 → 975; 925 → 11; 928 → 1021; 929 → 15; 930 → 837; 955 → 118; 965 → 35; 972 → 42; 988 → 709; 994 → 901; 1018 → 181;

Round 8: 0 → 16; 1 → 17; 4 → 20; 5 → 871; 8 → 874; 11 → 941; 15 → 79; 35 → 34; 42 → 26; 60 → 56; 64 → 63; 65 → 995; 66 → 996; 68 → 132; 69 → 73; 84 → 100; 118 → 114; 124 → 108; 128 → 127; 151 → 155; 157 → 158; 161 → 97; 167 → 163; 181 → 245; 192 → 176; 208 → 204; 233 → 232; 234 → 235; 238 → 302; 242 → 241; 250 → 251; 254 → 253; 258 → 194; 263 → 199; 266 → 202; 279 → 283; 280 → 284; 281 → 345; 294 → 293; 295 → 291; 335 → 351; 336 → 320; 337 → 338; 353 → 369; 359 → 375; 393 → 377; 396 → 380; 397 → 381; 400 → 384; 401 → 402; 405 → 963; 412 → 411; 423 → 427; 448 → 447; 449 → 433; 450 → 451; 454 → 455; 464 →

468; 465 → 481; 466 → 467; 470 → 474; 500 → 499; 501 → 565; 513 → 577; 521 → 585; 525 → 524; 529 → 533; 536 → 1001; 543 → 479; 552 → 568; 558 → 554; 559 → 575; 562 → 566; 593 → 592; 594 → 595; 606 → 602; 612 → 984; 616 → 632; 622 → 621; 638 → 639; 651 → 647; 657 → 641; 664 → 648; 667 → 671; 673 → 689; 678 → 694; 679 → 695; 680 → 681; 702 → 703; 709 → 708; 728 → 712; 744 → 740; 747 → 751; 748 → 752; 760 → 759; 761 → 762; 786 → 770; 787 → 783; 797 → 983; 801 → 800; 802 → 738; 806 → 810; 824 → 888; 837 → 853; 840 → 844; 846 → 830; 850 → 849; 862 → 863; 866 → 867; 882 → 883; 901 → 900; 914 → 898; 915 → 916; 919 → 923; 925 → 921; 928 → 864; 929 → 1022; 930 → 372; 955 → 304; 958 → 307; 965 → 779; 972 → 507; 975 → 138; 988 → 895; 994 → 715; 999 → 720; 1018 → 367; 1021 → 91;

Round 9: 0 → 926; 1 → 2; 4 → 918; 5 → 6; 8 → 24; 11 → 877; 15 → 31; 16 → 32; 17 → 13; 20 → 886; 26 → 22; 34 → 38; 35 → 39; 42 → 41; 56 → 40; 60 → 76; 63 → 47; 64 → 48; 65 → 61; 66 → 82; 68 → 52; 69 → 70; 73 → 89; 79 → 78; 84 → 85; 91 → 87; 97 → 101; 100 → 96; 108 → 109; 114 → 110; 118 → 102; 124 → 125; 127 → 111; 128 → 129; 132 → 136; 138 → 74; 151 → 150; 155 → 139; 157 → 153; 158 → 94; 161 → 145; 163 → 1000; 167 → 103; 176 → 175; 181 → 165; 192 → 196; 194 → 210; 199 → 183; 202 → 201; 204 → 205; 208 → 209; 232 → 248; 233 → 977; 234 → 170; 235 → 979; 238 → 222; 241 → 225; 242 → 226; 245 → 261; 250 → 314; 251 → 187; 253 → 189; 254 → 190; 258 → 322; 263 → 327; 266 → 330; 279 → 275; 280 → 276; 281 → 217; 283 → 287; 284 → 220; 291 → 227; 293 → 309; 294 → 310; 295 → 311; 302 → 953; 304 → 288; 307 → 323; 320 → 971; 335 → 271; 336 → 340; 337 →

273; 338 → 342; 345 → 346; 351 → 350; 353 → 289; 359 → 360; 367 → 366; 369 → 365; 372 → 356; 375 → 374; 377 → 378; 380 → 364; 381 → 382; 384 → 388; 393 → 394; 396 → 954; 397 → 398; 400 → 416; 401 → 385; 402 → 418; 405 → 421; 411 → 395; 412 → 408; 423 → 424; 427 → 428; 433 → 429; 447 → 1005; 448 → 1006; 449 → 445; 450 → 446; 451 → 387; 454 → 390; 455 → 456; 464 → 480; 465 → 461; 466 → 462; 467 → 403; 468 → 484; 470 → 486; 474 → 473; 479 → 478; 481 → 477; 499 → 515; 500 → 496; 501 → 485; 507 → 506; 513 → 509; 521 → 522; 524 → 508; 525 → 526; 529 → 545; 533 → 549; 536 → 535; 543 → 527; 552 → 488; 554 → 550; 558 → 494; 559 → 560; 562 → 546; 565 → 629; 566 → 582; 568 → 572; 575 → 579; 577 → 573; 585 → 957; 592 → 608; 593 → 597; 594 → 590; 595 → 599; 602 → 666; 606 → 610; 612 → 613; 616 → 620; 621 → 605; 622 → 626; 632 → 636; 638 → 634; 639 → 1011; 641 → 1013; 647 → 643; 648 → 644; 651 → 587; 657 → 653; 664 → 660; 667 → 668; 671 → 670; 673 → 669; 678 → 614; 679 → 615; 680 → 684; 681 → 665; 689 → 968; 694 → 710; 695 → 691; 702 → 701; 703 → 707; 708 → 692; 709 → 725; 712 → 713; 715 → 714; 720 → 721; 728 → 729; 738 → 734; 740 → 741; 744 → 745; 747 → 811; 748 → 812; 751 → 750; 752 → 768; 759 → 823; 760 → 764; 761 → 757; 762 → 778; 770 → 774; 779 → 795; 783 → 969; 786 → 790; 787 → 723; 797 → 733; 800 → 796; 801 → 817; 802 → 818; 806 → 822; 810 → 794; 824 → 820; 830 → 829; 837 → 821; 840 → 839; 844 → 860; 846 → 842; 849 → 833; 850 → 854; 853 → 869; 862 → 858; 863 → 859; 864 → 880; 866 → 959; 867 → 868; 871 → 875; 874 → 873; 882 → 881; 883 → 819; 888 → 889; 895 → 29; 898 → 897; 900 → 896; 901 → 885; 914 → 910; 915 → 49; 916 → 852; 919 → 903; 921 → 55; 923 → 907; 925 → 909;

928 → 14; 929 → 3; 930 → 93; 941 → 197; 955 → 211; 958 → 493; 963 → 312; 965 → 221; 972 → 414; 975 → 45; 983 → 425; 984 → 426; 988 → 430; 994 → 436; 995 → 809; 996 → 624; 999 → 627; 1001 → 71; 1018 → 832; 1021 → 184; 1022 → 185;

Round 10: 1 → 927; 2 → 18; 3 → 67; 4 → 934; 5 → 935; 6 → 920; 8 → 72; 11 → 75; 13 → 12; 14 → 944; 15 → 945; 16 → 946; 17 → 21; 20 → 19; 22 → 86; 24 → 88; 26 → 25; 29 → 28; 31 → 95; 32 → 36; 34 → 30; 35 → 99; 38 → 904; 39 → 905; 40 → 906; 41 → 105; 42 → 58; 45 → 911; 47 → 51; 48 → 978; 49 → 33; 52 → 982; 55 → 119; 56 → 986; 60 → 990; 61 → 57; 63 → 993; 64 → 80; 65 → 81; 66 → 130; 68 → 998; 69 → 133; 70 → 54; 71 → 135; 73 → 1003; 74 → 10; 76 → 77; 78 → 1008; 79 → 143; 82 → 1012; 84 → 1014; 85 → 149; 87 → 83; 89 → 1019; 91 → 107; 93 → 92; 94 → 90; 96 → 160; 97 → 98; 100 → 937; 101 → 117; 102 → 106; 103 → 104; 108 → 44; 109 → 173; 110 → 46; 111 → 112; 114 → 115; 118 → 122; 124 → 123; 125 → 141; 127 → 131; 128 → 144; 129 → 113; 132 → 148; 136 → 120; 138 → 142; 139 → 976; 145 → 146; 150 → 987; 151 → 152; 153 → 137; 155 → 171; 157 → 156; 158 → 162; 161 → 177; 163 → 164; 165 → 1002; 167 → 166; 170 → 154; 175 → 191; 176 → 172; 181 → 182; 183 → 179; 184 → 180; 185 → 169; 187 → 203; 189 → 188; 190 → 174; 192 → 936; 194 → 938; 196 → 260; 197 → 198; 199 → 200; 201 → 265; 202 → 218; 204 → 140; 205 → 949; 208 → 952; 209 → 193; 210 → 214; 211 → 195; 217 → 213; 220 → 236; 221 → 237; 222 → 206; 225 → 229; 226 → 970; 227 → 243; 232 → 168; 233 → 297; 234 → 230; 235 → 219; 238 → 239; 241 → 257; 242 → 178; 245 → 989; 248 → 244; 250 → 249; 251 → 252; 253 → 997; 254 → 270; 258 → 262; 261 → 325; 263 → 264; 266 → 1010; 271 → 207; 273 → 269; 275 → 259; 276 → 212; 279 →

278; 280 → 216; 281 → 282; 283 → 347; 284 → 268; 287 → 223; 288 → 224; 289 → 285; 291 → 290; 293 → 277; 294 → 358; 295 → 299; 302 → 298; 304 → 240; 307 → 303; 309 → 305; 310 → 246; 311 → 247; 312 → 296; 314 → 315; 320 → 324; 322 → 306; 323 → 974; 327 → 331; 330 → 326; 335 → 319; 336 → 272; 337 → 321; 338 → 274; 340 → 344; 342 → 406; 345 → 349; 346 → 410; 350 → 286; 351 → 355; 353 → 354; 356 → 292; 359 → 363; 360 → 376; 364 → 300; 365 → 301; 366 → 362; 367 → 383; 369 → 373; 372 → 308; 374 → 438; 375 → 439; 377 → 313; 378 → 379; 380 → 316; 381 → 317; 382 → 940; 384 → 368; 385 → 389; 387 → 371; 388 → 404; 390 → 391; 393 → 329; 394 → 458; 395 → 459; 396 → 332; 397 → 333; 398 → 334; 400 → 399; 401 → 417; 402 → 386; 403 → 339; 405 → 341; 408 → 409; 411 → 475; 412 → 348; 414 → 415; 416 → 352; 418 → 422; 421 → 357; 423 → 487; 424 → 440; 425 → 361; 426 → 490; 427 → 985; 428 → 432; 429 → 413; 430 → 431; 433 → 434; 436 → 437; 445 → 441; 446 → 442; 447 → 511; 448 → 452; 449 → 453; 450 → 514; 451 → 435; 454 → 518; 455 → 519; 456 → 392; 461 → 457; 462 → 463; 464 → 460; 465 → 469; 466 → 530; 467 → 471; 468 → 532; 470 → 534; 473 → 537; 474 → 939; 477 → 942; 478 → 482; 479 → 483; 480 → 476; 481 → 497; 484 → 420; 485 → 489; 486 → 502; 488 → 504; 493 → 492; 494 → 498; 496 → 961; 499 → 964; 500 → 516; 501 → 966; 506 → 570; 507 → 491; 508 → 444; 509 → 505; 513 → 512; 515 → 531; 521 → 520; 522 → 538; 524 → 540; 525 → 541; 526 → 510; 527 → 591; 529 → 528; 533 → 517; 535 → 539; 536 → 472; 543 → 607; 545 → 544; 546 → 547; 549 → 553; 550 → 551; 552 → 556; 554 → 555; 558 → 574; 559 → 495; 560 → 564; 562 → 578; 565 → 581; 566 → 567; 568 → 584; 572 → 588; 573 → 569; 575 → 576; 577 → 561; 579 →

580; 582 → 586; 585 → 649; 587 → 571; 590 → 962; 592 → 596; 593 → 589; 594 → 658; 595 → 967; 597 → 661; 599 → 603; 602 → 618; 605 → 604; 606 → 542; 608 → 672; 610 → 674; 612 → 628; 613 → 677; 614 → 598; 615 → 611; 616 → 617; 620 → 619; 621 → 557; 622 → 686; 624 → 623; 626 → 690; 627 → 563; 629 → 633; 632 → 631; 634 → 698; 636 → 700; 638 → 642; 639 → 655; 641 → 637; 643 → 659; 644 → 640; 647 → 711; 648 → 652; 651 → 635; 653 → 717; 657 → 656; 660 → 724; 664 → 600; 665 → 601; 666 → 730; 667 → 731; 668 → 947; 669 → 685; 670 → 654; 671 → 950; 673 → 609; 678 → 742; 679 → 663; 680 → 676; 681 → 682; 684 → 683; 689 → 625; 691 → 755; 692 → 693; 694 → 630; 695 → 699; 701 → 765; 702 → 766; 703 → 687; 707 → 771; 708 → 772; 709 → 645; 710 → 706; 712 → 776; 713 → 777; 714 → 718; 715 → 719; 720 → 784; 721 → 705; 723 → 739; 725 → 1004; 728 → 792; 729 → 793; 733 → 749; 734 → 735; 738 → 1017; 740 → 804; 741 → 1020; 744 → 743; 745 → 931; 747 → 933; 748 → 732; 750 → 814; 751 → 815; 752 → 688; 757 → 943; 759 → 775; 760 → 696; 761 → 697; 762 → 948; 764 → 828; 768 → 767; 770 → 956; 774 → 773; 778 → 782; 779 → 763; 783 → 847; 786 → 722; 787 → 973; 790 → 791; 794 → 980; 795 → 981; 796 → 780; 797 → 781; 800 → 736; 801 → 737; 802 → 803; 806 → 992; 809 → 813; 810 → 746; 811 → 827; 812 → 876; 817 → 753; 818 → 754; 819 → 835; 820 → 756; 821 → 825; 822 → 758; 823 → 1009; 824 → 808; 829 → 1015; 830 → 1016; 832 → 831; 833 → 769; 837 → 841; 839 → 932; 840 → 856; 842 → 826; 844 → 908; 846 → 845; 849 → 785; 850 → 834; 852 → 788; 853 → 789; 854 → 838; 858 → 951; 859 → 855; 860 → 924; 862 → 798; 863 → 799; 864 → 848; 866 → 870; 867 → 960; 868 → 872; 869 → 805; 871 → 807; 873 → 7; 874 → 890; 875 → 891; 877

$\rightarrow$ 861; 880 $\rightarrow$ 816; 881 $\rightarrow$ 865; 882 $\rightarrow$ 878; 883 $\rightarrow$ 879; 885 $\rightarrow$ 884; 886 $\rightarrow$ 887; 888 $\rightarrow$ 892; 889 $\rightarrow$ 23; 895 $\rightarrow$ 899; 896 $\rightarrow$ 912; 897 $\rightarrow$ 893; 898 $\rightarrow$ 991; 900 $\rightarrow$ 836; 901 $\rightarrow$ 917; 903 $\rightarrow$ 37; 907 $\rightarrow$ 843; 909 $\rightarrow$ 43; 910 $\rightarrow$ 894; 914 $\rightarrow$ 1007; 915 $\rightarrow$ 851; 916 $\rightarrow$ 50; 918 $\rightarrow$ 902; 919 $\rightarrow$ 53; 921 $\rightarrow$ 857; 923 $\rightarrow$ 9; 925 $\rightarrow$ 59; 926 $\rightarrow$ 922; 928 $\rightarrow$ 62; 929 $\rightarrow$ 913; 930 $\rightarrow$ 186; 941 $\rightarrow$ 662; 953 $\rightarrow$ 116; 954 $\rightarrow$ 675; 955 $\rightarrow$ 583; 957 $\rightarrow$ 27; 958 $\rightarrow$ 121; 959 $\rightarrow$ 215; 963 $\rightarrow$ 126; 965 $\rightarrow$ 407; 968 $\rightarrow$ 503; 969 $\rightarrow$ 318; 971 $\rightarrow$ 134; 972 $\rightarrow$ 228; 975 $\rightarrow$ 231; 977 $\rightarrow$ 419; 979 $\rightarrow$ 328; 983 $\rightarrow$ 704; 984 $\rightarrow$ 147; 988 $\rightarrow$ 523; 994 $\rightarrow$ 343; 995 $\rightarrow$ 716; 996 $\rightarrow$ 159; 999 $\rightarrow$ 255; 1000 $\rightarrow$ 256; 1001 $\rightarrow$ 443; 1005 $\rightarrow$ 726; 1006 $\rightarrow$ 727; 1011 $\rightarrow$ 267; 1013 $\rightarrow$ 548; 1018 $\rightarrow$ 646; 1021 $\rightarrow$ 370; 1022 $\rightarrow$ 650;

Appendix B

The 1-Broadcast Scheme of 1-*mbg* on 4095 Vertices

The 1-broadcast scheme of 1-*mbg* on 4095 vertices that originated at 0 is presented below.

Round 1: $0 \to 3780$;

Round 2: $0 \to 16$; $3780 \to 1890$;

Round 3: $0 \to 3764$; $16 \to 3796$; $1890 \to 1874$; $3780 \to 1260$;

Round 4: $0 \to 3524$; $16 \to 272$; $1260 \to 1244$; $1874 \to 4079$; $1890 \to 2146$; $3764 \to 3508$; $3780 \to 2835$; $3796 \to 2536$;

Round 5: $0 \to 64$; $16 \to 80$; $272 \to 528$; $1244 \to 1228$; $1260 \to 1004$; $1874 \to 1858$; $1890 \to 1634$; $2146 \to 2402$; $2536 \to 2552$; $2835 \to 2831$; $3508 \to 3252$; $3524 \to 3268$; $3764 \to 240$; $3780 \to 630$; $3796 \to 646$; $4079 \to 614$;

Round 6: $0 \to 3716$; $16 \to 15$; $64 \to 128$; $80 \to 3860$; $240 \to 4020$; $272 \to 4052$;

528 → 3993; 614 → 870; 630 → 634; 646 → 710; 1004 → 1068; 1228 → 1484; 1244 → 1180; 1260 → 1516; 1634 → 1378; 1858 → 1794; 1874 → 1810; 1890 → 1954; 2146 → 2082; 2402 → 3977; 2536 → 2537; 2552 → 2296; 2831 → 2827; 2835 → 2839; 3252 → 3882; 3268 → 3898; 3508 → 3507; 3524 → 3523; 3764 → 3700; 3780 → 3150; 3796 → 3166; 4079 → 2189;

Round 7: 0 → 1; 15 → 14; 16 → 17; 64 → 65; 80 → 84; 128 → 3908; 240 → 239; 272 → 273; 528 → 532; 614 → 615; 630 → 631; 634 → 633; 646 → 647; 710 → 714; 870 → 866; 1004 → 1000; 1068 → 1069; 1180 → 1176; 1228 → 1227; 1244 → 1240; 1260 → 1264; 1378 → 1374; 1484 → 1483; 1516 → 1517; 1634 → 1630; 1794 → 3999; 1810 → 1746; 1858 → 1857; 1874 → 1875; 1890 → 1894; 1954 → 1955; 2082 → 2081; 2146 → 2147; 2189 → 2188; 2296 → 3871; 2402 → 2406; 2536 → 2540; 2537 → 2538; 2552 → 2616; 2827 → 2763; 2831 → 2832; 2835 → 2834; 2839 → 2838; 3150 → 3149; 3166 → 3165; 3252 → 3253; 3268 → 3269; 3507 → 3822; 3508 → 3504; 3523 → 3838; 3524 → 3520; 3700 → 3696; 3716 → 3715; 3764 → 3765; 3780 → 945; 3796 → 3481; 3860 → 1655; 3882 → 417; 3898 → 2953; 3977 → 2717; 3993 → 3363; 4020 → 3390; 4052 → 2477; 4079 → 1559;

Round 8: 0 → 4; 1 → 257; 14 → 3794; 15 → 19; 16 → 12; 17 → 13; 64 → 68; 65 → 66; 80 → 76; 84 → 88; 128 → 132; 239 → 238; 240 → 244; 272 → 268; 273 → 269; 417 → 413; 528 → 524; 532 → 3997; 614 → 358; 615 → 611; 630 → 566; 631 → 627; 633 → 569; 634 → 638; 646 → 582; 647 → 643; 710 → 706; 714 → 718; 866 → 867; 870 → 869; 945 → 946; 1000 → 996; 1004 → 1020; 1068 → 1072; 1069 → 1070; 1176 → 1172; 1180 → 1179; 1227 → 1231; 1228 → 1229; 1240 → 1236; 1244 → 1248;

1260 → 1259; 1264 → 1268; 1374 → 3894; 1378 → 1379; 1483 → 1739; 1484 → 1480; 1516 → 1520; 1517 → 1513; 1559 → 1560; 1630 → 1626; 1634 → 1633; 1655 → 1656; 1746 → 1745; 1794 → 1790; 1810 → 1809; 1857 → 1861; 1858 → 1854; 1874 → 1870; 1875 → 1876; 1890 → 1889; 1894 → 1898; 1954 → 1958; 1955 → 1951; 2081 → 2080; 2082 → 2078; 2146 → 2150; 2147 → 2143; 2188 → 2187; 2189 → 2185; 2296 → 2297; 2402 → 2398; 2406 → 3981; 2477 → 2481; 2536 → 2532; 2537 → 2533; 2538 → 3798; 2540 → 2544; 2552 → 2548; 2616 → 2612; 2717 → 2721; 2763 → 2747; 2827 → 3083; 2831 → 2575; 2832 → 4092; 2834 → 2770; 2835 → 2771; 2838 → 3094; 2839 → 2903; 2953 → 2949; 3149 → 3148; 3150 → 3146; 3165 → 3164; 3166 → 3162; 3252 → 2996; 3253 → 2997; 3268 → 3272; 3269 → 3285; 3363 → 3367; 3390 → 3386; 3481 → 3482; 3504 → 3819; 3507 → 3511; 3508 → 3512; 3520 → 3516; 3523 → 3527; 3524 → 3528; 3696 → 3692; 3700 → 3704; 3715 → 4030; 3716 → 3720; 3764 → 3768; 3765 → 3766; 3780 → 315; 3796 → 1591; 3822 → 3192; 3838 → 2263; 3860 → 1025; 3871 → 1351; 3882 → 2307; 3898 → 118; 3908 → 2963; 3977 → 3662; 3993 → 3048; 3999 → 3054; 4020 → 3705; 4052 → 1217; 4079 → 299;

Round 9: 0 → 3776; 1 → 3717; 4 → 5; 12 → 3728; 13 → 29; 14 → 30; 15 → 31; 16 → 3732; 17 → 33; 19 → 3735; 64 → 320; 65 → 61; 66 → 70; 68 → 3848; 76 → 3600; 80 → 3604; 84 → 100; 88 → 89; 118 → 182; 128 → 384; 132 → 3912; 238 → 494; 239 → 495; 240 → 496; 244 → 245; 257 → 513; 268 → 204; 269 → 525; 272 → 288; 273 → 289; 299 → 363; 315 → 251; 358 → 362; 413 → 409; 417 → 481; 524 → 780; 528 → 592; 532 → 468; 566 → 502; 569 → 825; 582 → 838; 611 → 607; 614 → 678; 615 → 679; 627 → 691; 630 → 629; 631 → 695; 633 → 889; 634 → 698; 638 →

894; 643 → 899; 646 → 645; 647 → 663; 706 → 770; 710 → 726; 714 → 730; 718 → 782; 866 → 862; 867 → 803; 869 → 1125; 870 → 1126; 945 → 881; 946 → 1202; 996 → 740; 1000 → 744; 1004 → 748; 1020 → 956; 1025 → 1029; 1068 → 812; 1069 → 1065; 1070 → 3905; 1072 → 1088; 1172 → 1428; 1176 → 1112; 1179 → 923; 1180 → 1116; 1217 → 1153; 1227 → 1211; 1228 → 972; 1229 → 1165; 1231 → 1167; 1236 → 1237; 1240 → 1304; 1244 → 1308; 1248 → 1249; 1259 → 1258; 1260 → 1261; 1264 → 1328; 1268 → 1269; 1351 → 1347; 1374 → 1390; 1378 → 1394; 1379 → 1443; 1480 → 1476; 1483 → 1467; 1484 → 1548; 1513 → 1449; 1516 → 1772; 1517 → 1773; 1520 → 1456; 1559 → 1815; 1560 → 1816; 1591 → 1527; 1626 → 1370; 1630 → 1646; 1633 → 1632; 1634 → 1570; 1655 → 1399; 1656 → 1660; 1739 → 3944; 1745 → 3950; 1746 → 3951; 1790 → 3995; 1794 → 1778; 1809 → 2065; 1810 → 2066; 1854 → 4059; 1857 → 1856; 1858 → 2114; 1861 → 2117; 1870 → 1866; 1874 → 1938; 1875 → 1939; 1876 → 2132; 1889 → 1885; 1890 → 1891; 1894 → 1830; 1898 → 1897; 1951 → 1967; 1954 → 1970; 1955 → 1971; 1958 → 1974; 2078 → 2074; 2080 → 2096; 2081 → 2085; 2082 → 3972; 2143 → 2399; 2146 → 2210; 2147 → 2163; 2150 → 2166; 2185 → 2181; 2187 → 4077; 2188 → 2172; 2189 → 2253; 2263 → 2262; 2296 → 2040; 2297 → 2298; 2307 → 2371; 2398 → 2414; 2402 → 2386; 2406 → 2390; 2477 → 2493; 2481 → 2497; 2532 → 2468; 2533 → 2277; 2536 → 2472; 2537 → 2601; 2538 → 2602; 2540 → 2604; 2544 → 2608; 2548 → 3808; 2552 → 2556; 2575 → 2574; 2612 → 2628; 2616 → 2620; 2717 → 2701; 2721 → 2720; 2747 → 2683; 2763 → 2699; 2770 → 2766; 2771 → 2707; 2827 → 4087; 2831 → 2847; 2832 → 3088; 2834 → 2850; 2835 → 2836; 2838 → 2582; 2839 → 2840; 2903 → 2647; 2949 → 2693; 2953 → 2889; 2963 → 2947; 2996 → 2992; 2997 →

3942; 3048 → 2984; 3054 → 3310; 3083 → 3339; 3094 → 3350; 3146 → 3142; 3148 → 3152; 3149 → 3153; 3150 → 3214; 3162 → 3178; 3164 → 3228; 3165 → 3101; 3166 → 3182; 3192 → 3191; 3252 → 3236; 3253 → 3249; 3268 → 3204; 3269 → 3333; 3272 → 3276; 3285 → 3349; 3363 → 3362; 3367 → 3303; 3386 → 3322; 3390 → 3326; 3481 → 3417; 3482 → 3546; 3504 → 3488; 3507 → 3506; 3508 → 3492; 3511 → 3255; 3512 → 3827; 3516 → 3580; 3520 → 3456; 3523 → 3522; 3524 → 3460; 3527 → 3531; 3528 → 3532; 3662 → 3663; 3692 → 168; 3696 → 3632; 3700 → 3684; 3704 → 180; 3705 → 3689; 3715 → 3711; 3716 → 3652; 3720 → 3724; 3764 → 48; 3765 → 49; 3766 → 3750; 3768 → 52; 3780 → 1575; 3794 → 1274; 3796 → 2221; 3798 → 1593; 3819 → 2244; 3822 → 2877; 3838 → 1948; 3860 → 1340; 3871 → 2926; 3882 → 2622; 3894 → 2634; 3898 → 2008; 3908 → 3278; 3977 → 1142; 3981 → 1461; 3993 → 843; 3997 → 2107; 3999 → 219; 4020 → 3075; 4030 → 880; 4052 → 1847; 4079 → 2504; 4092 → 2517;

Round 10: 0 → 3779; 1 → 3777; 4 → 260; 5 → 6; 12 → 3792; 13 → 3793; 14 → 78; 15 → 79; 16 → 3540; 17 → 81; 19 → 3799; 29 → 3553; 30 → 94; 31 → 95; 33 → 37; 48 → 44; 49 → 113; 52 → 56; 61 → 125; 64 → 63; 65 → 129; 66 → 130; 68 → 324; 70 → 3850; 76 → 140; 80 → 144; 84 → 83; 88 → 344; 89 → 3613; 100 → 3880; 118 → 119; 128 → 127; 132 → 131; 168 → 232; 180 → 179; 182 → 198; 204 → 460; 219 → 218; 238 → 174; 239 → 175; 240 → 236; 244 → 228; 245 → 249; 251 → 247; 257 → 258; 268 → 264; 269 → 205; 272 → 271; 273 → 274; 288 → 292; 289 → 4069; 299 → 295; 315 → 319; 320 → 576; 358 → 294; 362 → 366; 363 → 364; 384 → 380; 409 → 3874; 413 → 412; 417 → 418; 468 → 467; 481 → 482; 494 → 3959; 495 →

751; 496 → 432; 502 → 503; 513 → 449; 524 → 520; 525 → 3990; 528 → 784; 532 → 536; 566 → 562; 569 → 553; 582 → 586; 592 → 656; 607 → 606; 611 → 355; 614 → 550; 615 → 616; 627 → 371; 629 → 625; 630 → 626; 631 → 375; 633 → 377; 634 → 635; 638 → 382; 643 → 387; 645 → 661; 646 → 390; 647 → 391; 663 → 407; 678 → 934; 679 → 423; 691 → 435; 695 → 759; 698 → 442; 706 → 450; 710 → 711; 714 → 458; 718 → 462; 726 → 982; 730 → 734; 740 → 3890; 744 → 488; 748 → 684; 770 → 771; 780 → 776; 782 → 1038; 803 → 799; 812 → 556; 825 → 841; 838 → 839; 843 → 1099; 862 → 926; 866 → 850; 867 → 4017; 869 → 853; 870 → 874; 880 → 879; 881 → 817; 889 → 905; 894 → 910; 899 → 963; 923 → 922; 945 → 689; 946 → 1010; 956 → 700; 972 → 968; 996 → 995; 1000 → 999; 1004 → 1005; 1020 → 1019; 1025 → 1041; 1029 → 1033; 1065 → 1049; 1068 → 3903; 1069 → 1053; 1070 → 1086; 1072 → 1076; 1088 → 3923; 1112 → 856; 1116 → 860; 1125 → 1381; 1126 → 1110; 1142 → 1078; 1153 → 1409; 1165 → 1421; 1167 → 4002; 1172 → 4007; 1176 → 4011; 1179 → 1178; 1180 → 924; 1202 → 1218; 1211 → 1147; 1217 → 1281; 1227 → 1223; 1228 → 4063; 1229 → 1293; 1231 → 1295; 1236 → 1235; 1237 → 1238; 1240 → 1239; 1244 → 1245; 1248 → 1247; 1249 → 1250; 1258 → 1002; 1259 → 1263; 1260 → 1196; 1261 → 1197; 1264 → 1280; 1268 → 1204; 1269 → 1285; 1274 → 1290; 1304 → 1300; 1308 → 1312; 1328 → 1584; 1340 → 1341; 1347 → 1343; 1351 → 1350; 1370 → 1354; 1374 → 1358; 1378 → 1362; 1379 → 1380; 1390 → 1406; 1394 → 1398; 1399 → 1143; 1428 → 1684; 1443 → 1507; 1449 → 3969; 1456 → 1712; 1461 → 1717; 1467 → 1471; 1476 → 1732; 1480 → 1481; 1483 → 1482; 1484 → 1485; 1513 → 1769; 1516 → 1580; 1517 → 1501; 1520 → 1519; 1527 → 1531; 1548 → 1612; 1559 → 1558; 1560 → 1561; 1570 → 1569;

1575 → 1579; 1591 → 1595; 1593 → 1609; 1626 → 1690; 1630 → 1566; 1632 → 1616;

1633 → 1649; 1634 → 1635; 1646 → 1662; 1655 → 1671; 1656 → 1672; 1660 → 1664;

1739 → 1995; 1745 → 1729; 1746 → 1490; 1772 → 1771; 1773 → 2029; 1778 → 1777;

1790 → 1726; 1794 → 1538; 1809 → 4014; 1810 → 1554; 1815 → 2071; 1816 → 1800;

1830 → 1766; 1847 → 1863; 1854 → 1855; 1856 → 4061; 1857 → 1601; 1858 → 1862;

1861 → 1605; 1866 → 1930; 1870 → 1614; 1874 → 1618; 1875 → 1871; 1876 → 1860;

1885 → 1884; 1889 → 1905; 1890 → 1906; 1891 → 1827; 1894 → 1910; 1897 → 1896;

1898 → 1899; 1938 → 1682; 1939 → 1935; 1948 → 1964; 1951 → 2015; 1954 → 1950;

1955 → 2019; 1958 → 1702; 1967 → 1983; 1970 → 1969; 1971 → 1987; 1974 → 1978;

2008 → 2009; 2040 → 2041; 2065 → 2049; 2066 → 2067; 2074 → 2010; 2078 → 2014;

2080 → 3970; 2081 → 2097; 2082 → 2338; 2085 → 2021; 2096 → 3986; 2107 → 2043;

2114 → 2178; 2117 → 2053; 2132 → 2136; 2143 → 2139; 2146 → 2145; 2147 → 2151;

2150 → 4040; 2163 → 2419; 2166 → 2422; 2172 → 4062; 2181 → 1925; 2185 → 2441;

2187 → 2251; 2188 → 2192; 2189 → 2190; 2210 → 2274; 2221 → 2222; 2244 → 2240;

2253 → 2254; 2262 → 2246; 2263 → 2279; 2277 → 2293; 2296 → 2295; 2297 → 2361;

2298 → 3873; 2307 → 2303; 2371 → 2435; 2386 → 2385; 2390 → 2374; 2398 → 2462;

2399 → 2395; 2402 → 2658; 2406 → 2662; 2414 → 2430; 2468 → 2452; 2472 → 2408;

2477 → 2733; 2481 → 2482; 2493 → 2489; 2497 → 2496; 2504 → 2503; 2517 → 2521;

2532 → 2531; 2533 → 2789; 2536 → 2535; 2537 → 2793; 2538 → 2794; 2540 → 2524;

2544 → 2288; 2548 → 2547; 2552 → 2551; 2556 → 3816; 2574 → 2573; 2575 → 2511;

2582 → 2566; 2601 → 2665; 2602 → 3862; 2604 → 2348; 2608 → 2609; 2612 → 2613;

2616 → 2615; 2620 → 2364; 2622 → 2626; 2628 → 2624; 2634 → 2378; 2647 → 2663;

2683 → 2939; 2693 → 2694; 2699 → 2443; 2701 → 2637; 2707 → 2723; 2717 → 2713;

2720 → 2976; 2721 → 2977; 2747 → 2746; 2763 → 2779; 2766 → 2762; 2770 → 3026;

2771 → 2787; 2827 → 2811; 2831 → 2815; 2832 → 2576; 2834 → 2818; 2835 → 2851;

2836 → 2852; 2838 → 2822; 2839 → 2855; 2840 → 3096; 2847 → 3103; 2850 → 2866;

2877 → 2813; 2889 → 2825; 2903 → 2967; 2926 → 2930; 2947 → 2943; 2949 → 2933;

2953 → 3017; 2963 → 2979; 2984 → 2983; 2992 → 2991; 2996 → 3941; 2997 → 2998;

3048 → 3049; 3054 → 3058; 3075 → 3059; 3083 → 3067; 3088 → 3104; 3094 → 3030;

3101 → 3037; 3142 → 3126; 3146 → 3082; 3148 → 3132; 3149 → 3085; 3150 → 3151;

3152 → 3136; 3153 → 3137; 3162 → 3226; 3164 → 2908; 3165 → 3421; 3166 → 3167;

3178 → 3179; 3182 → 3183; 3191 → 3127; 3192 → 3128; 3204 → 3200; 3214 → 3215;

3228 → 3292; 3236 → 3172; 3249 → 3879; 3252 → 3188; 3253 → 3883; 3255 → 3885;

3268 → 3267; 3269 → 3013; 3272 → 3336; 3276 → 3906; 3278 → 3279; 3285 → 3301;

3303 → 3047; 3310 → 3309; 3322 → 3066; 3326 → 3325; 3333 → 3329; 3339 → 3355;

3349 → 3979; 3350 → 3606; 3362 → 3361; 3363 → 3379; 3367 → 3371; 3386 → 3382;

3390 → 3389; 3417 → 3673; 3456 → 3457; 3460 → 3396; 3481 → 3485; 3482 → 3738;

3488 → 3487; 3492 → 3428; 3504 → 3505; 3506 → 3570; 3507 → 3503; 3508 → 3444;

3511 → 3575; 3512 → 3576; 3516 → 3517; 3520 → 3835; 3522 → 3266; 3523 → 3539;

3524 → 3525; 3527 → 3591; 3528 → 3544; 3531 → 3547; 3532 → 3847; 3546 → 3610;

3580 → 3564; 3600 → 3915; 3604 → 3603; 3632 → 3376; 3652 → 3653; 3662 → 138;

3663 → 3667; 3684 → 3683; 3689 → 3693; 3692 → 3691; 3696 → 3697; 3700 → 3701;

3704 → 3640; 3705 → 3709; 3711 → 3455; 3715 → 3714; 3716 → 3712; 3717 → 3718;

3720 → 3719; 3724 → 3740; 3728 → 4043; 3732 → 3668; 3735 → 3671; 3750 → 4065;

3764 → 3763; 3765 → 3761; 3766 → 4081; 3768 → 3752; 3776 → 3775; 3780 → 3465; 3794 → 2219; 3796 → 331; 3798 → 2853; 3808 → 658; 3819 → 669; 3822 → 1302; 3827 → 2252; 3838 → 688; 3848 → 3218; 3860 → 395; 3871 → 1981; 3882 → 1047; 3894 → 429; 3898 → 433; 3905 → 1700; 3908 → 2333; 3912 → 447; 3942 → 1737; 3944 → 794; 3950 → 1430; 3951 → 2376; 3972 → 2397; 3977 → 2087; 3981 → 3666; 3993 → 3678; 3995 → 215; 3997 → 217; 3999 → 534; 4020 → 2760; 4030 → 3400; 4052 → 3107; 4059 → 279; 4077 → 297; 4079 → 929; 4087 → 1567; 4092 → 312;

Round 11: 0 → 256; 1 → 2; 4 → 8; 5 → 9; 6 → 3530; 12 → 28; 13 → 3537; 14 → 10; 15 → 11; 16 → 20; 17 → 21; 19 → 35; 29 → 25; 30 → 26; 31 → 27; 33 → 97; 37 → 3561; 44 → 3824; 48 → 3572; 49 → 3573; 52 → 308; 56 → 3836; 61 → 317; 63 → 59; 64 → 3844; 65 → 321; 66 → 3846; 68 → 69; 70 → 71; 76 → 92; 78 → 3858; 79 → 75; 80 → 96; 81 → 337; 83 → 87; 84 → 85; 88 → 3612; 89 → 73; 94 → 98; 95 → 91; 100 → 101; 113 → 3893; 118 → 54; 119 → 3643; 125 → 109; 127 → 3907; 128 → 112; 129 → 133; 130 → 146; 131 → 147; 132 → 136; 138 → 137; 140 → 3920; 144 → 160; 168 → 152; 174 → 158; 175 → 171; 179 → 163; 180 → 184; 182 → 166; 198 → 262; 204 → 3984; 205 → 3985; 215 → 214; 217 → 473; 218 → 282; 219 → 475; 228 → 212; 232 → 216; 236 → 220; 238 → 254; 239 → 223; 240 → 224; 244 → 500; 245 → 309; 247 → 3771; 249 → 265; 251 → 250; 257 → 193; 258 → 4038; 260 → 259; 264 → 4044; 268 → 267; 269 → 270; 271 → 207; 272 → 208; 273 → 277; 274 → 338; 279 → 535; 288 → 352; 289 → 225; 292 → 548; 294 → 230; 295 → 39; 297 → 41; 299 → 43; 312 → 568; 315 → 314; 319 → 318; 320 → 304; 324 → 325; 331 → 330; 344 → 345; 355 → 339; 358 → 342; 362 → 106; 363 → 107; 364 → 3829; 366 →

302; 371 $\rightarrow$ 372; 375 $\rightarrow$ 376; 377 $\rightarrow$ 121; 380 $\rightarrow$ 444; 382 $\rightarrow$ 398; 384 $\rightarrow$ 640; 387 $\rightarrow$ 3852; 390 $\rightarrow$ 406; 391 $\rightarrow$ 392; 395 $\rightarrow$ 399; 407 $\rightarrow$ 151; 409 $\rightarrow$ 410; 412 $\rightarrow$ 156; 413 $\rightarrow$ 157; 417 $\rightarrow$ 401; 418 $\rightarrow$ 162; 423 $\rightarrow$ 427; 429 $\rightarrow$ 425; 432 $\rightarrow$ 416; 433 $\rightarrow$ 437; 435 $\rightarrow$ 3900; 442 $\rightarrow$ 186; 447 $\rightarrow$ 703; 449 $\rightarrow$ 453; 450 $\rightarrow$ 451; 458 $\rightarrow$ 457; 460 $\rightarrow$ 456; 462 $\rightarrow$ 3927; 467 $\rightarrow$ 483; 468 $\rightarrow$ 452; 481 $\rightarrow$ 737; 482 $\rightarrow$ 226; 488 $\rightarrow$ 489; 494 $\rightarrow$ 490; 495 $\rightarrow$ 491; 496 $\rightarrow$ 560; 502 $\rightarrow$ 501; 503 $\rightarrow$ 519; 513 $\rightarrow$ 517; 520 $\rightarrow$ 584; 524 $\rightarrow$ 523; 525 $\rightarrow$ 541; 528 $\rightarrow$ 527; 532 $\rightarrow$ 788; 534 $\rightarrow$ 538; 536 $\rightarrow$ 540; 550 $\rightarrow$ 554; 553 $\rightarrow$ 557; 556 $\rightarrow$ 572; 562 $\rightarrow$ 306; 566 $\rightarrow$ 565; 569 $\rightarrow$ 573; 576 $\rightarrow$ 580; 582 $\rightarrow$ 581; 586 $\rightarrow$ 522; 592 $\rightarrow$ 593; 606 $\rightarrow$ 602; 607 $\rightarrow$ 603; 611 $\rightarrow$ 595; 614 $\rightarrow$ 613; 615 $\rightarrow$ 599; 616 $\rightarrow$ 617; 625 $\rightarrow$ 609; 626 $\rightarrow$ 622; 627 $\rightarrow$ 563; 629 $\rightarrow$ 628; 630 $\rightarrow$ 886; 631 $\rightarrow$ 887; 633 $\rightarrow$ 697; 634 $\rightarrow$ 570; 635 $\rightarrow$ 571; 638 $\rightarrow$ 574; 643 $\rightarrow$ 579; 645 $\rightarrow$ 644; 646 $\rightarrow$ 650; 647 $\rightarrow$ 648; 656 $\rightarrow$ 652; 658 $\rightarrow$ 914; 661 $\rightarrow$ 405; 663 $\rightarrow$ 664; 669 $\rightarrow$ 653; 678 $\rightarrow$ 682; 679 $\rightarrow$ 680; 684 $\rightarrow$ 685; 688 $\rightarrow$ 704; 689 $\rightarrow$ 693; 691 $\rightarrow$ 692; 695 $\rightarrow$ 951; 698 $\rightarrow$ 954; 700 $\rightarrow$ 701; 706 $\rightarrow$ 702; 710 $\rightarrow$ 709; 711 $\rightarrow$ 455; 714 $\rightarrow$ 713; 718 $\rightarrow$ 717; 726 $\rightarrow$ 470; 730 $\rightarrow$ 731; 734 $\rightarrow$ 750; 740 $\rightarrow$ 741; 744 $\rightarrow$ 745; 748 $\rightarrow$ 747; 751 $\rightarrow$ 3901; 759 $\rightarrow$ 758; 770 $\rightarrow$ 754; 771 $\rightarrow$ 515; 776 $\rightarrow$ 772; 780 $\rightarrow$ 3930; 782 $\rightarrow$ 766; 784 $\rightarrow$ 768; 794 $\rightarrow$ 795; 799 $\rightarrow$ 543; 803 $\rightarrow$ 807; 812 $\rightarrow$ 811; 817 $\rightarrow$ 833; 825 $\rightarrow$ 761; 838 $\rightarrow$ 774; 839 $\rightarrow$ 823; 841 $\rightarrow$ 585; 843 $\rightarrow$ 779; 850 $\rightarrow$ 851; 853 $\rightarrow$ 789; 856 $\rightarrow$ 855; 860 $\rightarrow$ 604; 862 $\rightarrow$ 1118; 866 $\rightarrow$ 802; 867 $\rightarrow$ 1123; 869 $\rightarrow$ 805; 870 $\rightarrow$ 806; 874 $\rightarrow$ 810; 879 $\rightarrow$ 943; 880 $\rightarrow$ 884; 881 $\rightarrow$ 877; 889 $\rightarrow$ 1145; 894 $\rightarrow$ 895; 899 $\rightarrow$ 900; 905 $\rightarrow$ 901; 910 $\rightarrow$ 4060; 922 $\rightarrow$ 921; 923 $\rightarrow$ 939; 924 $\rightarrow$ 908; 926 $\rightarrow$ 990; 929 $\rightarrow$ 1185; 934 $\rightarrow$ 1190; 945 $\rightarrow$ 949; 946 $\rightarrow$ 942; 956 $\rightarrow$ 892; 963 $\rightarrow$ 959; 968 $\rightarrow$ 712; 972 $\rightarrow$ 976; 982 $\rightarrow$ 978; 995 $\rightarrow$ 994; 996 $\rightarrow$ 932; 999 $\rightarrow$ 935;

1000 → 936; 1002 → 746; 1004 → 1008; 1005 → 749; 1010 → 1074; 1019 → 1023;
1020 → 1016; 1025 → 769; 1029 → 965; 1033 → 1017; 1038 → 1042; 1041 → 977;
1047 → 1031; 1049 → 985; 1053 → 797; 1065 → 1321; 1068 → 1067; 1069 → 1133;
1070 → 814; 1072 → 1136; 1076 → 1075; 1078 → 1062; 1086 → 1090; 1088 → 1104;
1099 → 1103; 1110 → 3945; 1112 → 1096; 1116 → 1120; 1125 → 1121; 1126 → 1127;
1142 → 1141; 1143 → 1139; 1147 → 1131; 1153 → 1157; 1165 → 1101; 1167 → 1423;
1172 → 916; 1176 → 1175; 1178 → 4013; 1179 → 1435; 1180 → 1436; 1196 → 1192;
1197 → 1193; 1202 → 1186; 1204 → 948; 1211 → 1210; 1217 → 1213; 1218 → 1214;
1223 → 1207; 1227 → 971; 1228 → 1164; 1229 → 1225; 1231 → 975; 1235 → 4070;
1236 → 980; 1237 → 1493; 1238 → 1174; 1239 → 1255; 1240 → 1496; 1244 → 988;
1245 → 1241; 1247 → 1183; 1248 → 1504; 1249 → 1505; 1250 → 1314; 1258 → 1322;
1259 → 1323; 1260 → 1256; 1261 → 1262; 1263 → 1199; 1264 → 1200; 1268 → 1524;
1269 → 1333; 1274 → 1270; 1280 → 1216; 1281 → 3801; 1285 → 3805; 1290 → 1546;
1293 → 1297; 1295 → 1551; 1300 → 1556; 1302 → 1046; 1304 → 1320; 1308 → 1307;
1312 → 1056; 1328 → 1329; 1340 → 1336; 1341 → 1337; 1343 → 1087; 1347 → 1091;
1350 → 3870; 1351 → 1352; 1354 → 1418; 1358 → 1102; 1362 → 1361; 1370 → 1371;
1374 → 1373; 1378 → 1442; 1379 → 1375; 1380 → 1396; 1381 → 1317; 1390 → 1391;
1394 → 1330; 1398 → 1402; 1399 → 1400; 1406 → 3926; 1409 → 1413; 1421 → 1425;
1428 → 1412; 1430 → 1429; 1443 → 1187; 1449 → 1433; 1456 → 1440; 1461 → 1465;
1467 → 1451; 1471 → 3991; 1476 → 1540; 1480 → 1464; 1481 → 1417; 1482 → 1478;
1483 → 1419; 1484 → 1488; 1485 → 1469; 1490 → 1426; 1501 → 1437; 1507 → 1508;
1513 → 1509; 1516 → 1512; 1517 → 1533; 1519 → 1518; 1520 → 1776; 1527 → 1271;

1531 → 1530; 1538 → 1522; 1548 → 1804; 1554 → 1555; 1558 → 1542; 1559 → 1623;

1560 → 1624; 1561 → 1817; 1566 → 1310; 1567 → 1823; 1569 → 1573; 1570 → 1586;

1575 → 1319; 1579 → 1578; 1580 → 1581; 1584 → 1840; 1591 → 1590; 1593 → 1849;

1595 → 1339; 1601 → 3806; 1605 → 1621; 1609 → 3814; 1612 → 1676; 1614 → 1550;

1616 → 1552; 1618 → 1619; 1626 → 1627; 1630 → 1629; 1632 → 1696; 1633 → 1697;

1634 → 1638; 1635 → 1651; 1646 → 1647; 1649 → 1585; 1655 → 1719; 1656 → 1652;

1660 → 1724; 1662 → 1658; 1664 → 1408; 1671 → 1670; 1672 → 1928; 1682 → 3887;

1684 → 3889; 1690 → 1691; 1700 → 1704; 1702 → 1718; 1712 → 3917; 1717 → 1781;

1726 → 1730; 1729 → 1733; 1732 → 1796; 1737 → 1738; 1739 → 1743; 1745 → 1744;

1746 → 1747; 1766 → 1750; 1769 → 1705; 1771 → 1755; 1772 → 1756; 1773 → 1837;

1777 → 1761; 1778 → 1779; 1790 → 1786; 1794 → 1795; 1800 → 1784; 1809 → 1805;

1810 → 1806; 1815 → 1751; 1816 → 1820; 1827 → 1828; 1830 → 1829; 1847 → 1848;

1854 → 1838; 1855 → 1839; 1856 → 1852; 1857 → 1853; 1858 → 1602; 1860 → 1844;

1861 → 1845; 1862 → 1798; 1863 → 2119; 1866 → 1802; 1870 → 1869; 1871 → 1807;

1874 → 1878; 1875 → 1879; 1876 → 1877; 1884 → 1883; 1885 → 1821; 1889 → 1888;

1890 → 1886; 1891 → 1892; 1894 → 1893; 1896 → 2152; 1897 → 1641; 1898 → 1834;

1899 → 1915; 1905 → 1904; 1906 → 1842; 1910 → 1654; 1925 → 1941; 1930 → 1674;

1935 → 1679; 1938 → 1937; 1939 → 1943; 1948 → 1692; 1950 → 1694; 1951 → 1695;

1954 → 1953; 1955 → 1956; 1958 → 1942; 1964 → 1900; 1967 → 3857; 1969 → 1973;

1970 → 1714; 1971 → 1715; 1974 → 2038; 1978 → 1722; 1981 → 1917; 1983 → 1727;

1987 → 1731; 1995 → 1991; 2008 → 1752; 2009 → 3899; 2010 → 2026; 2014 → 1758;

2015 → 1759; 2019 → 2035; 2021 → 2020; 2029 → 3919; 2040 → 2036; 2041 → 3931;

2043 → 2027; 2049 → 2048; 2053 → 2309; 2065 → 2321; 2066 → 2062; 2067 → 2063;
2071 → 2055; 2074 → 1818; 2078 → 2094; 2080 → 1824; 2081 → 2017; 2082 → 2098;
2085 → 2341; 2087 → 2088; 2096 → 2095; 2097 → 2093; 2107 → 2106; 2114 → 4004;
2117 → 2133; 2132 → 2068; 2136 → 2137; 2139 → 4029; 2143 → 2159; 2145 → 2129;
2146 → 4036; 2147 → 2148; 2150 → 2134; 2151 → 2155; 2163 → 2099; 2166 → 2102;
2172 → 2156; 2178 → 2242; 2181 → 2177; 2185 → 2201; 2187 → 2183; 2188 → 2124;
2189 → 1933; 2190 → 2126; 2192 → 2208; 2210 → 2209; 2219 → 2475; 2221 → 2157;
2222 → 2286; 2240 → 1984; 2244 → 2228; 2246 → 1990; 2251 → 2235; 2252 → 2316;
2253 → 2257; 2254 → 1998; 2262 → 2198; 2263 → 2199; 2274 → 2290; 2277 → 2213;
2279 → 2215; 2288 → 2352; 2293 → 2229; 2295 → 2231; 2296 → 2232; 2297 → 2301;
2298 → 2234; 2303 → 2367; 2307 → 2306; 2333 → 2329; 2338 → 2339; 2348 → 2347;
2361 → 3936; 2364 → 2363; 2371 → 2355; 2374 → 3949; 2376 → 2120; 2378 → 2122;
2385 → 2449; 2386 → 2450; 2390 → 3965; 2395 → 2331; 2397 → 2393; 2398 → 3973;
2399 → 2335; 2402 → 2401; 2406 → 2470; 2408 → 2412; 2414 → 2350; 2419 → 2675;
2422 → 2421; 2430 → 4005; 2435 → 2431; 2441 → 2425; 2443 → 2444; 2452 → 2388;
2462 → 2458; 2468 → 2212; 2472 → 2216; 2477 → 2478; 2481 → 2417; 2482 → 2738;
2489 → 2485; 2493 → 2429; 2496 → 2752; 2497 → 2433; 2503 → 2759; 2504 → 2568;
2511 → 2495; 2517 → 2513; 2521 → 2265; 2524 → 2460; 2531 → 2530; 2532 → 2596;
2533 → 2529; 2535 → 2471; 2536 → 2280; 2537 → 2281; 2538 → 2474; 2540 → 2796;
2544 → 2800; 2547 → 2543; 2548 → 2804; 2551 → 2550; 2552 → 2808; 2556 → 2300;
2566 → 2565; 2573 → 3833; 2574 → 2318; 2575 → 2639; 2576 → 2320; 2582 → 2581;
2601 → 2345; 2602 → 2346; 2604 → 2668; 2608 → 2607; 2609 → 2593; 2612 → 2676;

$2613 \to 2357$; $2615 \to 2359$; $2616 \to 2680$; $2620 \to 2636$; $2622 \to 2558$; $2624 \to 2688$;
$2626 \to 3886$; $2628 \to 2644$; $2634 \to 2650$; $2637 \to 2621$; $2647 \to 2631$; $2658 \to 2659$;
$2662 \to 2918$; $2663 \to 2664$; $2665 \to 2921$; $2683 \to 2667$; $2693 \to 2689$; $2694 \to 3954$;
$2699 \to 2700$; $2701 \to 2685$; $2707 \to 2703$; $2713 \to 2714$; $2717 \to 2718$; $2720 \to 2656$;
$2721 \to 2657$; $2723 \to 2724$; $2733 \to 2732$; $2746 \to 2730$; $2747 \to 2748$; $2760 \to 2696$;
$2762 \to 4022$; $2763 \to 4023$; $2766 \to 2782$; $2770 \to 2514$; $2771 \to 2755$; $2779 \to 3035$;
$2787 \to 2786$; $2789 \to 2785$; $2793 \to 2777$; $2794 \to 4054$; $2811 \to 2807$; $2813 \to 4073$;
$2815 \to 2816$; $2818 \to 2754$; $2822 \to 2806$; $2825 \to 2569$; $2827 \to 2891$; $2831 \to 2895$;
$2832 \to 2768$; $2834 \to 2898$; $2835 \to 2579$; $2836 \to 2900$; $2838 \to 2842$; $2839 \to 2583$;
$2840 \to 2584$; $2847 \to 2591$; $2850 \to 2846$; $2851 \to 2595$; $2852 \to 2916$; $2853 \to 3109$;
$2855 \to 2859$; $2866 \to 2862$; $2877 \to 2861$; $2889 \to 2888$; $2903 \to 2887$; $2908 \to 2924$;
$2926 \to 2925$; $2930 \to 2674$; $2933 \to 2917$; $2939 \to 2940$; $2943 \to 2687$; $2947 \to 2883$;
$2949 \to 2965$; $2953 \to 2952$; $2963 \to 2964$; $2967 \to 2971$; $2976 \to 2960$; $2977 \to 2913$;
$2979 \to 2975$; $2983 \to 2987$; $2984 \to 2920$; $2991 \to 2735$; $2992 \to 3008$; $2996 \to 2740$;
$2997 \to 2741$; $2998 \to 2982$; $3013 \to 3009$; $3017 \to 3016$; $3026 \to 3010$; $3030 \to 3286$;
$3037 \to 3038$; $3047 \to 3031$; $3048 \to 3044$; $3049 \to 3113$; $3054 \to 2990$; $3058 \to 3042$;
$3059 \to 3060$; $3066 \to 3002$; $3067 \to 3003$; $3075 \to 3076$; $3082 \to 3018$; $3083 \to 4028$;
$3085 \to 3341$; $3088 \to 3024$; $3094 \to 3093$; $3096 \to 3097$; $3101 \to 2845$; $3103 \to 3039$;
$3104 \to 3100$; $3107 \to 3091$; $3126 \to 3125$; $3127 \to 3143$; $3128 \to 3129$; $3132 \to 3068$;
$3136 \to 2880$; $3137 \to 3121$; $3142 \to 3206$; $3146 \to 3402$; $3148 \to 2892$; $3149 \to 3213$;
$3150 \to 2894$; $3151 \to 3155$; $3152 \to 3408$; $3153 \to 2897$; $3162 \to 2906$; $3164 \to 3160$;
$3165 \to 2909$; $3166 \to 3170$; $3167 \to 2911$; $3172 \to 3173$; $3178 \to 3174$; $3179 \to 2923$;

3182 → 3438; 3183 → 2927; 3188 → 3818; 3191 → 3207; 3192 → 2936; 3200 → 2944; 3204 → 2948; 3214 → 3470; 3215 → 3211; 3218 → 3234; 3226 → 3242; 3228 → 3224; 3236 → 3866; 3249 → 3185; 3252 → 3316; 3253 → 3257; 3255 → 3259; 3266 → 3896; 3267 → 3011; 3268 → 3012; 3269 → 3270; 3272 → 3902; 3276 → 3020; 3278 → 3274; 3279 → 3023; 3285 → 3221; 3292 → 3296; 3301 → 3302; 3303 → 3287; 3309 → 3939; 3310 → 3566; 3322 → 3258; 3325 → 3261; 3326 → 3327; 3329 → 3328; 3333 → 3077; 3336 → 3966; 3339 → 3403; 3349 → 3365; 3350 → 3414; 3355 → 3419; 3361 → 3617; 3362 → 3426; 3363 → 3359; 3367 → 3623; 3371 → 3115; 3376 → 3312; 3379 → 4009; 3382 → 3446; 3386 → 3450; 3389 → 3373; 3390 → 3394; 3396 → 4026; 3400 → 3399; 3417 → 3353; 3421 → 3420; 3428 → 3429; 3444 → 3445; 3455 → 3391; 3456 → 3452; 3457 → 3473; 3460 → 3476; 3465 → 3209; 3481 → 3497; 3482 → 3498; 3485 → 3469; 3487 → 3551; 3488 → 3424; 3492 → 3493; 3503 → 3499; 3504 → 3500; 3505 → 3441; 3506 → 3442; 3507 → 3251; 3508 → 3823; 3511 → 3495; 3512 → 3513; 3516 → 3260; 3517 → 3518; 3520 → 3584; 3522 → 3458; 3523 → 3587; 3524 → 3839; 3525 → 3526; 3527 → 3463; 3528 → 3843; 3531 → 3467; 3532 → 3468; 3539 → 3475; 3540 → 3284; 3544 → 3480; 3546 → 3550; 3547 → 3611; 3553 → 3557; 3564 → 3308; 3570 → 3634; 3575 → 51; 3576 → 3577; 3580 → 3324; 3591 → 3335; 3600 → 3601; 3603 → 3599; 3604 → 3605; 3606 → 3622; 3610 → 86; 3613 → 3614; 3632 → 3631; 3640 → 3624; 3652 → 3648; 3653 → 3637; 3662 → 3598; 3663 → 3659; 3666 → 3665; 3667 → 3411; 3668 → 3412; 3671 → 3415; 3673 → 149; 3678 → 154; 3683 → 3679; 3684 → 3620; 3689 → 3625; 3691 → 3755; 3692 → 3436; 3693 → 3694; 3696 → 172; 3697 → 3681; 3700 → 4015; 3701 → 3685; 3704 → 3703; 3705 → 3641; 3709 → 185; 3711 → 187;

3712 → 188; 3714 → 3730; 3715 → 3459; 3716 → 192; 3717 → 4032; 3718 → 3462; 3719 → 3723; 3720 → 4035; 3724 → 3660; 3728 → 3727; 3732 → 3733; 3735 → 4050; 3738 → 3742; 3740 → 24; 3750 → 3494; 3752 → 4067; 3761 → 3757; 3763 → 3759; 3764 → 3748; 3765 → 3749; 3766 → 3770; 3768 → 4083; 3775 → 3774; 3776 → 252; 3777 → 3773; 3779 → 3; 3780 → 2205; 3792 → 957; 3793 → 1273; 3794 → 329; 3796 → 961; 3798 → 1908; 3799 → 964; 3808 → 1288; 3816 → 666; 3819 → 2874; 3822 → 42; 3827 → 3197; 3835 → 2260; 3838 → 58; 3847 → 2587; 3848 → 1643; 3850 → 2905; 3860 → 3230; 3862 → 2287; 3871 → 721; 3873 → 3558; 3874 → 724; 3879 → 1044; 3880 → 415; 3882 → 732; 3883 → 2308; 3885 → 420; 3890 → 2000; 3894 → 1059; 3898 → 1693; 3903 → 2328; 3905 → 2645; 3906 → 756; 3908 → 1388; 3912 → 1707; 3915 → 2340; 3923 → 773; 3941 → 3626; 3942 → 1107; 3944 → 3629; 3950 → 3005; 3951 → 3006; 3959 → 2069; 3969 → 1764; 3970 → 1135; 3972 → 3027; 3977 → 3032; 3979 → 2404; 3981 → 831; 3986 → 836; 3990 → 2415; 3993 → 213; 3995 → 2420; 3997 → 847; 3999 → 2109; 4002 → 537; 4007 → 2432; 4011 → 3381; 4014 → 234; 4017 → 1182; 4020 → 555; 4030 → 2455; 4040 → 1205; 4043 → 1208; 4052 → 902; 4059 → 2799; 4061 → 2801; 4062 → 597; 4063 → 283; 4065 → 2175; 4069 → 3754; 4077 → 3762; 4079 → 3134; 4081 → 2506; 4087 → 937; 4092 → 1572;

Round 12: 1 → 3781; 2 → 3778; 3 → 3783; 4 → 3784; 5 → 3769; 6 → 3722; 8 → 3788; 9 → 3789; 10 → 3726; 11 → 3535; 12 → 3536; 13 → 3729; 14 → 18; 15 → 3731; 16 → 32; 17 → 3797; 19 → 3543; 20 → 3800; 21 → 3545; 24 → 3804; 25 → 3741; 26 → 90; 27 → 3807; 28 → 284; 29 → 285; 30 → 3746; 31 → 3555; 33 → 34; 35 → 3751; 37 → 3817; 39 → 3563; 41 → 40; 42 → 3758; 43 → 3567; 44 → 300; 48

$\rightarrow$ 3828; 49 $\rightarrow$ 305; 51 $\rightarrow$ 3831; 52 $\rightarrow$ 3832; 54 $\rightarrow$ 3578; 56 $\rightarrow$ 55; 58 $\rightarrow$ 122; 59 $\rightarrow$ 123; 61 $\rightarrow$ 3841; 63 $\rightarrow$ 62; 64 $\rightarrow$ 60; 65 $\rightarrow$ 3589; 66 $\rightarrow$ 322; 68 $\rightarrow$ 67; 69 $\rightarrow$ 53; 70 $\rightarrow$ 3594; 71 $\rightarrow$ 3851; 73 $\rightarrow$ 77; 75 $\rightarrow$ 74; 76 $\rightarrow$ 72; 78 $\rightarrow$ 3602; 79 $\rightarrow$ 3859; 80 $\rightarrow$ 336; 81 $\rightarrow$ 145; 83 $\rightarrow$ 82; 84 $\rightarrow$ 3608; 85 $\rightarrow$ 341; 86 $\rightarrow$ 102; 87 $\rightarrow$ 103; 88 $\rightarrow$ 3868; 89 $\rightarrow$ 153; 91 $\rightarrow$ 347; 92 $\rightarrow$ 3616; 94 $\rightarrow$ 3618; 95 $\rightarrow$ 3619; 96 $\rightarrow$ 3876; 97 $\rightarrow$ 3621; 98 $\rightarrow$ 114; 100 $\rightarrow$ 104; 101 $\rightarrow$ 3881; 106 $\rightarrow$ 3630; 107 $\rightarrow$ 111; 109 $\rightarrow$ 365; 112 $\rightarrow$ 368; 113 $\rightarrow$ 369; 118 $\rightarrow$ 3642; 119 $\rightarrow$ 115; 121 $\rightarrow$ 3645; 125 $\rightarrow$ 3649; 127 $\rightarrow$ 126; 128 $\rightarrow$ 124; 129 $\rightarrow$ 385; 130 $\rightarrow$ 3654; 131 $\rightarrow$ 3655; 132 $\rightarrow$ 388; 133 $\rightarrow$ 3657; 136 $\rightarrow$ 120; 137 $\rightarrow$ 3661; 138 $\rightarrow$ 142; 140 $\rightarrow$ 3664; 144 $\rightarrow$ 400; 146 $\rightarrow$ 3670; 147 $\rightarrow$ 148; 149 $\rightarrow$ 3929; 151 $\rightarrow$ 3675; 152 $\rightarrow$ 408; 154 $\rightarrow$ 150; 156 $\rightarrow$ 155; 157 $\rightarrow$ 141; 158 $\rightarrow$ 3682; 160 $\rightarrow$ 164; 162 $\rightarrow$ 178; 163 $\rightarrow$ 3687; 166 $\rightarrow$ 3690; 168 $\rightarrow$ 3948; 171 $\rightarrow$ 3695; 172 $\rightarrow$ 428; 174 $\rightarrow$ 430; 175 $\rightarrow$ 3699; 179 $\rightarrow$ 183; 180 $\rightarrow$ 3960; 182 $\rightarrow$ 3962; 184 $\rightarrow$ 248; 185 $\rightarrow$ 201; 186 $\rightarrow$ 202; 187 $\rightarrow$ 3967; 188 $\rightarrow$ 189; 192 $\rightarrow$ 448; 193 $\rightarrow$ 197; 198 $\rightarrow$ 134; 204 $\rightarrow$ 203; 205 $\rightarrow$ 221; 207 $\rightarrow$ 3987; 208 $\rightarrow$ 464; 212 $\rightarrow$ 3992; 213 $\rightarrow$ 3737; 214 $\rightarrow$ 210; 215 $\rightarrow$ 3739; 216 $\rightarrow$ 280; 217 $\rightarrow$ 233; 218 $\rightarrow$ 3998; 219 $\rightarrow$ 3743; 220 $\rightarrow$ 4000; 223 $\rightarrow$ 3747; 224 $\rightarrow$ 480; 225 $\rightarrow$ 229; 226 $\rightarrow$ 222; 228 $\rightarrow$ 227; 230 $\rightarrow$ 4010; 232 $\rightarrow$ 3756; 234 $\rightarrow$ 298; 236 $\rightarrow$ 237; 238 $\rightarrow$ 4018; 239 $\rightarrow$ 243; 240 $\rightarrow$ 176; 244 $\rightarrow$ 4024; 245 $\rightarrow$ 4025; 247 $\rightarrow$ 4027; 249 $\rightarrow$ 313; 250 $\rightarrow$ 266; 251 $\rightarrow$ 507; 252 $\rightarrow$ 508; 254 $\rightarrow$ 4034; 256 $\rightarrow$ 512; 257 $\rightarrow$ 4037; 258 $\rightarrow$ 242; 259 $\rightarrow$ 323; 260 $\rightarrow$ 261; 262 $\rightarrow$ 4042; 264 $\rightarrow$ 328; 265 $\rightarrow$ 4045; 267 $\rightarrow$ 263; 268 $\rightarrow$ 332; 269 $\rightarrow$ 4049; 270 $\rightarrow$ 286; 271 $\rightarrow$ 4051; 272 $\rightarrow$ 276; 273 $\rightarrow$ 4053; 274 $\rightarrow$ 275; 277 $\rightarrow$ 281; 279 $\rightarrow$ 278; 282 $\rightarrow$ 346; 283 $\rightarrow$ 539; 288 $\rightarrow$ 287; 289 $\rightarrow$ 545; 292 $\rightarrow$ 291; 294 $\rightarrow$ 290; 295 $\rightarrow$ 551; 297 $\rightarrow$ 296; 299 $\rightarrow$ 303; 302 $\rightarrow$ 301; 304 $\rightarrow$ 4084; 306 $\rightarrow$ 307;

308 → 4088; 309 → 293; 312 → 311; 314 → 310; 315 → 379; 317 → 316; 318 → 334; 319 → 383; 320 → 3785; 321 → 3786; 324 → 340; 325 → 3790; 329 → 333; 330 → 3795; 331 → 335; 337 → 3802; 338 → 354; 339 → 403; 342 → 326; 344 → 360; 345 → 361; 352 → 351; 355 → 3820; 358 → 357; 362 → 426; 363 → 367; 364 → 348; 366 → 350; 371 → 370; 372 → 3837; 375 → 359; 376 → 632; 377 → 393; 380 → 636; 382 → 381; 384 → 3849; 387 → 386; 390 → 389; 391 → 327; 392 → 396; 395 → 394; 398 → 654; 399 → 463; 401 → 397; 405 → 404; 406 → 402; 407 → 411; 409 → 665; 410 → 3875; 412 → 3877; 413 → 349; 415 → 671; 416 → 672; 417 → 673; 418 → 419; 420 → 356; 423 → 3888; 425 → 681; 427 → 683; 429 → 493; 432 → 436; 433 → 434; 435 → 499; 437 → 421; 442 → 438; 444 → 443; 447 → 511; 449 → 705; 450 → 194; 451 → 707; 452 → 708; 453 → 3918; 455 → 454; 456 → 440; 457 → 3922; 458 → 474; 460 → 3925; 462 → 466; 467 → 531; 468 → 3933; 470 → 469; 473 → 3938; 475 → 476; 481 → 465; 482 → 498; 483 → 739; 488 → 472; 489 → 485; 490 → 3955; 491 → 487; 494 → 510; 495 → 479; 496 → 497; 500 → 484; 501 → 757; 502 → 486; 503 → 504; 513 → 514; 515 → 3980; 517 → 3982; 519 → 518; 520 → 521; 522 → 778; 523 → 459; 524 → 3989; 525 → 529; 527 → 591; 528 → 544; 532 → 516; 534 → 530; 535 → 791; 536 → 600; 537 → 793; 538 → 542; 540 → 796; 541 → 477; 543 → 4008; 548 → 547; 550 → 546; 553 → 809; 554 → 618; 555 → 619; 556 → 552; 557 → 621; 560 → 559; 562 → 818; 563 → 819; 565 → 549; 566 → 822; 568 → 564; 569 → 505; 570 → 506; 571 → 575; 572 → 828; 573 → 509; 574 → 590; 576 → 832; 579 → 583; 580 → 596; 581 → 577; 582 → 578; 584 → 840; 585 → 589; 586 → 842; 592 → 588; 593 → 849; 595 → 594; 597 → 533; 599 → 598; 602 → 601; 603 → 667; 604 → 668;

606 → 4071; 607 → 863; 609 → 353; 611 → 675; 613 → 4078; 614 → 610; 615 → 4080; 616 → 620; 617 → 873; 622 → 623; 625 → 561; 626 → 690; 627 → 883; 628 → 612; 629 → 373; 630 → 694; 631 → 567; 633 → 649; 634 → 378; 635 → 639; 638 → 637; 640 → 624; 643 → 659; 644 → 660; 645 → 641; 646 → 662; 647 → 651; 648 → 904; 650 → 906; 652 → 716; 653 → 909; 656 → 655; 658 → 722; 661 → 3811; 663 → 3813; 664 → 920; 666 → 670; 669 → 605; 678 → 674; 679 → 743; 680 → 424; 682 → 938; 684 → 3834; 685 → 941; 688 → 687; 689 → 753; 691 → 755; 692 → 676; 693 → 677; 695 → 439; 697 → 441; 698 → 699; 700 → 764; 701 → 445; 702 → 446; 703 → 719; 704 → 720; 706 → 962; 709 → 725; 710 → 966; 711 → 727; 712 → 696; 713 → 969; 714 → 715; 717 → 781; 718 → 974; 721 → 657; 724 → 728; 726 → 790; 730 → 729; 731 → 987; 732 → 733; 734 → 478; 737 → 738; 740 → 736; 741 → 997; 744 → 760; 745 → 1001; 746 → 762; 747 → 1003; 748 → 492; 749 → 765; 750 → 1006; 751 → 735; 754 → 3904; 756 → 1012; 758 → 1014; 759 → 1015; 761 → 3911; 766 → 767; 768 → 752; 769 → 785; 770 → 834; 771 → 1027; 772 → 1028; 773 → 837; 774 → 775; 776 → 1032; 779 → 1035; 780 → 1036; 782 → 846; 784 → 783; 788 → 787; 789 → 1045; 794 → 1050; 795 → 859; 797 → 3947; 799 → 1055; 802 → 786; 803 → 804; 805 → 821; 806 → 742; 807 → 3957; 810 → 826; 811 → 827; 812 → 808; 814 → 798; 817 → 1073; 823 → 824; 825 → 3975; 831 → 830; 833 → 829; 836 → 1092; 838 → 3988; 839 → 1095; 841 → 845; 843 → 907; 847 → 911; 850 → 1106; 851 → 835; 853 → 1109; 855 → 1111; 856 → 857; 860 → 844; 862 → 4012; 866 → 4016; 867 → 871; 869 → 865; 870 → 854; 874 → 1130; 877 → 893; 879 → 815; 880 → 896; 881 → 885; 884 → 820; 886 → 950; 887 → 903; 889 → 888; 892 → 1148; 894 → 1150; 895

→ 891; 899 → 898; 900 → 1156; 901 → 917; 902 → 918; 905 → 4055; 908 → 4058; 910 → 1166; 914 → 915; 916 → 852; 921 → 925; 922 → 858; 923 → 919; 924 → 928; 926 → 927; 929 → 913; 932 → 868; 934 → 998; 935 → 1191; 936 → 872; 937 → 933; 939 → 1195; 942 → 878; 943 → 1007; 945 → 1201; 946 → 882; 948 → 947; 949 → 1013; 951 → 967; 954 → 890; 956 → 952; 957 → 1021; 959 → 958; 961 → 897; 963 → 1219; 964 → 960; 965 → 1221; 968 → 1224; 971 → 970; 972 → 973; 975 → 991; 976 → 1040; 977 → 3812; 978 → 1234; 980 → 3815; 982 → 983; 985 → 981; 988 → 1052; 990 → 3825; 994 → 993; 995 → 979; 996 → 1060; 999 → 1063; 1000 → 1064; 1002 → 986; 1004 → 940; 1005 → 1009; 1008 → 992; 1010 → 1011; 1016 → 1272; 1017 → 953; 1019 → 763; 1020 → 1084; 1023 → 1039; 1025 → 1089; 1029 → 1093; 1031 → 1287; 1033 → 1034; 1038 → 1294; 1041 → 1037; 1042 → 1026; 1044 → 1108; 1046 → 1030; 1047 → 1048; 1049 → 1305; 1053 → 1117; 1056 → 1057; 1059 → 1043; 1062 → 1318; 1065 → 1129; 1067 → 1083; 1068 → 1132; 1069 → 813; 1070 → 1066; 1072 → 816; 1074 → 1138; 1075 → 1079; 1076 → 1332; 1078 → 3913; 1086 → 1082; 1087 → 1071; 1088 → 1344; 1090 → 1154; 1091 → 1155; 1096 → 1080; 1099 → 1163; 1101 → 1097; 1102 → 1098; 1103 → 1359; 1104 → 848; 1107 → 1363; 1110 → 1366; 1112 → 1113; 1116 → 1115; 1118 → 1114; 1120 → 1376; 1121 → 3956; 1123 → 3958; 1125 → 1189; 1126 → 3961; 1127 → 1128; 1131 → 875; 1133 → 1389; 1135 → 1134; 1136 → 1137; 1139 → 3974; 1141 → 3976; 1142 → 1146; 1143 → 1159; 1145 → 1401; 1147 → 1403; 1153 → 1152; 1157 → 1161; 1164 → 1100; 1165 → 1169; 1167 → 1168; 1172 → 1171; 1174 → 1173; 1175 → 1431; 1176 → 1160; 1178 → 1162; 1179 → 1243; 1180 → 1184; 1182 → 1246; 1183 → 1119; 1185 → 1441; 1186 → 930; 1187 → 1203;

1190 → 1446; 1192 → 1448; 1193 → 1257; 1196 → 1452; 1197 → 1198; 1199 → 1455; 1200 → 944; 1202 → 1206; 1204 → 1188; 1205 → 1209; 1207 → 1463; 1208 → 1144; 1210 → 1466; 1211 → 955; 1213 → 1277; 1214 → 1470; 1216 → 1472; 1217 → 1233; 1218 → 1282; 1223 → 1479; 1225 → 1226; 1227 → 1291; 1228 → 1212; 1229 → 1230; 1231 → 1215; 1235 → 1491; 1236 → 1220; 1237 → 1253; 1238 → 1222; 1239 → 1303; 1240 → 984; 1241 → 1497; 1244 → 1500; 1245 → 989; 1247 → 1503; 1248 → 1232; 1249 → 1313; 1250 → 1254; 1255 → 1511; 1256 → 1252; 1258 → 1242; 1259 → 1275; 1260 → 1324; 1261 → 1325; 1262 → 1266; 1263 → 1279; 1264 → 1265; 1268 → 1284; 1269 → 1525; 1270 → 1526; 1271 → 1267; 1273 → 1529; 1274 → 1018; 1280 → 1024; 1281 → 1537; 1285 → 1541; 1288 → 1289; 1290 → 1306; 1293 → 1549; 1295 → 1299; 1297 → 1301; 1300 → 1316; 1302 → 1286; 1304 → 1368; 1307 → 1051; 1308 → 1564; 1310 → 1054; 1312 → 1568; 1314 → 1058; 1317 → 1061; 1319 → 1383; 1320 → 1384; 1321 → 1577; 1322 → 1386; 1323 → 1327; 1328 → 1392; 1329 → 1393; 1330 → 1331; 1333 → 1397; 1336 → 1592; 1337 → 1081; 1339 → 1355; 1340 → 1356; 1341 → 1085; 1343 → 3863; 1347 → 1283; 1350 → 1094; 1351 → 1607; 1352 → 1608; 1354 → 1610; 1358 → 1422; 1361 → 1105; 1362 → 1346; 1370 → 1434; 1371 → 1367; 1373 → 1369; 1374 → 1438; 1375 → 1439; 1378 → 1122; 1379 → 1315; 1380 → 1124; 1381 → 1382; 1388 → 1644; 1390 → 1326; 1391 → 1407; 1394 → 1410; 1396 → 1140; 1398 → 1462; 1399 → 1415; 1400 → 1404; 1402 → 1338; 1406 → 1405; 1408 → 3928; 1409 → 1473; 1412 → 1348; 1413 → 1669; 1417 → 1673; 1418 → 1414; 1419 → 1675; 1421 → 1357; 1423 → 3943; 1425 → 1681; 1426 → 1427; 1428 → 1364; 1429 → 1685; 1430 → 1494; 1433 → 1177; 1435 → 1499; 1436 → 1372; 1437 → 1181; 1440 → 1444; 1442 → 1698;

1443 → 3963; 1449 → 1450; 1451 → 1447; 1456 → 1460; 1461 → 1445; 1464 → 1720;
1465 → 1721; 1467 → 1723; 1469 → 1725; 1471 → 1475; 1476 → 1477; 1478 → 1474;
1480 → 1736; 1481 → 1545; 1482 → 1498; 1483 → 1547; 1484 → 1468; 1485 → 1741;
1488 → 1424; 1490 → 1486; 1493 → 1749; 1496 → 1432; 1501 → 1565; 1504 → 1760;
1505 → 1489; 1507 → 1251; 1508 → 1492; 1509 → 1765; 1512 → 1528; 1513 → 1514;
1516 → 1532; 1517 → 1453; 1518 → 1454; 1519 → 1535; 1520 → 1536; 1522 → 1523;
1524 → 1780; 1527 → 4047; 1530 → 1594; 1531 → 1787; 1533 → 1789; 1538 → 1539;
1540 → 1604; 1542 → 1543; 1546 → 1562; 1548 → 1292; 1550 → 1534; 1551 → 1615;
1552 → 1553; 1554 → 1298; 1555 → 1811; 1556 → 1812; 1558 → 1574; 1559 → 1495;
1560 → 1544; 1561 → 1625; 1566 → 1502; 1567 → 1311; 1569 → 4089; 1570 → 1506;
1572 → 1588; 1573 → 1557; 1575 → 1576; 1578 → 1642; 1579 → 1563; 1580 → 1596;
1581 → 1645; 1584 → 1600; 1585 → 1521; 1586 → 1587; 1590 → 1334; 1591 → 1335;
1593 → 1657; 1595 → 1851; 1601 → 1345; 1602 → 1666; 1605 → 1349; 1609 → 1353;
1612 → 1868; 1614 → 1598; 1616 → 1360; 1618 → 1617; 1619 → 1603; 1621 → 1365;
1623 → 1687; 1624 → 1880; 1626 → 1882; 1627 → 1611; 1629 → 1613; 1630 → 1631;
1632 → 1628; 1633 → 1377; 1634 → 1650; 1635 → 3840; 1638 → 1639; 1641 → 1640;
1643 → 1387; 1646 → 1902; 1647 → 1648; 1649 → 3854; 1651 → 1395; 1652 → 1668;
1654 → 1653; 1655 → 1911; 1656 → 1912; 1658 → 1914; 1660 → 1661; 1662 → 1918;
1664 → 1665; 1670 → 1926; 1671 → 1927; 1672 → 1416; 1674 → 1678; 1676 → 1420;
1679 → 1663; 1682 → 1683; 1684 → 1940; 1690 → 1946; 1691 → 1947; 1692 → 1708;
1693 → 1689; 1694 → 1710; 1695 → 1711; 1696 → 1680; 1697 → 1713; 1700 → 1716;
1702 → 1703; 1704 → 1960; 1705 → 3910; 1707 → 1963; 1712 → 1968; 1714 → 1458;

1715 → 1459; 1717 → 1701; 1718 → 1782; 1719 → 3924; 1722 → 1706; 1724 → 1980;
1726 → 1982; 1727 → 1791; 1729 → 1985; 1730 → 1734; 1731 → 1667; 1732 → 1988;
1733 → 1989; 1737 → 1801; 1738 → 1994; 1739 → 1735; 1743 → 1487; 1744 → 1728;
1745 → 2001; 1746 → 2002; 1747 → 3952; 1750 → 2006; 1751 → 2007; 1752 → 1688;
1755 → 2011; 1756 → 2012; 1758 → 1762; 1759 → 3964; 1761 → 1757; 1764 → 1748;
1766 → 2022; 1769 → 1833; 1771 → 1770; 1772 → 2028; 1773 → 1709; 1776 → 1775;
1777 → 2033; 1778 → 2034; 1779 → 1763; 1781 → 2037; 1784 → 1788; 1786 → 2042;
1790 → 1774; 1794 → 1793; 1795 → 1859; 1796 → 4001; 1798 → 2054; 1800 → 2056;
1802 → 2058; 1804 → 2060; 1805 → 2061; 1806 → 1742; 1807 → 1803; 1809 → 1825;
1810 → 1814; 1815 → 1799; 1816 → 2072; 1817 → 1753; 1818 → 1754; 1820 → 2076;
1821 → 2077; 1823 → 1819; 1824 → 1808; 1827 → 1571; 1828 → 2084; 1829 → 1813;
1830 → 1831; 1834 → 4039; 1837 → 1841; 1838 → 1582; 1839 → 1583; 1840 → 1836;
1842 → 1843; 1844 → 2100; 1845 → 1909; 1847 → 1783; 1848 → 2104; 1849 → 1913;
1852 → 2108; 1853 → 1597; 1854 → 2110; 1855 → 1599; 1856 → 2112; 1857 → 2113;
1858 → 1922; 1860 → 2116; 1861 → 1797; 1862 → 1606; 1863 → 1864; 1866 → 1850;
1869 → 1865; 1870 → 1934; 1871 → 4076; 1874 → 2130; 1875 → 2131; 1876 → 1620;
1877 → 1881; 1878 → 1622; 1879 → 2135; 1883 → 1867; 1884 → 2140; 1885 → 1901;
1886 → 1822; 1888 → 1872; 1889 → 1873; 1890 → 1826; 1891 → 1887; 1892 → 1636;
1893 → 1637; 1894 → 1895; 1896 → 1832; 1897 → 1961; 1898 → 1962; 1899 → 1835;
1900 → 1916; 1904 → 2160; 1905 → 2161; 1906 → 2162; 1908 → 2164; 1910 → 1846;
1915 → 1659; 1917 → 1921; 1925 → 1924; 1928 → 1992; 1930 → 2186; 1933 → 1929;
1935 → 1936; 1937 → 2193; 1938 → 2194; 1939 → 1923; 1941 → 2197; 1942 → 1686;

1943 → 1944; 1948 → 2204; 1950 → 2206; 1951 → 2207; 1953 → 1949; 1954 → 2018; 1955 → 1699; 1956 → 1972; 1958 → 2214; 1964 → 2220; 1967 → 1966; 1969 → 1965; 1970 → 2226; 1971 → 2227; 1973 → 1977; 1974 → 3864; 1978 → 1979; 1981 → 2237; 1983 → 2239; 1984 → 1920; 1987 → 2243; 1990 → 1986; 1991 → 2247; 1995 → 1931; 1998 → 1997; 2000 → 1999; 2008 → 2264; 2009 → 1993; 2010 → 2266; 2014 → 2270; 2015 → 2271; 2017 → 2273; 2019 → 2003; 2020 → 2024; 2021 → 2005; 2026 → 2282; 2027 → 2091; 2029 → 2025; 2035 → 2291; 2036 → 2052; 2038 → 2294; 2040 → 1976; 2041 → 2045; 2043 → 2299; 2048 → 2044; 2049 → 2305; 2053 → 2057; 2055 → 2311; 2062 → 2046; 2063 → 2047; 2065 → 2064; 2066 → 2322; 2067 → 2051; 2068 → 2324; 2069 → 2070; 2071 → 2327; 2074 → 2330; 2078 → 2334; 2080 → 2079; 2081 → 3971; 2082 → 2086; 2085 → 2089; 2087 → 2343; 2088 → 2344; 2093 → 2349; 2094 → 2158; 2095 → 2351; 2096 → 2032; 2097 → 2353; 2098 → 2354; 2099 → 2083; 2102 → 2358; 2106 → 2362; 2107 → 2123; 2109 → 2173; 2114 → 2050; 2117 → 2373; 2119 → 2103; 2120 → 2184; 2122 → 2121; 2124 → 2380; 2126 → 2382; 2129 → 2128; 2132 → 2196; 2133 → 2389; 2134 → 2138; 2136 → 2392; 2137 → 2153; 2139 → 2203; 2143 → 2127; 2145 → 2141; 2146 → 2142; 2147 → 2403; 2148 → 2144; 2150 → 2149; 2151 → 2167; 2152 → 2168; 2155 → 2171; 2156 → 2092; 2157 → 2413; 2159 → 1903; 2163 → 1907; 2166 → 4056; 2172 → 2428; 2175 → 1919; 2177 → 2241; 2178 → 2182; 2181 → 2437; 2183 → 2439; 2185 → 2169; 2187 → 2191; 2188 → 1932; 2189 → 2125; 2190 → 2446; 2192 → 2448; 2198 → 2454; 2199 → 2195; 2201 → 2457; 2205 → 2461; 2208 → 1952; 2209 → 2225; 2210 → 2466; 2212 → 2211; 2213 → 1957; 2215 → 1959; 2216 → 2200; 2219 → 2223; 2221 → 2285; 2222 → 2238; 2228 → 2224; 2229 → 2165; 2231 → 1975;

2232 → 2248; 2234 → 2250; 2235 → 2491; 2240 → 2304; 2242 → 2498; 2244 → 2180;

2246 → 2230; 2251 → 2315; 2252 → 2508; 2253 → 2509; 2254 → 2255; 2257 → 2256;

2260 → 2004; 2262 → 2518; 2263 → 2259; 2265 → 2249; 2274 → 2278; 2277 → 2261;

2279 → 2023; 2280 → 2284; 2281 → 2217; 2286 → 3861; 2287 → 2031; 2288 → 2272;

2290 → 2289; 2293 → 2549; 2295 → 2039; 2296 → 2360; 2297 → 2233; 2298 → 2554;

2300 → 2236; 2301 → 2557; 2303 → 2302; 2306 → 2310; 2307 → 2563; 2308 → 2564;

2309 → 2245; 2316 → 2317; 2318 → 2314; 2320 → 3895; 2321 → 2577; 2328 → 2312;

2329 → 2073; 2331 → 2267; 2333 → 2589; 2335 → 2319; 2338 → 2594; 2339 → 2275;

2340 → 2356; 2341 → 2337; 2345 → 2409; 2346 → 2090; 2347 → 2283; 2348 → 2332;

2350 → 2366; 2352 → 2336; 2355 → 2611; 2357 → 2101; 2359 → 3934; 2361 → 2617;

2363 → 2619; 2364 → 2368; 2367 → 2111; 2371 → 2115; 2374 → 2118; 2376 → 2375;

2378 → 2377; 2385 → 2381; 2386 → 2387; 2388 → 2372; 2390 → 2646; 2393 → 2649;

2395 → 2411; 2397 → 2653; 2398 → 2654; 2399 → 2400; 2401 → 2465; 2402 → 2418;

2404 → 2660; 2406 → 2405; 2408 → 2424; 2412 → 2396; 2414 → 2670; 2415 → 2416;

2417 → 2673; 2419 → 2423; 2420 → 2436; 2421 → 3996; 2422 → 2678; 2425 → 2426;

2429 → 2365; 2430 → 2174; 2431 → 2427; 2432 → 2176; 2433 → 2369; 2435 → 2179;

2441 → 2697; 2443 → 2459; 2444 → 2440; 2449 → 2705; 2450 → 2706; 2452 → 2456;

2455 → 2711; 2458 → 2394; 2460 → 2716; 2462 → 2526; 2468 → 2467; 2470 → 2486;

2471 → 2487; 2472 → 2728; 2474 → 2410; 2475 → 2731; 2477 → 2473; 2478 → 2734;

2481 → 2737; 2482 → 4057; 2485 → 2501; 2489 → 2745; 2493 → 2749; 2495 → 2494;

2496 → 2480; 2497 → 2561; 2503 → 2499; 2504 → 2500; 2506 → 2505; 2511 → 2507;

2513 → 2769; 2514 → 2510; 2517 → 2516; 2521 → 2525; 2524 → 2523; 2529 → 2528;

2530 → 2534; 2531 → 2515; 2532 → 2276; 2533 → 2469; 2535 → 2791; 2536 → 2792;
2537 → 2541; 2538 → 2522; 2540 → 2476; 2543 → 3803; 2544 → 2545; 2547 → 2483;
2548 → 2292; 2550 → 2614; 2551 → 2555; 2552 → 2553; 2556 → 2572; 2558 → 2542;
2565 → 2821; 2566 → 2570; 2568 → 2824; 2569 → 2313; 2573 → 2829; 2574 → 2830;
2575 → 2571; 2576 → 2640; 2579 → 2643; 2581 → 2597; 2582 → 3842; 2583 → 2519;
2584 → 2585; 2587 → 2651; 2591 → 2527; 2593 → 2849; 2595 → 2599; 2596 → 2592;
2601 → 2857; 2602 → 2858; 2604 → 2860; 2607 → 3867; 2608 → 2864; 2609 → 2865;
2612 → 2868; 2613 → 2869; 2615 → 2871; 2616 → 2872; 2620 → 2684; 2621 → 2605;
2622 → 2618; 2624 → 2560; 2626 → 2370; 2628 → 2884; 2631 → 3891; 2634 → 2698;
2636 → 2635; 2637 → 2893; 2639 → 2383; 2644 → 2580; 2645 → 2901; 2647 → 2391;
2650 → 2666; 2656 → 2652; 2657 → 2641; 2658 → 2642; 2659 → 2915; 2662 → 2661;
2663 → 2407; 2664 → 2648; 2665 → 2729; 2667 → 2603; 2668 → 2669; 2674 → 2690;
2675 → 3935; 2676 → 2677; 2680 → 2744; 2683 → 2682; 2685 → 2686; 2687 → 2623;
2688 → 2672; 2689 → 2945; 2693 → 2757; 2694 → 2438; 2696 → 2712; 2699 → 2695;
2700 → 2704; 2701 → 2765; 2703 → 2702; 2707 → 2451; 2713 → 2969; 2714 → 2710;
2717 → 2973; 2718 → 3978; 2720 → 2464; 2721 → 2722; 2723 → 2727; 2724 → 2725;
2730 → 2986; 2732 → 2988; 2733 → 2797; 2735 → 2671; 2738 → 2994; 2740 → 2484;
2741 → 2805; 2746 → 2490; 2747 → 2743; 2748 → 2764; 2752 → 2736; 2754 → 2758;
2755 → 2751; 2759 → 4019; 2760 → 2761; 2762 → 2778; 2763 → 3019; 2766 → 3022;
2768 → 2784; 2770 → 2774; 2771 → 2767; 2777 → 3033; 2779 → 2715; 2782 → 2798;
2785 → 3041; 2786 → 2802; 2787 → 2803; 2789 → 2773; 2793 → 2809; 2794 → 2790;
2796 → 3052; 2799 → 2863; 2800 → 3056; 2801 → 3057; 2804 → 4064; 2806 → 2870;

$2807 \to 3063$; $2808 \to 3064$; $2811 \to 2875$; $2813 \to 3069$; $2815 \to 2814$; $2816 \to 2817$;

$2818 \to 3074$; $2822 \to 4082$; $2825 \to 3081$; $2827 \to 2826$; $2831 \to 4091$; $2832 \to 2833$;

$2834 \to 4094$; $2835 \to 2899$; $2836 \to 3092$; $2838 \to 2837$; $2839 \to 3095$; $2840 \to 2844$;

$2842 \to 3787$; $2845 \to 2781$; $2846 \to 3791$; $2847 \to 2843$; $2850 \to 3106$; $2851 \to 2867$;

$2852 \to 2788$; $2853 \to 2854$; $2855 \to 3111$; $2859 \to 2795$; $2861 \to 3117$; $2862 \to 3118$;

$2866 \to 3122$; $2874 \to 3130$; $2877 \to 2878$; $2880 \to 2881$; $2883 \to 2627$; $2887 \to 2886$;

$2888 \to 2632$; $2889 \to 2633$; $2891 \to 2907$; $2892 \to 2828$; $2894 \to 2638$; $2895 \to 2879$;

$2897 \to 2961$; $2898 \to 2882$; $2900 \to 2904$; $2903 \to 2902$; $2905 \to 3161$; $2906 \to 2922$;

$2908 \to 3853$; $2909 \to 2910$; $2911 \to 2655$; $2913 \to 3169$; $2916 \to 2912$; $2917 \to 2981$;

$2918 \to 2934$; $2920 \to 3865$; $2921 \to 3177$; $2923 \to 2919$; $2924 \to 3869$; $2925 \to 3181$;

$2926 \to 2942$; $2927 \to 2931$; $2930 \to 3186$; $2933 \to 3878$; $2936 \to 3000$; $2939 \to 3884$;

$2940 \to 2876$; $2943 \to 3199$; $2944 \to 2928$; $2947 \to 2691$; $2948 \to 2692$; $2949 \to 2885$;

$2952 \to 2956$; $2953 \to 2937$; $2960 \to 3216$; $2963 \to 2959$; $2964 \to 2708$; $2965 \to 2966$;

$2967 \to 3223$; $2971 \to 3227$; $2975 \to 2719$; $2976 \to 3921$; $2977 \to 3233$; $2979 \to 3235$;

$2982 \to 2726$; $2983 \to 3239$; $2984 \to 3240$; $2987 \to 3932$; $2990 \to 3246$; $2991 \to 3007$;

$2992 \to 3937$; $2996 \to 2932$; $2997 \to 2993$; $2998 \to 3254$; $3002 \to 3001$; $3003 \to 3004$;

$3005 \to 2989$; $3006 \to 2750$; $3008 \to 3953$; $3009 \to 3265$; $3010 \to 2946$; $3011 \to 3015$;

$3012 \to 2756$; $3013 \to 3029$; $3016 \to 3080$; $3017 \to 3273$; $3018 \to 3014$; $3020 \to 3084$;

$3023 \to 3968$; $3024 \to 3028$; $3026 \to 3282$; $3027 \to 3043$; $3030 \to 3046$; $3031 \to 2775$;

$3032 \to 3288$; $3035 \to 3291$; $3037 \to 3293$; $3038 \to 3294$; $3039 \to 3055$; $3042 \to 3298$;

$3044 \to 3300$; $3047 \to 3051$; $3048 \to 3304$; $3049 \to 3994$; $3054 \to 3050$; $3058 \to 4003$;

$3059 \to 3315$; $3060 \to 3061$; $3066 \to 3065$; $3067 \to 3131$; $3068 \to 2812$; $3075 \to 3071$;

3076 → 2820; 3077 → 3073; 3082 → 3338; 3083 → 3099; 3085 → 3086; 3088 → 4033;
3091 → 3090; 3093 → 3089; 3094 → 3158; 3096 → 3352; 3097 → 2841; 3100 → 3116;
3101 → 4046; 3103 → 4048; 3104 → 2848; 3107 → 3123; 3109 → 3110; 3113 → 3112;
3115 → 3119; 3121 → 4066; 3125 → 3141; 3126 → 3190; 3127 → 4072; 3128 → 3384;
3129 → 4074; 3132 → 3388; 3134 → 3138; 3136 → 3135; 3137 → 3393; 3142 → 3078;
3143 → 3139; 3146 → 2890; 3148 → 4093; 3149 → 3405; 3150 → 3154; 3151 → 3087;
3152 → 2896; 3153 → 3409; 3155 → 3171; 3160 → 3144; 3162 → 3098; 3164 → 3180;
3165 → 3229; 3166 → 3102; 3167 → 3163; 3170 → 2914; 3172 → 3108; 3173 → 3157;
3174 → 3430; 3178 → 3434; 3179 → 3435; 3182 → 3198; 3183 → 3439; 3185 → 3201;
3188 → 3187; 3191 → 3821; 3192 → 3176; 3197 → 2941; 3200 → 3830; 3204 → 3140;
3206 → 2950; 3207 → 2951; 3209 → 3145; 3211 → 2955; 3213 → 3277; 3214 → 2958;
3215 → 3845; 3218 → 3474; 3221 → 3477; 3224 → 2968; 3226 → 3856; 3228 → 2972;
3230 → 2974; 3234 → 3490; 3236 → 2980; 3242 → 3306; 3249 → 3313; 3251 → 2995;
3252 → 3256; 3253 → 3189; 3255 → 3319; 3257 → 3321; 3258 → 3514; 3259 → 3195;
3260 → 3196; 3261 → 3245; 3266 → 3330; 3267 → 3897; 3268 → 3332; 3269 → 3205;
3270 → 3334; 3272 → 3208; 3274 → 3210; 3276 → 3280; 3278 → 3534; 3279 → 3909;
3284 → 3914; 3285 → 3281; 3286 → 3916; 3287 → 3351; 3292 → 3548; 3296 → 3552;
3301 → 3045; 3302 → 3238; 3303 → 3559; 3308 → 3307; 3309 → 3053; 3310 → 3940;
3312 → 3311; 3316 → 3946; 3322 → 3318; 3324 → 3340; 3325 → 3581; 3326 → 3582;
3327 → 3583; 3328 → 3072; 3329 → 3585; 3333 → 3337; 3335 → 3079; 3336 → 3592;
3339 → 3323; 3341 → 3597; 3349 → 3413; 3350 → 3346; 3353 → 3983; 3355 → 3354;
3359 → 3423; 3361 → 3360; 3362 → 3378; 3363 → 3299; 3365 → 3364; 3367 → 3368;

$3371 \to 3627$; $3373 \to 3437$; $3376 \to 4006$; $3379 \to 3635$; $3381 \to 3385$; $3382 \to 3638$; $3386 \to 3387$; $3389 \to 3133$; $3390 \to 3646$; $3391 \to 4021$; $3394 \to 3650$; $3396 \to 3395$; $3399 \to 3383$; $3400 \to 3656$; $3402 \to 3658$; $3403 \to 3147$; $3408 \to 3407$; $3411 \to 4041$; $3412 \to 3156$; $3414 \to 3478$; $3415 \to 3159$; $3417 \to 3401$; $3419 \to 3483$; $3420 \to 3676$; $3421 \to 3677$; $3424 \to 3168$; $3426 \to 3427$; $3428 \to 3432$; $3429 \to 3433$; $3436 \to 3372$; $3438 \to 4068$; $3441 \to 3377$; $3442 \to 3698$; $3444 \to 3380$; $3445 \to 4075$; $3446 \to 3702$; $3450 \to 3706$; $3452 \to 3451$; $3455 \to 4085$; $3456 \to 4086$; $3457 \to 3713$; $3458 \to 3202$; $3459 \to 3203$; $3460 \to 3464$; $3462 \to 3398$; $3463 \to 3479$; $3465 \to 3721$; $3467 \to 3782$; $3468 \to 3212$; $3469 \to 3533$; $3470 \to 3454$; $3473 \to 3489$; $3475 \to 3219$; $3476 \to 3220$; $3480 \to 3416$; $3481 \to 3225$; $3482 \to 3418$; $3485 \to 3549$; $3487 \to 3231$; $3488 \to 3232$; $3492 \to 3491$; $3493 \to 3237$; $3494 \to 3809$; $3495 \to 3810$; $3497 \to 3753$; $3498 \to 3562$; $3499 \to 3243$; $3500 \to 3244$; $3503 \to 3247$; $3504 \to 3248$; $3505 \to 3569$; $3506 \to 3250$; $3507 \to 3443$; $3508 \to 3509$; $3511 \to 3826$; $3512 \to 3496$; $3513 \to 3449$; $3516 \to 3515$; $3517 \to 3453$; $3518 \to 3262$; $3520 \to 3264$; $3522 \to 3586$; $3523 \to 3519$; $3524 \to 3588$; $3525 \to 3521$; $3526 \to 3510$; $3527 \to 3271$; $3528 \to 3529$; $3530 \to 3466$; $3531 \to 3275$; $3532 \to 3596$; $3537 \to 3538$; $3539 \to 3283$; $3540 \to 3855$; $3544 \to 3560$; $3546 \to 3290$; $3547 \to 23$; $3550 \to 3554$; $3551 \to 3295$; $3553 \to 3297$; $3557 \to 3872$; $3558 \to 3574$; $3561 \to 3305$; $3564 \to 3565$; $3566 \to 3502$; $3570 \to 3314$; $3572 \to 3571$; $3573 \to 3317$; $3575 \to 3639$; $3576 \to 3320$; $3577 \to 3892$; $3580 \to 3644$; $3584 \to 3568$; $3587 \to 3331$; $3591 \to 3590$; $3598 \to 3342$; $3599 \to 3343$; $3600 \to 3344$; $3601 \to 3345$; $3603 \to 3347$; $3604 \to 3348$; $3605 \to 3541$; $3606 \to 3542$; $3610 \to 3674$; $3611 \to 3595$; $3612 \to 3356$; $3613 \to 3357$; $3614 \to 3358$; $3617 \to 93$; $3620 \to 3556$; $3622 \to 3366$; $3623 \to 99$; 3624

→ 3688; 3625 → 3369; 3626 → 3370; 3629 → 105; 3631 → 3375; 3632 → 108; 3634 → 110; 3637 → 3633; 3640 → 116; 3641 → 117; 3643 → 3707; 3648 → 3392; 3652 → 3651; 3653 → 3397; 3659 → 135; 3660 → 3404; 3662 → 3406; 3663 → 139; 3665 → 3669; 3666 → 3410; 3667 → 143; 3668 → 3672; 3671 → 3607; 3673 → 3609; 3678 → 3422; 3679 → 3615; 3681 → 3425; 3683 → 159; 3684 → 3680; 3685 → 161; 3689 → 165; 3691 → 167; 3692 → 3628; 3693 → 169; 3694 → 170; 3696 → 3440; 3697 → 173; 3700 → 3636; 3701 → 177; 3703 → 3447; 3704 → 3448; 3705 → 181; 3709 → 3725; 3711 → 3647; 3712 → 3708; 3714 → 190; 3715 → 191; 3716 → 4031; 3717 → 3461; 3718 → 3734; 3719 → 195; 3720 → 196; 3723 → 199; 3724 → 200; 3727 → 3471; 3728 → 3472; 3730 → 206; 3732 → 3736; 3733 → 209; 3735 → 211; 3738 → 22; 3740 → 3484; 3742 → 3486; 3748 → 3744; 3749 → 3745; 3750 → 3686; 3752 → 36; 3754 → 38; 3755 → 231; 3757 → 3501; 3759 → 235; 3761 → 45; 3762 → 46; 3763 → 47; 3764 → 3760; 3765 → 241; 3766 → 50; 3768 → 3767; 3770 → 246; 3771 → 7; 3773 → 57; 3774 → 3710; 3775 → 4090; 3776 → 3772; 3777 → 253; 3779 → 255; 3780 → 2520; 3792 → 642; 3793 → 2218; 3794 → 1589; 3796 → 1276; 3798 → 1278; 3799 → 2539; 3801 → 2856; 3805 → 3175; 3806 → 2546; 3808 → 343; 3814 → 3184; 3816 → 1296; 3818 → 2873; 3819 → 2559; 3822 → 2562; 3823 → 3193; 3824 → 3194; 3827 → 2567; 3829 → 1309; 3833 → 2258; 3835 → 1945; 3836 → 686; 3838 → 2578; 3839 → 374; 3843 → 2268; 3844 → 2269; 3846 → 2586; 3847 → 3217; 3848 → 2588; 3850 → 2590; 3852 → 3222; 3857 → 1022; 3858 → 2598; 3860 → 2600; 3862 → 1342; 3866 → 2606; 3870 → 2610; 3871 → 3241; 3873 → 723; 3874 → 2929; 3879 → 414; 3880 → 2935; 3882 → 1677; 3883 → 2938; 3885 → 2625; 3886 → 1996; 3887 → 422;

$3889 \rightarrow 2629$; $3890 \rightarrow 2630$; $3893 \rightarrow 3263$; $3894 \rightarrow 3579$; $3896 \rightarrow 431$; $3898 \rightarrow 2323$; $3899 \rightarrow 2954$; $3900 \rightarrow 2325$; $3901 \rightarrow 2326$; $3902 \rightarrow 2957$; $3903 \rightarrow 2013$; $3905 \rightarrow 1385$; $3906 \rightarrow 2016$; $3907 \rightarrow 2962$; $3908 \rightarrow 3593$; $3912 \rightarrow 1077$; $3915 \rightarrow 2970$; $3917 \rightarrow 2342$; $3919 \rightarrow 3289$; $3920 \rightarrow 2030$; $3923 \rightarrow 2978$; $3926 \rightarrow 461$; $3927 \rightarrow 777$; $3930 \rightarrow 2985$; $3931 \rightarrow 1411$; $3936 \rightarrow 471$; $3939 \rightarrow 2679$; $3941 \rightarrow 2681$; $3942 \rightarrow 792$; $3944 \rightarrow 2999$; $3945 \rightarrow 1740$; $3949 \rightarrow 2059$; $3950 \rightarrow 800$; $3951 \rightarrow 801$; $3954 \rightarrow 2379$; $3959 \rightarrow 2384$; $3965 \rightarrow 2075$; $3966 \rightarrow 3021$; $3969 \rightarrow 2709$; $3970 \rightarrow 3025$; $3972 \rightarrow 1767$; $3973 \rightarrow 1768$; $3977 \rightarrow 1457$; $3979 \rightarrow 3034$; $3981 \rightarrow 3036$; $3984 \rightarrow 1149$; $3985 \rightarrow 3040$; $3986 \rightarrow 1151$; $3990 \rightarrow 1785$; $3991 \rightarrow 526$; $3993 \rightarrow 1158$; $3995 \rightarrow 2105$; $3997 \rightarrow 1792$; $3999 \rightarrow 2739$; $4002 \rightarrow 2742$; $4004 \rightarrow 3374$; $4005 \rightarrow 1170$; $4007 \rightarrow 3062$; $4009 \rightarrow 2434$; $4011 \rightarrow 861$; $4013 \rightarrow 2753$; $4014 \rightarrow 864$; $4015 \rightarrow 3070$; $4017 \rightarrow 2442$; $4020 \rightarrow 2445$; $4022 \rightarrow 2447$; $4023 \rightarrow 558$; $4026 \rightarrow 876$; $4028 \rightarrow 2453$; $4029 \rightarrow 1194$; $4030 \rightarrow 1510$; $4032 \rightarrow 2772$; $4035 \rightarrow 1515$; $4036 \rightarrow 2776$; $4038 \rightarrow 2463$; $4040 \rightarrow 2780$; $4043 \rightarrow 2783$; $4044 \rightarrow 2154$; $4050 \rightarrow 3105$; $4052 \rightarrow 587$; $4054 \rightarrow 2479$; $4059 \rightarrow 3114$; $4060 \rightarrow 2170$; $4061 \rightarrow 3431$; $4062 \rightarrow 912$; $4063 \rightarrow 2488$; $4065 \rightarrow 3120$; $4067 \rightarrow 2492$; $4069 \rightarrow 3124$; $4070 \rightarrow 2810$; $4073 \rightarrow 608$; $4077 \rightarrow 2502$; $4079 \rightarrow 2819$; $4081 \rightarrow 931$; $4083 \rightarrow 2823$; $4087 \rightarrow 2512$; $4092 \rightarrow 2202$;