UNIVERSITY OF CALIFORNIA, SAN DIEGO

Trees for Group Key Management with Batch Update

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science and Engineering

by

Nan Zang

Committee in charge:

Professor Ronald L. Graham, Chair
Professor Chung-Kuan Cheng
Professor Fan Chung Graham
Professor William Helton
Professor Jeff Remmel
Professor George Varghese

2008

The dissertation of Nan Zang is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2008

DEDICATION

To my beloved parents.

To my dear husband.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor Ronald L. Graham. You introduced me to this interesting area of research and provided guidance for my research directions.

I am also grateful to Professor Fan Chung Graham, who not only taught me combinatorial methods and graph theory but helped me to build confidence, which is priceless in my life.

Finally, I would like to thank Steve Butler for giving me very valuable comments about this dissertation.

The text of Chapter Two, in part, is adapted from the paper which has been submitted for publication to the Journal of Combinatorics and Number Theory. The dissertation author was the primary investigator and author of this paper.

The text of Chapter Four, in part, is adapted from the paper which has been accepted for publication to Theoretical Computer Science. The dissertation author was the primary investigator and author of this paper.

VITA

2003	B. S. in Computer Science, University of Science and Technology of China, Hefei, China
2005	M. S. in Computer Science, University of California, San Diego
2003-2007	Graduate Research Assistant, University of California, San Diego
2007-2008	Graduate Teaching Assistant, University of California, San Diego
2008	Ph. D. in Computer Science, University of California, San Diego

PUBLICATIONS

Ronald L. Graham, Nan Zang, "Enumerating split-pair arrangements", *Journal of Combinatorial Theory, Series A*, 115-2, 2008, 293-303.

Minming Li, Ze Feng, Nan Zang, Ronald L. Graham and F. F. Yao, "Approximately optimal trees for group key management with batch updates", accepted by *Theoretical Computer Science*.

Steve Butler, Ronald L. Graham, Nan Zang, "Optimal jumping patterns", submitted to *Journal of Combinatorics and Number Theory*.

ABSTRACT OF THE DISSERTATION

Trees for Group Key Management with Batch Update

by

Nan Zang

Doctor of Philosophy in Computer Science and Engineering

University of California San Diego, 2008

Professor Ronald L. Graham, Chair

Due to the tremendous increase in bandwidth of the network, the group-oriented broadcast services are becoming increasingly popular. These group-oriented broadcast services, such as teleconferencing and pay-per-view TV, have a large number of subscribers and a central group controller (GC), which is in charge of all the security related administrative tasks. All the members in the group share a key called the Traffic Encryption Key (TEK), and the messages broadcast by the group members are encrypted by the TEK. For security reasons, if there is a group membership change, the TEK has to be updated. So, the GC needs to send some rekeying messages. To minimize the number of rekeying messages, an efficient and popular method is to arrange the members in a tree structure and assign the auxiliary keys, called the Key Encryption Key (KEK), to the members belonging to a same subtree. The problem of finding an optimal tree structure to

minimize the rekeying messages to be sent is called the Group Key Management (GKM) problem. The tree structures are called Group Key Management trees.

The group Key Management problem has been a popular research topic for several years. Many models have been proposed. In this dissertation, we mainly focus on a batch rekeying model. In this model, the number of group members n is fixed and each member has probability p ($p = 1 - q$) of being replaced by a new member during a batch period.

In this dissertation, we focus on the problem of constructing "good" trees and analyzing the properties of GKM optimal trees for fixed q and n. Two efficient approximation algorithms are proposed to construct such trees: one is for the limiting case when the number of group members becomes unbounded and the other is for the limiting case as $q \to 1$. We also relax the GKM problem to a more general mathematical problem, called the Jumping Sequence Problem. We give detailed analysis of the properties of the jumping sequences. Based on those properties, we also provide a lower bound for optimal GKM trees.

Chapter 1

Introduction

Because of the tremendous increase in bandwidth in recent years, group-oriented broadcast services are becoming increasingly popular. These group broadcast network services, such as teleconferencing and pay-per-view TV, have n subscribers (n is a large number) and a central group controller (GC) that broadcasts data packages to all the subscribers over an insecure channel. The group-oriented broadcast model was first introduced by Harney and Muckenhirn [1][2]. Each group has a single GC which is in charge of all the security related administrative tasks such as member verification and distribution of the group keys. Each member in the group can use IP multicast to broadcast data packages to other group members, which is different from the traditional unicast communication technology which requires sending n copies of each data package. However, security must be guaranteed, which means only the verified members in the group can access the

data packages which are being broadcast. However scalable, IP multicast cannot provide an effective mechanism to control the access to the broadcast data [3]. This security issue in the group-oriented broadcast is called the secure multicast communication problem.

The secure multicast communication problem has received significant research attention in recent years [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17]. A simple and popular method of limiting the access to the broadcast data packages is through encryption. One symmetric group key K_g is shared by the central group controller (GC) and all n group members. Whenever, a new member wants to join the group, GC and the new member will mutually authenticate each other using a protocol such as SSL [18]. If the new member is verified, GC will assign a private key to him, which is only shared by the new member and GC. After the joining process, GC will broadcast the group key information, encrypted by the private key of the new member. So, each verified member in the group will hold the group key K_g. All the data packages sent among the group members are encrypted by the group key K_g. However, this popular encryption mechanism brings up another security issue. The members of the group are not stable. Some current members may leave in the future. After a member leaves, the previous group key K_g can no longer be used and a message containing the new group key must be sent to all the remaining members, which is called the rekeying message. If each member only has two keys, the group key K_g and its own private key, the new group key

must be encrypted for each remaining group member using its individual private key. So, $n-1$ messages have to be sent. A good way to decrease the number of rekeying messages is to manage the group members in a hierarchical architecture: decompose a large group of members into many subgroups, and members in each subgroup share one subgroup key [19] [20].

This dissertation addresses the problem of finding the optimal key tree to minimize the rekeying messages which are required for the group security. Member verification mechanisms are not addressed in this dissertation.

In this chapter, we will first introduce a mechanism, called the key graph approach, to represent the key hierarchical architecture and some popular key graphs in the literature. In Section 2, we introduce a more efficient rekeying mechanism, which uses periodical batch rekeying update after several joins/leaves. The problems discussed in this dissertation are based on the batch update rekeying mechanism.

1.1 Key Graphs

The secure group discussed above can be represented as follows [20].

A secure group is a triple $(U; K; R)$ where

- U is a finite nonempty set of users,
- K is a finite nonempty set of keys, and

- R is a binary relation between U and K, that is, $R \subseteq U \times K$, called the user-key relation. User u has key k if and only if $(u; k)$ is in R.

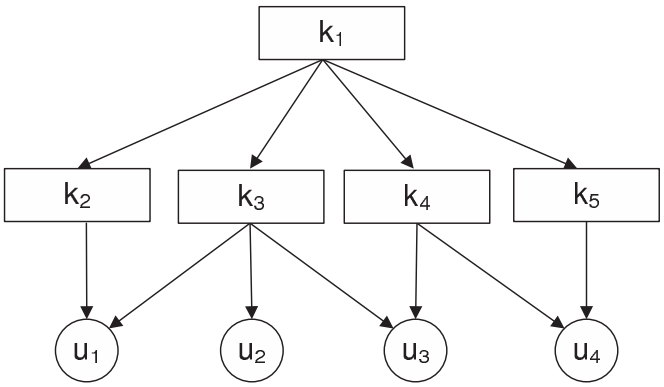


Figure 1.1: An example key graph.

A key graph is a directed acyclic graph used to represent the user-key relation. A key graph contains two types of nodes: user-nodes representing users U and key-nodes representing keys K. In a key graph, each user-node has one or more incoming edges but no outgoing edges. If a key-node has only outgoing edges and no incoming edges, then this key-node is called the root. Given a key graph $G = (V, E)$, it specifies a secure group as follows.

- For each user u in U, there is one and only one corresponding user-node in G.

- For each key k in K, there is one and only one corresponding key-node in G.

- User-key relation $(u; k)$ is in R, if and only if there is a direct path from key-node k to user-node u.

A key graph example is given in Figure 1.1. In the example, we can see $U = \{u_1, u_2, u_3, u_4\}$ and $K = \{k_1, k_2, k_3, k_4, k_5\}$, with u_1 holding the keys $\{k_1, k_2, k_3\}$, u_2 holding $\{k_1, k_3\}$, u_3 holding $\{k_1, k_4\}$ and u_4 holding $\{k_1, k_4, k_5\}$. Note, G doesn't show the private key of each user. Each user u_i's private key pk_i is only held by GC and itself.

A key graph is used by GC for key management purposes. In a key graph, each key is shared by its reachable members. For security reasons, when a member leaves, every key that has been held by the leaving member and shared by other members should be replaced. Let u_a be the member who leaves, and let k_a be such a key. Denote U_a as the set of users holding k_a. To replace k_a, GC randomly generates a new key and sends it to every member who holds k_a except u_a ($U_a \backslash u_a$). To guarantee the security, GC needs to find a subset of keys K' such that the members in $U_a \backslash u_a$ hold at least one key in K'. So, GC can encrypt the new key by each key in K', and send out these $|K'|$ messages. Each member in $U_a \backslash u_a$ can read at least one encrypted rekeying message. To minimize the number of rekeying messages ($|K'|$), GC has to find a minimal size subset of keys, which cover exactly all the members in U_a. In Figure 1.1, if u_1 leaves, GC has to replace keys k_1, k_2, k_3 and can do it using $K' = \{pk_2, k_4\}$. The problem was first introduced by Wong and Lam [21], called the Key Covering problem, and the authors showed that it is NP-complete, which is proved by showing the NP-hard Set Covering problem can be reduced to the Key Covering problem in polynomial time.

Definition 1.1.1 (Key Covering Problem)**.** Given a key graph G representing the secure group $(U; K; R)$, and a subset S of U. Find a minimum size subset K' of K such that $S \subseteq userset(K')$. Here, $userset(K')$ is defined as the set of users holding at least one key in K'.

1.2 Special Key Graphs

1.2.1 Key Trees

In this dissertation, we only consider key graphs with very special structures, namely key trees. In this case, the key graph is a single-rooted tree. In the key tree model, all the internal nodes are key-nodes, and all the leaves are the user-nodes. For example, Figure 1.2 gives a key tree, in which $U = \{u_1, \ldots, u_8\}$ and $K = \{k_1, \ldots, k_5\}$.

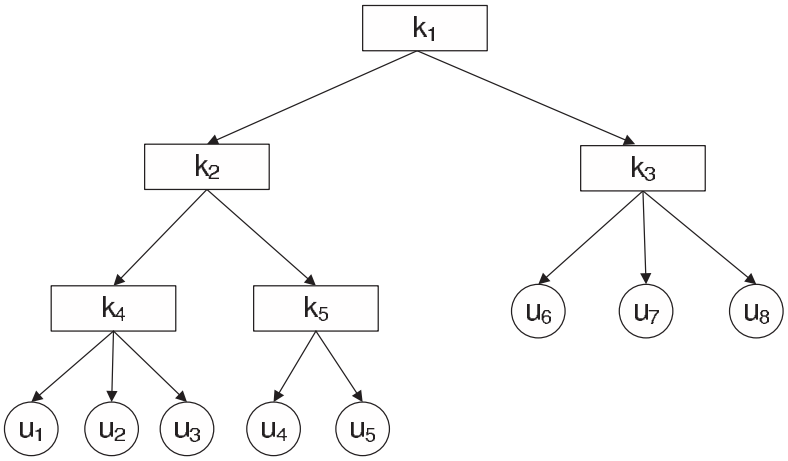


Figure 1.2: An example key tree.

As we can see, in the key tree, GC is represented by the root, and the n members of the group represented by the n leaves of the tree. Associated with every internal node of the tree is an encryption key. The key associated with the root is called the Traffic Encryption Key (TEK), which is used for communicating confidential information among the group members. The key k_v associated with each non-root node v is called a Key Encryption Key (KEK) which is used for updating the TEK when necessary. Each member possesses all the keys along the path from the leaf representing itself to the root.

In the key tree, there is always an efficient way to find a rekeying process, which sends the minimum number of rekeying messages. After a user leaves, the GC accomplishes the rekeying task by broadcasting the new keys, in encrypted form, from the lowest level upward recursively as follows: Let v be an internal node at the lowest level whose key needs to be (but has not yet been) updated. For each child u of v, the GC broadcasts a message containing $E_{k_u^{new}}(k_v^{new})$, which means the encryption of k_v^{new} with the key k_u^{new}. Thus the GC sends out d_v broadcast messages for updating k_v if v has d_v children. Updating this way ensures that the member which left will not know any information about the new keys while current members can use one of their KEKs to decrypt the useful $E_{k_u^{new}}(k_v^{new})$ sequentially until they get the new TEK.

Take Figure 1.2 as an example; if member u_1 leaves the group, GC has to update keys K_4, K_2, K_1. Six rekeying messages have to be sent:

- $E_{pk_2}(K_4^{new})$, $E_{pk_3}(K_4^{new})$ to update K_4;
- $E_{K_4^{new}}(K_2^{new})$, $E_{K_5}(K_2^{new})$ to update K_2;
- $E_{K_2^{new}}(K_1^{new})$, $E_{K_3}(K_1^{new})$ to update the TEK K_1.

In a key tree T, if u leaves and the keys are updated following the above rekeying mechanism, the minimum number of rekeying messages has to be sent is $\sum_{v \in path(u,r)} d_v$, where $path(u, r)$ is defined as the path from u to the root r.

1.2.2 Key Stars

A key star is a special structure of a key tree: the level of key tree is 2; the degree of the root equals group size. An example is shown in Figure 1.3. Key star models the traditional approach we discussed earlier. Every user has two keys: its private key and the group key. Whenever a group member leaves, $n - 1$ rekeying messages are required, where n is the size of the group.

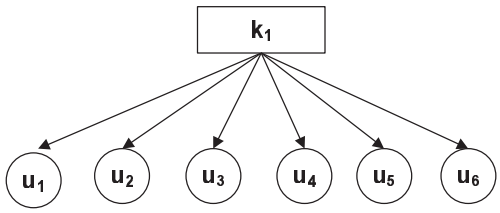


Figure 1.3: An example key star.

1.3 Group Key Management with Batch Update

The group key management with batch rekeying update is first introduced by Li et al. [18]. They showed that the rekeying method after each join/leave has two problems. First, it is relatively inefficient, especially when the members of the group change frequently. Second, there is an out-of-sync problem between keys and data. Because the data packages,which are transmitted by the network, always encounter a delay problem. A group member might receive a data message encrypted by an old group key, or it might receive a data message encrypted by a new group key, which has not yet been received. In the batch rekeying model, GC waits for a period of time, called a batch period, collects all the join and leave requests, generates new keys, constructs rekeying messages, and multicasts these messages.

Zhu et al. [22] introduced a new batch rekeying model based on some very popular network services. For those network services, they can only serve at most n members and many other customers must wait to be served. After a period of time, the system will eject a current member with probability p. So, in this new model, the number of members joining is assumed to be equal to the number of members leaving during a batch updating period and every current member has probability p of being replaced by a new member. In the key tree model, if a member changes, the rekeying update process will be performed from the lowest level upward, as in the rekeying process introduced in Section 1.2. Here, we will

calculate the update cost involved in the rekeying process.

During a batch period, each member has probability p to change. We define the update cost as the number of rekeying messages to be sent in a batch period. As introduced in Section 1.2.1, if one member changes, we have to update all the keys this member possessed; that is all the internal nodes along the path from the corresponding leaf to the root. We update them following a bottom to top order. Here, we use Figure 1.2 to show how to calculate the update cost. If u_1 changes, GC first sends the new key K_4 to u_1 and its siblings (u_2, u_3). Since they don't have a common accessed key, GC has to send three messages which are the key K_4 encrypted by the private keys of u_1, u_2 and u_3, respectively. To update key K_2, GC needs to send the new key K_2 to all its descendants, but only two messages are required to be sent, which are the key K_2 encrypted by the subgroup keys K_4, K_5. Due to the above process, it is easy to see that we need to send d_v messages to renew the key K_v, where d_v is the outgoing degree of v. In a batch period, each key-node v has probability $1-(1-p)^{N_v}$ to change, where N_v is the number of members holding the key K_v. So, the number of expected messages to be sent in the batch period is $\sum_{v \in V}(d_v \cdot (1-(1-p)^{N_v}))$.

1.4 The GKM Tree Problem

In this section, we will first define some terminology, and then give the mathematical definition of the GKM tree problem, which this dissertation will

focus on.

Definition 1.4.1. For a tree T, $L(T)$ is defined as the set of leaves in T, and $|L(T)|$ is the number of leaves in T. For every node u in T, d_u is defined as the outgoing degree of u. We also denote the subtree rooted at the node u as T_u and N_u as the number of leaves of T_u.

The GKM tree problem is defined as follows: given the number of leaves n and a probability p a member changes, how can we arrange the members in a tree structure to minimize the expected number of rekeying messages to be sent. Mathematically, we can restate this in the following form.

Definition 1.4.2 (The GKM Tree Problem)**.** Given an integer $n > 1$ and a constant p $(0 < p < 1)$; the weight of a tree T is defined as

$$C(q, T) = \sum_{v} d_v \cdot (1 - q^{N_v}),$$

where the sum is taken over all the nodes v of T and $q = 1 - p$. The problem is to find a tree structure T^* with n leaves so that

$$C(q, T^*) = \min_{|L(T)|=n} \{C(q, T)\}.$$

The (q, n)-optimal tree refers to the tree structure T^*. The minimum cost is denoted as $OPT(q, n)$. In this dissertation, we will simplify $C(q, T)$ as $C(T)$, if there is no ambiguity about the value of q in the context.

1.5 Contributions

This dissertation focuses on the problem of constructing "good" GKM trees given q and n. We make the following contributions in this dissertation.

1. For the limiting case as $n \to \infty$, we show, in optimal GKM trees, there always exists a unique subtree structure of the root, which appears an unbounded number of times. In the case that this structure is symmetric, we prove it has very good properties, and give an $O(\log n)$ time algorithm to find it.

2. For the limiting case as $q \to 1$, we give a nearly 2-approximation algorithm to build GKM trees with time $O(\log n)$.

3. We define the Jumping Sequence Problem, which is to find the jumping sequences from 1 to n where there is a cost associated with each jump. We give detailed analysis of the properties of the jumping sequences, and provide a lower bound for optimal GKM trees.

1.6 Dissertation Organization

We organize the rest of this dissertation as follows. In Chapter Two, first we give the properties of optimal GKM trees, and then relax the GKM tree problem to a more general mathematical problem, called the Jumping Sequence Problem. By analyzing the properties of the jumping sequences, we give a lower bound of the cost of optimal GKM trees. In Chapter Three, we consider the case when the

number of group members n is unbounded, and design an efficient approximation algorithm to construct GKM trees. In Chapter Four, we consider the other limiting case when $q \to 1$, and modify the cost functions defined in Chapter One. For these new cost functions, we analyze the properties of the jumping sequences and give a nearly 2-approximation algorithm to build a GKM tree in $O(\log n)$ time. Finally, Chapter Five presents conclusions and future directions.

Chapter 2

Properties of Optimal GKM Trees

The GMK tree problem was first introduced by Zhu et al. [22]. The authors analyzed the properties of the trees with the restriction that the degree of the node can be only a power of 2. Graham et al. [23] did a detailed analysis of the properties of optimal tree structures. This was the first theoretical paper studying this problem.

2.1 Related Work

In this section, we will survey some results concerning GKM trees in the literature.

2.1.1 Trees with Restricted Degree

Zhu et al. [22] first defined a special set of "symmetric" trees whose degrees are restricted to powers of 2. $T(a_1, a_2, ..., a_t)$ denotes a tree with t-levels, such that the outgoing degree of the root is 2^{a_1}, and the nodes on the ith level have outgoing degree 2^{a_i}.

Lemma 2.1.1. *If $T(a_1, a_2, ..., a_t)$ is an optimal tree for q and m, then the tree $T(a_i, a_{i+1})$ is an optimal tree for q_1 and m_1. Here, $m = \sum_{h=1}^{t} a_h$, $q_1 = q^{-(\sum_{k=i+2}^{t} a_k)}$, $m_1 = a_i + a_{i+1}$ and $i < t - 1$.*

Proof. The cost of the optimal tree $T(a_1, a_2, ..., a_t; q)$ is

$$C(q, T) = 2^{a_1}(1 - q^{2^m}) + 2^{a_1+a_2}(1 - q^{2^{m-a_1}}) + ... + 2^m(1 - q^{2^{a_t}}).$$

The cost function could be rewritten in the following form, for any $1 \leq i < j \leq t$:

$$\begin{aligned} C(q, T) =& C(q_{i-1}, T(a_1, ..., a_i)) + 2^{a_1+...+a_{i-1}} C(q_j, T(a_i, ..., a_j) \\ &+ 2^{a_1+...+a_j} C(q_j, T(a_{j+1}, ..., a_t)), \end{aligned}$$

where $q_{i-1} = q^{-(\sum_{k=i}^{t} a_k)}$ and $q_j = q^{-(\sum_{k=j+1}^{t} a_k)}$.

From the above transformation, we can see if $T(a_i, ..., a_j)$ is not an optimal tree for $q = q_j$ and $m_j = \sum_{h=i}^{j} a_h$, the $T(a_1, ..., a_t)$ is not an optimal tree which is a contradiction. By setting $j = i + 1$, we can see that if $T(a_1, a_2, ..., a_t)$ is optimal for q, then for any $i \leq t - 1$, $T(a_i, a_{i+1})$ must be an optimal tree structure for $q_1 = q^{-(\sum_{k=i+2}^{t} a_k)}$ and $m_1 = a_i + a_{i+1}$. □

The following Theorem was first introduced in [22]. Here, I give a simplified proof by using Lemma 2.1.1.

Theorem 2.1.2. *If a tree $T(a_1, a_2, ..., a_t)$ is an optimal tree, then $a_1 \geq 2$, $a_2 = a_3 = ... = a_{t-1} = 2$ and $a_t = 1$ or 2.*

Proof. From Lemma 2.1.1, we know if a tree $T(a_1, a_2, ..., a_t)$ is an optimal tree for q, $T(a_i, a_{i+1})$ is optimal for $q_1 = q^{-(\sum_{k=i+2}^{t} a_k)}$. Here, we want to prove for any two-level tree, if $T(a_i, a_{i+1})$ is optimal, then $a_{i+1} \leq 2$ and $a_i \geq 2$.

We first prove $a_{i+1} \leq 2$ by contradiction. We assume $a_{i+1} > 2$. We consider two alternative trees $T(a_i, a_{i+1} - 1, 1)$ and $T(m_i)$, and show that for any $0 < q < 1$, $T(a_i, a_{i+1})$ can always be replaced by a better alternative tree, where $m_i = a_i + a_{i+1}$. The cost of the original tree is $C(q, T(a_i, a_{i+1})) = 2^{a_i}(1-q^{2^{m_i}}) + 2^{m_i}(1-q^{2^{a_{i+1}}})$. The cost of the first alternative tree is $C(q, T(a_i, a_{i+1}-1, 1)) = 2^{a_i}(1-q^{2^{m_i}}) + 2^{m_i-1}(1-q^{2^{a_{i+1}}}) + 2^{m_i}(1-q^2)$; the cost of the second alternative tree is $C(q, T(m_i)) = 2^{m_i}(1-q^{2^{m_i}})$. It is easy to check when $0 \leq q < 2^{-\frac{a_{i+1}}{2^{a_{i+1}}}}$, $C(q, T(a_i, a_{i+1}-1, 1))$ is smaller; and when $q \geq 2^{-\frac{a_{i+1}}{2^{a_{i+1}}}}$, $C(q, T(m_i))$ is smaller. Thus, $T(a_i, a_{i+1})$ cannot be the optimal tree when $a_{i+1} > 2$.

Second, we prove $a_i \geq 2$. Since we have proved that if $T(a_i, a_{i+1})$ is an optimal tree then $a_{i+1} \leq 2$, it is sufficient to show that neither of $T(1,1)$ nor $T(1,2)$ is optimal. $T(2)$ is always better than $T(1,1)$, since $C(q, T(1,1)) - C(q, T(2)) = 2(1-q^4) + 4(1-q^2) - 4(1-q^4) = 2(q^2-1)^2 \geq 0$. By drawing the function behavior of $f(q)$, $g(q)$ and $h(q)$ in Figure 2.1, it is obvious that $T(1,2)$ can never

be an optimal tree, where $f(q) = C(q, T(2,1)) = 4(1-q^8) + 8(1-q^2)$, $g(q) = C(q, T(1,2)) = 2(1-q^8) + 8(1-q^4)+$ and $h(q) = C(q, T(3)) = 8(1-q^8)$.

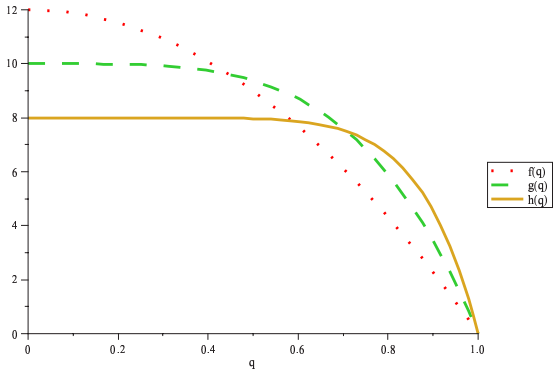


Figure 2.1: Behavior of functions $f(q)$, $g(q)$ and $h(q)$ for $q = 0 \cdots 1$.

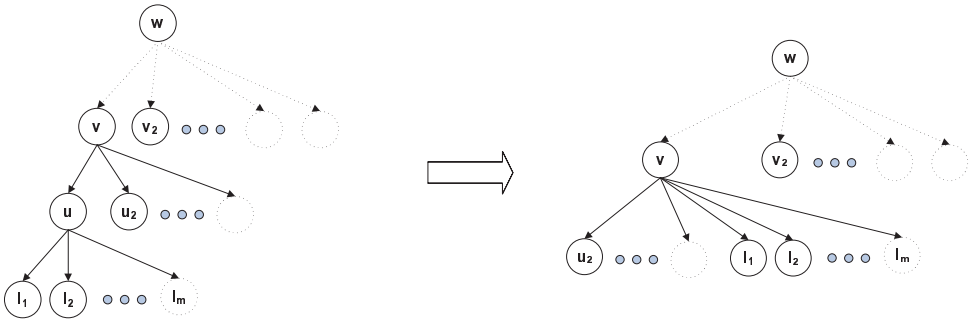
From the above analysis, we know if $T(a_1, a_2, ..., a_t)$ is optimal, then $a_i \geq 2$, $a_{i+1} \leq 2$ for any $i \leq t-1$. So the proof is completed. □

2.1.2 Degree Restrictions of Optimal GKM Trees

In this section, we introduce some important properties of optimal GKM trees. Based on these properties, Graham et al. [23] gave an algorithm to construct a (q, n)-optimal tree where $0 < q < 1$ and $n > 0$.

Theorem 2.1.3. *Any subtree of the (q, n)-optimal tree is an optimal tree.*

Proof. We prove it by contradiction. Suppose one subtree with n_1 leaves of a (q, n)-optimal tree is not optimal. Then we can substitute this nonoptimal subtree by (q, n_1)-optimal tree to get another (q, n) tree with smaller cost than the (q, n)-optimal tree. This is a contradiction. □



(a) T_1 (b) T_2

Figure 2.2: Tree transformation one.

Theorem 2.1.4. *When $0 \leq q < 3^{-1/3}$, a key star with n leaves is the optimal tree for any n.*

Proof. This Theorem was first given in [23]. Here, we give another version of the proof using Theorem 2.1.3. We prove it by contradiction.

Assume for $0 \leq q < 3^{-1/3}$, the (q, n)-optimal tree is not a star. There must be a subtree of (q, n)-optimal tree with the structure shown in Figure 2.2(a), in which v is an internal node with degree $d_v \geq 2$, u is a child of v with $d_u \geq 2$ and all the children of u are leaves. Now, we show that the cost of the (q, n)-optimal tree can be decreased by connecting u's children to v directly, as shown in Figure 2.3. The cost of the original tree is

$$C(T_1) = d_v(1 - q^{N_v}) + d_u(1 - q^{N_u}) + C',$$

where C' is the contribution of the cost of other edges except for the solid edges

shown in Figure 2.2(a); on the other hand, the cost of the new tree is

$$C(T_2) = (d_v + d_u - 1)(1 - q^{N_v}) + C'.$$

By this transformation, we change the total cost by

$$\begin{aligned}\Delta(C) &= C(T_1) - C(T_2) \\ &= d_v(1 - q^{N_v}) + d_u(1 - q^{N_u}) - (d_v + d_u - 1)(1 - q^{N_v}) \\ &= 1 - d_u q^{N_u} + q^{N_v}(d_u - 1)\end{aligned}$$

By assumption, we know $d_u - 1 \geq 0$. Now we want to prove $1 - d_u q^{N_u} \geq 0$ when $0 \leq q < 3^{-1/3}$. It is easy to see that for fixed x, the function: $f(q) = xq^x$ is maximized when $x = 1/\ln(1/q)$. Based on the value of q, we consider the following two cases:

1. If $0 < q < 1/e^{1/2}$, which implies $1/\ln(1/q) < 2$, the function $d_u q^{N_u}$ reaches the maximum value when $d_u = 2$. It is easy to verify that $2q^2 < 2/e < 1$ when $0 < q < 1/e^{1/2}$.

2. If $1/e^{1/2} \leq q < 1/3^{1/3}$, which implies $2 \leq 1/\ln(1/q) < 3$, the function $d_u q^{N_u}$ reaches the maximum value when $d_u = 2$ or $d_u = 3$. It is easy to verify that $2q^2 \leq 2(1/3)^{2/3} \approx 0.97 < 1$, and $3q^3 < 1$ when $q < 3^{-1/3}$.

From this analysis, we show that the cost of the subtree of the (q, n)-optimal tree will be decreased by the above transformation. This contradicts our original assumption. So, when $0 \leq q < 3^{-1/3}$, a star is the optimal tree for any n. $\square$

From Theorem 2.1.4, we know the n-star is always the unique optimal key tree when $0 < q < 3^{-1/3}$. So, to find the optimal GKM tree, we only need to consider the case when $3^{-1/3} \leq q < 1$. In the following two theorems, some important properties of the optimal trees are given, the degrees of all the internal nodes other than the root are bounded, and the number of leaves of all the subtrees are bounded. Base on these properties, a polynomial dynamic programming algorithm is given to construct an optimal GKM tree when $3^{-1/3} \leq q < 1$.

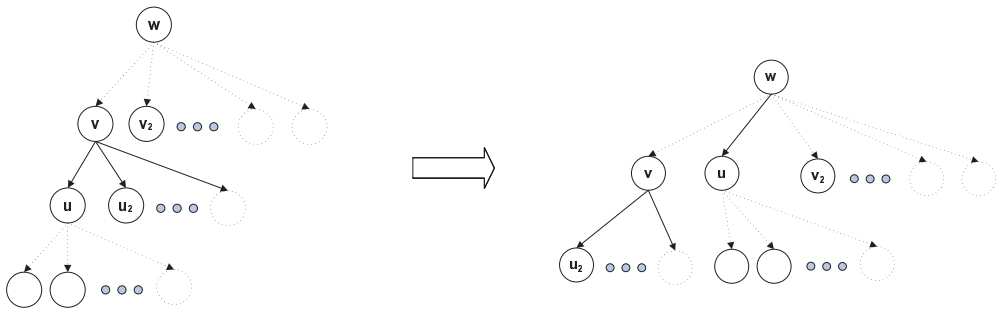
Theorem 2.1.5. *In a (q, n)-optimal tree, all internal nodes other than the root, must have degree 4 or less.*

Theorem 2.1.5 is proved by a switching technique. For a tree with an internal node of degrees more than 4, there always exists a better tree whose degrees are bounded by 4. For the details of the proof see [23].

Theorem 2.1.6. *In a (q, n)-optimal tree, if v is a child of the root, then the number N_v of leaves underneath v is upper bounded by* $\max\{4(\log q^{-1})^{-1}, 1\}$.

Proof. For a tree T, we associate a value $t_v = q^{N_v}$ with every node v. Recall the subtree rooted at v is denoted by T_v. We say T_u is a subtree of v, if u is a child of v. First, we need to show in a (q, n)-optimal tree, $t_u \geq \frac{d_u - 1}{d_u}$, where u is a child of a non-root internal node v.

We prove this by contradiction. If $t_u < \frac{d_u}{d_u - 1}$, then we can move u up to become a sibling of v, as shown in Figure 2.1.2. In this transformation, we change



(a) T_1 (b) T_2

Figure 2.3: Tree transformation two.

the total cost of the tree by

$$\begin{aligned}\Delta(C) &= 1 - q^{N_w} + (d_v - 1)(1 - q^{N_v - N_u}) - d_v(1 - q^{N_v}) \\ &< 1 + (d_v - 1)(1 - q^{N_v - N_u}) - d_v(1 - q^{N_v - N_u} t_u) \\ &= q^{N_v - N_u}(d_v t_u - (d_v - 1)) \\ &< 0,\end{aligned}$$

where w is the parent of v and N_v represents the value before the transformation. This contradicts the cost optimality of the original tree.

If u is not a leaf, we know $d_u \geq 2$ and $t_u = q^{N_u} \geq 1/2$. Thus, $N_u < (\log q^{-1})^{-1}$. Because v has at most 4 children by Theorem 2.1.5 and $N_v \geq 1$, we have $N_v \leq \max\{4(\log q^{-1})^{-1}, 1\}$. □

We will revisit Theorem 2.1.6 in Chapter Three, when analyzing the case $n \to \infty$. If q is a constant and $n \to \infty$, Theorem 2.1.6 guarantees that the size of the subtrees of the root is bounded by a constant.

2.1.3 Algorithm for Constructing Optimal GKM Trees

Based on Theorem 2.1.5 and Theorem 2.1.6, one can give a dynamic programming algorithm for constructing optimal GKM trees. First, the algorithm builds all the optimal subtrees which have outgoing degrees 2, 3 or 4, and which have at most $4(\log q^{-1})^{-1}$ descendant leaves. Second, among all the optimal subtrees, the algorithm groups them together, connecting to the root, to make a (q, n)-optimal tree. The algorithm is shown as follows (Algorithm 1). Here is a definition used in the algorithm.

Definition 2.1.7. A (q, L, n)-forest is defined as a forest with a lot of L leaves. The cost of a tree edge $e = (u, v)$ is defined as $w(e) = 1 - q^{N_v}$, where v is u's parent. The cost of the forest is the sum of tree costs plus $k(1 - q^n)$, where k is the number of trees in the forest. We define the optimal-(q, L, n)-forest to be a (q, L, n)-forest with minimum cost.

The time complexity of Algorithm 1 is $O(nK+K^4)$, where $K = 4(\log q^{-1})^{-1}$. The "while" loop in Algorithm 1 takes time $O(K^4)$. For each $R(i)$, $2 \leq i \leq 4$, we have to consider all the decompositions: $i_1 + i_2 + i_3 + i_4 = i$, where $0 \leq i_1 \leq i_2 \leq i_3 \leq i_4 \leq i$. There are K^3 possible decompositions. The second "for" loop in Algorithm 1 takes time $O(nK)$.

If q is a constant, the time complexity of the algorithm gets larger as $n \to \infty$ or $q \to 1$. In the next two chapters, we will focus on these two special cases, and will give two approximation algorithms with better time complexity for the limiting

Algorithm 1 *COT (Constructing Optimal Tree)*

Input: n and q

Output: An optimal GKM tree for n and a

$K = \min\{4(\log q^{-1})^{-1}, 1\}$

if $q < 3^{-1/3}$ or $2 \leq n \leq 4$ **then**

$OPT(q, n) \leftarrow n(1 - q^n)$

Return $OPT(q, n)$

end if

$R(1) = 0$

for $i = 2$ to 4 **do**

$R(i) = i * (1 - q^i)$

end for

$i = 5$

while $i < K$ **do**

Compute $R(i)$, cost of the restricted (q, i)-optimal tree.

$i = i + 1$

end while

$Forest(q, n, 0) \leftarrow 0$

for $L = 1$ to n **do**

$Forest(q, n, L) \leftarrow min(R(j) + 1 - q^n + F(q, n, L - j))$ over all j, $1 \leq j \leq \min\{K, L\}$

end for

$OPT(q, n) \leftarrow Forest(q, n, n)$

Return $OPT(q, n)$

case when $n \to \infty$ or when $q \to 1$. In the next section, we first relax the GKM tree problem to a more general mathematical problem which we call the jumping sequence problem, and give a lower bound on the weight of the (q, n)-optimal tree by analysis of the behavior of the jumping sequences.

2.2 Lower Bound for Optimal GKM Tree

In this section, we give the definition of the Jumping Sequence Problem. Then, we will look at the properties of jumping sequences from 1 to n.

2.2.1 The Jumping Sequence Problem

In a tree T with n leaves, $L(T)$ is the set of the leaves of T. For each leaf u, let the set of ancestor nodes of u (including u itself) be denoted by $Anc(u)$. To obtain a lower bound for the optimal tree cost, we first rewrite $C(q, T)$ as

$$\begin{aligned} C(q,T) &= \sum_{u} d_u \cdot (1 - q^{N_u}) \\ &= \sum_{u \in V} w(u) = \sum_{u \in V} N_u \cdot \frac{w(u)}{N_u} \\ &= \sum_{u \in L(T)} \sum_{x \in Anc(u)} \frac{w(x)}{N_x} \\ &= \sum_{u \in L(T)} c(q,u) \end{aligned}$$

where we define $c(q, u) = \sum_{x \in Anc(u)} \frac{w(x)}{N_x}$. In other words, we distribute the weight $w(x)$ associated with every node $x \in V$ evenly among its leaf descendants. So,

the weight of the tree $C(q,T)$ is composed of the new weight function of each leaf $c(q,u)$ over all leaves of T. Therefore, $c(q,u)$ is the cost per leaf for the leaf u.

Let the path from a leaf u to the root r be $p_0p_1\ldots p_{k-1}p_k$, where $p_0 = u$ and $p_k = r$. Note that $c(q,u) = \sum_{i=0}^{k-1} \frac{1-q^{N_{p_i+1}}}{N_{p_i}}$ is uniquely determined by the sequence of numbers $\{N_{p_0}, N_{p_1}, \ldots, N_{p_k}\}$, where $N_{p_0} = 1$ and $N_{p_k} = n$. We will thus extend the definition of c to all such sequences $(1 = a_0, a_1, \ldots, a_k = n)$, where a_i is a real number. We will analyze the minimum value of c.

Definition 2.2.1 (Jumping Sequence Problem)**.** Let S_n denote any sequence of (real) numbers $(a_0, a_1, \ldots, a_k)$ satisfying $1 = a_0 < a_1 < \ldots < a_k = n$. We call S_n an n-progression. Define $c(q, S_n)$ to be $c(q, S_n) = \sum_{i=1}^{k} \frac{1-q^{a_i}}{a_{i-1}}$ and let $F(q,n)$ be the minimum of $c(q, S_n)$ over all n-progressions S_n. Given q and n, we call the problem of finding the best sequences from 1 to n with the minimum cost as the jumping sequence problem.

It is easy to see in any GKM tree with n leaves, the number of leaves of the nodes along the path from a leaf u to the root r $(1 = N_u, N_{p_1}, \ldots, N_r = n)$ is a special n-progression. Thus, $F(q,n)$ gives a lower bound for the cost per leaf value for any leaf in any tree T. So, we get $OPT(q,T) \geq n \cdot F(q,n)$.

2.2.2 Properties of Jumping Sequences

In this section, we focus on properties of the jumping sequences. First, we derive the following monotone property for $F(q,n)$.

Theorem 2.2.2. *For a fixed q, $F(q,n) < F(q,n+1)$.*

Proof. Suppose $S^*_{n+1} = \{1 = a_0, a_1, \ldots, a_{k-1}, a_k = n+1\}$ is an optimal jump sequence from 1 to $n+1$. Because $\frac{1-q^{n+1}}{a_{k-1}} > \frac{1-q^n}{a_{k-1}}$, we have $c(q, S^*_{n+1}) > c(q, S^*_n)$, where $S^*_n = (1 = a_0, a_1, \cdots, a_{k-1}, a_k = n)$. Since $F(q,n)$ is defined as the minimum cost over all n-progression S_n, we have $F(q,n) < c(q, S^*_n)$. Combining these two facts, we proved that $F(q,n) < F(q,n+1)$. □

Optimal jump sequences have the following natural recursive relationship. This is helpful as it allows us to reduce the problem of how to make a general jump to the problem of how to jump from 1.

Theorem 2.2.3. *Let $1 < a_1 < a_2 < \cdots < a_{k-1} < n$ be an optimal jump sequence associated with q and n. Then for any $1 \le i \le k$, the sequence $1 < a_{i+1}/a_i < a_{i+2}/a_i < \cdots < a_{k-1}/a_i < n/a_i$ is an optimal jump sequence for $q' = q^{a_i}$ and $n' = n/a_i$.*

Proof. The proof follows by noting that the cost function is optimal, and can be rewritten as

$$\begin{aligned} F(q,n) &= \sum_{j=0}^{k-1} \frac{1-q^{a_{j+1}}}{a_j} = \sum_{j=0}^{i-1} \frac{1-q^{a_{j+1}}}{a_j} + \sum_{j=i}^{k-1} \frac{1-q^{a_{j+1}}}{a_j} \\ &= \sum_{j=0}^{i-1} \frac{1-q^{a_{j+1}}}{a_j} + \frac{1}{a_i} \sum_{j=i}^{k-1} \frac{1-(q^{a_i})^{(a_{j+1}/a_i)}}{(a_j/a_i)} \end{aligned} \tag{2.1}$$

The second term in (2.1) is the cost function of the jump sequence associated with $1 < a_{i+1}/a_i < a_{i+2}/a_i < \cdots < a_{k-1}/a_i < n/a_i$ for $q' = q^{a_i}$ and $n' = n/a_i$.

If this were not optimal we could replace it with the optimal cost function thus lowering $F(q,n)$, but this of course contradicts that we started with an optimal jump sequence. □

Theorem 2.2.4. *Let $1 < a_1 < a_2 < \cdots < a_{k-1} < n$ be an optimal jump sequence for a given value of q and n. Then for $i = 0, 1, \ldots, k-2$*

$$\sqrt{2} \leq \frac{a_{i+1}}{a_i} < \frac{19}{4},$$

In addition, if $n \leq 2$ or $q < e^{-1/e}$ then the optimal thing to do is to jump straight from 1 to n. On the other hand if $n \geq 5$ and $q > 0.9$ then there will be at least one intermediate jump between 1 and n.

The proof of Theorem 2.2.4 will be given in the next section. In this section we will give some consequences following from Theorem 2.2.4. The first consequence of Theorem 2.2.3 and Theorem 2.2.4 is that we can recursively construct $F(q,n)$ by finding the first jump. In the special case when $n = \infty$ we get the following recursive relationship.

Corollary 2.2.5. *The function $F(q,\infty)$ satisfies the following recursive relationship*

$$F(q,\infty) = \begin{cases} 1 & \text{if } q < e^{-1/e}; \\ \min\limits_{\sqrt{2}<a<19/4} \left(1 - q^a + \dfrac{1}{a} F(q^a, \infty)\right) & \text{otherwise.} \end{cases}$$

Another consequence is that the bound on the jumps gives a natural lower bound for the cost. Namely, $F(q,n)$ is at least as much as the cost of the first jump. This gives us the following result.

Corollary 2.2.6. *For a fixed $0 < q < 1$, let $n \geq \sqrt{2}$ then $F(q,n) \geq 1 - q^{\sqrt{2}}$.*

Perhaps the most important consequence of Theorem 2.2.4 is that it gives a natural bound on the number of jumps that can be taken. In particular it can be shown that the number of jumps from 1 to n is $\Theta\big(\min(\ln(n), -\ln\big(\ln(\frac{1}{q})\big))\big)$. So if we fix n and let $q \to 1$ then the number of jumps is of order $\ln(n)$. While if we fix q and let $n \to \infty$ then the number of jumps is of order $-\ln\big(\ln(\frac{1}{q})\big)$ (which is still finite, and indeed has a bounded maximum jump value).

Corollary 2.2.7. *Let $n < \infty$ and $q < 1$, and let k be the number of jumps, (the number of intermediate terms that are visited in the optimal jump sequence from 1 to n is $k-1$). Then*

$$\begin{aligned} k &\geq \min\big(0.64\ln(n) - 0.04, -0.64\ln\big(\ln(\frac{1}{q})\big) - 0.45\big), \\ k &\leq \min\big(2.89\ln(ncv), -2.89\ln\big(\ln(\frac{1}{q})\big) - 0.89\big). \end{aligned}$$

Proof. If $1 < a_1 < a_2 < \cdots < a_{k-1} < n$ is the optimal jump sequence then it follows from Theorem 2.2.4 that $\sqrt{2}^{k-1} \leq a_{k-1} \leq (4.75)^{k-1}$. To find a lower bound we note that by Theorem 2.2.4 we must continue jumping until either $(4.75)^{k-1} > n/5$ or $q^{(4.75)^{k-1}} < 0.9$. Solving for k in the first equation we have that $k > \big(\ln(n) - \ln(5)\big)/\ln(4.75) > 0.64\ln(n) - 0.04$. Solving for k in the second equation we first have that $(4.75)^{k-1}\ln(q) < \ln(0.9)$ or $(4.75)^{k-1} > \ln(0.9)/\ln(q)$, so that $k > \big(\ln\big(\ln(10/9)\big) - \ln\big(\ln(\frac{1}{q})\big)\big)/\ln(4.75) + 1 > -0.64\ln\big(\ln(\frac{1}{q})\big) - 0.45$

To find an upper bound we note that by Theorem 2.2.4 it is possible to continue jumping as long as $(\sqrt{2})^{k-2} < n/2$ and $q^{(\sqrt{2})^{k-2}} \geq e^{-1/e}$. Solving for k

in the first equation we have that $k < \big(\ln(n) - \ln(2)\big)/\ln(\sqrt{2}) + 2 < 2.89\ln(n)$. Solving for k in the second equation we first have that $(\sqrt{2})^{k-2} \leq 1/\big(e\ln(\frac{1}{q})\big)$, so that $k \leq \big(-\ln\big(\ln(\frac{1}{q})\big) - 1\big)/\ln(\sqrt{2}) + 2 < -2.89\ln\big(\ln(\frac{1}{q})\big) - 0.89$. □

Recalling the definitions of GKM trees and the jump sequences, it is easy to see that the number of jumps k gives the depth information of the (q, n)-optimal tree. Based on Corollary 2.2.7 when $n \to \infty$, the depth of a (q, n)-optimal tree is bounded by $O(\ln(\ln q^{-1})^{-1})$, and the average degree of the node in (q, n)-optimal tree is also bounded by a constant. So, Corollary 2.2.7 shows that the number of leaves under any node v is bounded by $O(\ln(q^{-1})^{-1})$, where v is a child of the root r, which is consistent with with Theorem 2.1.6.

2.2.3 Proof of Theorem 2.2.4

Here we will give a proof of Theorem 2.2.4. We will break the proof into a series of claims that will establish the upper and lower bounds. The main technique is to compare costs of different sequences and show that one is always better. We will first start with an observation which will be useful for proving the special cases.

Observation 2.2.8. *For a fixed q if the optimal jump sequence of length m beats the optimal jump sequence of length $m+1$, then the optimal jump sequence of length m beats the optimal jump sequence of length ℓ for each $\ell \geq m+1$.*

This observation follows by noting that for the optimal jump sequence of

length ℓ the associated cost of the first first $m+1$ terms is bounded by the total cost but is also at least as large as the cost of the optimal jump sequence of length $m+1$. The result then follows.

Claim 2.2.9. *If $n \leq 2$ or $q < e^{-1/e}$ then the optimal thing to do is to jump from 1 straight to n.*

Proof. From Observation 2.2.8 it suffices to show that the cost of jumping from 1 straight to n is better than the cost of jumping from 1 to x to n for all $1 < x < n$. Now suppose that $n \leq 2$, then this is equivalent to showing that

$$\frac{1-q^n}{1} < \frac{1-q^x}{1} + \frac{1-q^n}{x} \qquad \text{or} \qquad 0 < 1 - xq^x + (x-1)q^n.$$

We need to show $h_x(q) = 1 - xq^x + (x-1)q^n$ is positive in the range. Start by noting that $h_x(1) = 0$, it therefore suffices to show that $h'_x(q) < 0$ for $q < 1$ to establish the result. A calculation shows that

$$h'_x(q) = -x^2 q^{x-1} + n(x-1)q^{n-1} = xq^{n-1}\big(n - \frac{n}{x} - xq^{x-n}\big).$$

Now since $x - n < 0$ then $q^{x-n} > 1$ and we have

$$n - \frac{n}{x} - xq^{x-n} < n - \frac{n}{x} - x \leq n - 2\sqrt{n} < 0,$$

where the last step is a simple minimization problem and uses $n \leq 2$.

Now suppose that $q < e^{-1/e}$, then it is easy to check that $xq^x \leq -1/e \ln q < 1$. So $0 < 1 - xq^x + (x-1)q^n$ is easily satisfied. □

Note that Claim 2.2.9 is the real number version of Theorem 2.1.4.

Claim 2.2.10. *If $n \geq 5$ and $q > 0.9$ then there will be at least one intermediate jump between 1 and n.*

Proof. It suffices to show jumping from 1 to 2 to n gives a lower cost then jumping from 1 to n. This will hold if

$$\frac{1-q^n}{1} > \frac{1-q^2}{1} + \frac{1-q^n}{2} \qquad \text{or} \qquad 1 - 2q^2 + q^n < 0.$$

Since $1 - 2q^2 + q^n \leq 1 - 2q^2 + q^5$ it suffices to show this holds when $n = 5$. In this case we have that $f(0.9) = -0.02951$ while $f(1) = 0$ and $f''(q) = -4+20q^3 \geq 10.58$ for $0.9 < q < 1$. Combining these establishes the result. □

We now turn to establishing the upper and lower bounds. By Theorem 2.2.3 it will suffice to show that the bounds hold for the first jump.

Claim 2.2.11. *If an optimal jump sequence starts $1 < a < b < \cdots$, then $b \geq 2$.*

Proof. Since we are making an intermediate jump, by Claim 2.2.9 we may assume that $q \geq e^{-1/e} > 0.69$. Now suppose $b < 2$. Comparing the costs of jumping from 1 to b and from jumping 1 to a to b we have that the first sequence has lower cost if

$$\frac{1-q^b}{1} < \frac{1-q^a}{1} + \frac{1-q^b}{a}.$$

[All other terms associated with the cost are equal and drop out.] This is equivalent to $(a-1)q^b - aq^a + 1 > 0$. Since we assumed $b < 2$ and $a - 1 > 0$, it suffices to show this holds when $b = 2$, i.e., it suffices to show

$$f(a) = (a-1)q^2 - aq^a + 1 > 0 \quad \text{for} \quad 0.69 \leq q < 1, \text{ and } 1 < a < 2.$$

We have

$$f'(a) = q^2 - q^a - a\ln(q)\, q^a \quad \text{and} \quad f''(a) = -\ln(q)\, q^a(2 + a\ln(q)).$$

For the range given for a and q it is easy to check that $f''(a) > 0$. Since we also have that $f(1) = 1-q > 0$, it suffices to show that $f'(1) > 0$ and the result follows. Substituting, we have

$$g(q) = f'(1) = q^2 - q - q\ln(q).$$

Since we know $g(1) = 0$, to show this is positive we again do a similar trick. Namely by calculus we have

$$g'(q) = 2q - 2 - \ln(q) \quad \text{and} \quad g''(q) = 2 - \frac{1}{q}.$$

Since $g'(1) = 0$ and $g'' > 0$ in the range of q we are interested in, g is minimal at $q = 1$, i.e., $g(q) > 0$ for $0.69 < q < 1$ and so the result follows. In particular, skipping over a lowers the cost, which is a contradiction. □

Claim 2.2.12. *If an optimal jump sequence starts* $1 < a < b < \cdots$ *then* $a \geq \sqrt{2}$.

Proof. Again we may assume $q \geq 0.69$. Now suppose $a < \sqrt{2}$. we compare the costs of jumping from 1 to a to b with jumping from 1 to $\sqrt{2}$ to b, and we see that the second sequence has a lower cost if

$$\frac{1-q^{\sqrt{2}}}{1} + \frac{1-q^b}{\sqrt{2}} < \frac{1-q^a}{1} + \frac{1-q^b}{a}.$$

[Again all other terms drop out.] It again suffices to show this holds for $b = 2$, and so we want to show

$$f(a) = a\sqrt{2}(q^{\sqrt{2}} - q^a) + (\sqrt{2} - a)(1-q^2) > 0 \quad \text{for} \quad 0.69 \leq q < 1, \text{ and } 1 < a < \sqrt{2}.$$

By calculation we have

$$f'(a) = \sqrt{2}(q^{\sqrt{2}} - q^a) - a\sqrt{2}\ln(q)\, q^a - 1 + q^2 \quad \text{and} \quad f''(a) = -\sqrt{2}\ln(q)\, q^a(2 + a\ln(q)).$$

Again we have that $f'' > 0$ in the range that we are interested in. Since $f(\sqrt{2}) = 0$ it suffices to show that

$$g(q) = f'(\sqrt{2}) = -2\ln(q)q^{\sqrt{2}} - 1 + q^2 < 0.$$

Since $g(1) = 0$ to show that this is negative we again compute to get

$$g'(q) = 2q - 2\sqrt{2}\ln(q)q^{\sqrt{2}-1} - 2q^{\sqrt{2}-1} \quad \text{and}$$

$$g''(q) = 2 - 2\sqrt{2}(\sqrt{2} - 1)\ln(q)q^{\sqrt{2}-2} - (4\sqrt{2} - 2)q^{\sqrt{2}-2}.$$

Since $g'(1) = 0$ and $g'' < 0$ (the term with the $\ln(q)$ makes an insignificant contribution and the other terms then easily give a negative term) in the range that we are interested in the result follows, showing we should never take a first jump below $\sqrt{2}$. □

For the upper bound we will break the proof into two cases, namely when q is "small" and when q is "large". Of course we already know by Claim 2.2.9 that there are no intermediate jumps for $q < e^{-1/e}$ and so the upper bound holds trivially in that range.

Claim 2.2.13. *If $q \leq 0.88$ then an optimal jump sequence makes at most one intermediate jump between 1 and n, and further such a jump is bounded above by 4.75.*

Proof. By Claim 2.2.9 we can assume $q \geq 0.69$. To show that there is at most one intermediate jump it suffices to show that the optimal cost of two intermediate jumps will never beat the optimal cost of one intermediate jump. To find the optimal two intermediate jump sequence and one intermediate jump sequence, we minimize

$$q(x,y) = \frac{1-q^x}{1} + \frac{1-q^y}{x} + \frac{1-q^n}{y} \quad \text{and} \quad r(z) = \frac{1-q^z}{1} + \frac{1-q^n}{z}$$

respectively. In particular we need,

$$\begin{aligned} q_y(x,y) &= \frac{-\ln(q)q^y}{x} - \frac{1-q^n}{y^2} = \frac{-\ln(q)}{xy^2}\left(y^2q^y - \frac{x(1-q^n)}{-\ln(q)}\right) = 0, \\ r'(z) &= -\ln(q)q^z - \frac{1-q^n}{z^2} = \frac{-\ln(q)}{z^2}\left(z^2q^z - \frac{1-q^n}{-\ln(q)}\right) = 0. \end{aligned}$$

Given that we know $q \geq 0.69$ and the shape of the curve t^2q^t there will be two possible solutions for y and z, since we are trying to minimize, we will want to find the *first* such solution in each case. In particular since $x > 1$ it follows that we have $y > z$.

We then have that the one jump sequence always dominates a two jump sequence if for any x and y we can find a z so that

$$\frac{1-q^x}{1} + \frac{1-q^y}{x} + \frac{1-q^n}{y} > \frac{1-q^z}{1} + \frac{1-q^n}{z},$$

or rearranging,

$$\left(\frac{1-q^x}{1} + \frac{1-q^y}{x} + \frac{1}{y}\right) - \left(\frac{1-q^z}{1} + \frac{1}{z}\right) > \underbrace{q^n\left(\frac{1}{y} - \frac{1}{z}\right)}_{<0}.$$

So it is certainly sufficient to show that the left hand side is > 0. By using a computer algebra system it can be checked that this holds for $q \leq 0.88$.

Finally, the optimal jump will be the first solution to $z^2q^z+(1-q^n)/\ln(q) = 0$ which will certainly occur before the solution to $z^2q^z + 1/\ln(q) = 0$. Plotting $(4.75)^2q^{4.75} + 1/\ln(q)$ in the range $0.69 \leq q \leq 0.88$, we see that it shows that the solution occurs before 4.75, i.e., the optimal jump has length bounded above by 4.75. □

Claim 2.2.14. *If an optimal jump sequence starts* $1 < b < \cdots$ *and* $q \geq 0.88$ *then* $b < 4.75$.

Proof. Now suppose that an optimal jump sequence starts by jumping from 1 to b where $b > 4.75$. Then we claim that it is better to start by jumping from 1 to 2 to b. This last statement holds if

$$\frac{1-q^2}{1} + \frac{1-q^b}{2} < \frac{1-q^b}{1}.$$

[Again all other terms drop out.] Rearranging, this will be equivalent to showing $0 < 2q^2 - q^b - 1$. It suffices to show that this last statement holds for $b = 4.75$, i.e., it suffices to show

$$g(q) = 2q^2 - q^{4.75} - 1 > 0 \quad \text{for} \quad 0.88 \leq q < 1.$$

Note that $g(0.88) = 0.00393\ldots > 0$ and $g(1) = 0$. Since $g'(q) = 4q - 4.75q^{3.75}$ and $g''(q) = 4 - 17.8125q^{2.75}$ we have that the graph is concave down for our range of

q, the result then follows. This shows that we would never jump more than 4.75 in an optimal jump sequence. □

This completes the proof of Theorem 2.2.4.

2.3 Acknowledgements

The text of Chapter Two, in part, is adapted from the paper, "Optimal Jumping Pattern", which has been submitted for publication to Journal of Combinatorics and Number Theory. I would like to thank my co-authors, Steve Butler and Ronald L. Graham for allowing me to include our joint work in this thesis.

Chapter 3

Optimal GKM Trees As $n \to \infty$

3.1 Optimal Trees as $n \to \infty$

In this section, we will analyze the properties of the GKM tree as $n \to \infty$. Based on the properties, we will give an approximation algorithm to build a "good" GKM tree.

As the number of leaves $n \to \infty$, the cost of the GKM tree $C(q, T) = \sum_v d_v \cdot (1 - q^{N_v}) \to \infty$. To compare the cost value of trees as $n \to \infty$, we define a new cost function called the *cost per leaf* for tree T.

Definition 3.1.1. Recall the definition of *cost per leaf* for leaf u,

$$c(q, u) = \sum_{x \in Anc(u)} \frac{w(x)}{N_x},$$

where N_x is the number of leaves under x and $Anc(u)$ is the ancestor set of leaf u. Here we define the cost per leaf function for tree T as the average cost per leaf

values among all the leaves in T:

$$CL(T) = \frac{C(T)}{n}.$$

We also define $CL(T_1)$ as the average cost per leaf value among all the leaves of subtree T_1.

Lemma 3.1.2. *(Majority Property) If q is fixed, as the number of leaves $n \to \infty$, in a (q, n)-optimal tree, the root has an increasing number of subtrees; but, these subtrees only have a constant number of structures. There must be an optimal tree T_{opt}, in which only one tree structure show up an unbounded number of times, and we call it the dominant subtree. We call other subtree structures of T_{opt} residual subtrees.*

Proof. This lemma follows from Theorem 2.1.6 in Chapter Two. Because each subtree of the root has a constant number of leaves. When $n \to \infty$, the root has an increasing number of subtrees. Due to the definition of cost per leaf, there must be at least subtrees has the minimum cost per leaf value, so it can replace any other subtree to make a better tree. □

So, to find a "good" GKM tree as $n \to \infty$, the main task is to find the dominant subtree. Here, we assume, the dominant subtree has a uniform property (defined below). We assume that the "good" GKM trees have the "uniform property", because, from the definition of the GKM problem, each group member is identical and each has the same probability p of changing membership, we should treat them in the same manner.

Definition 3.1.3 (Uniform Property)**.** A tree has the Uniform Property if all the nodes at the same level have the same degree. In this case, the tree which has t-levels can be written in the form $T(a_1, a_2, \ldots, a_t)$. The nodes at the ith level has outgoing degree a_i. Note, the Uniform Property is a general relaxation of the degree restrictions mentioned in [22].

Lemma 3.1.4. *Denote T_d as the dominant tree of the optimal GKM tree T_{opt} as $n \to \infty$. If T_d has the Uniform Property and can be written as $T_d = T(a_1, a_2, \ldots, a_t)$, then T_d can be expressed in the concatenating form: $T_d = T(S_{4s}; S_{3r}; S_2)$, where $S_{4s} = \{\underbrace{4, 4, ...4}_{s}\}$; $S_{3r} = \{\underbrace{3, 3, ...3}_{r}\}$; $S_2 = \{2\}$ or $\emptyset$, where $s \geq 0$ and $t \geq 0$.*

Proof. Recalling Lemma 2.1.1 in Chapter Two, if a tree $T(a_1, a_2, ..., a_t)$ is an optimal tree for q and $m = \sum_{i=1}^{t} a_i$, then $T(a_i, a_{i+1})$ is optimal for $q_1 = q^{-(\sum_{k=i+2}^{t} a_k)}$. Due to Theorem 2.1.5 in Chapter Two, if T_d is an optimal tree, a_i must be 2, 3, or 4. To prove Lemma 3.1.4, we show that, when $q \geq 3^{-1/3}$, $T(2,2)$, $T(2,3)$, $T(2,4)$ or $T(3,4)$ cannot be optimal tree by finding a better alternative tree.

Case A) $T(2,2)$ cannot be an optimal tree when $n = 4$. In this case $C_1 = C(q, T(2,2)) = 2(1-q^4) + 4(1-q^2)$; while the alternate tree $T(4)$ has cost $C_2 = C(q, T(4)) = 4(1-q^4)$. It is easy to check $\Delta(C) = C_1 - C_2 = 2(1-q^2)^2 \geq 0$.

Case B) $T(2,3)$ cannot be an optimal tree when $n = 6$. In this case $C_1 = C(q, T(2,3)) = 2(1-q^6) + 6(1-q^3)$; while the two alternate trees we consider are $T(6)$ and $T(3,2)$. These have cost $C_2 = C(q, T(6)) = 6(1-q^6)$ and $C_3 = C(q, T(3,2)) = 3(1-q^6) + 8(1-q^2)$. We can see $q < \sqrt[3]{1/2}$, $\Delta(C_1) = C_1 - C_2 =$

$(2q^3-1)(q^3-1) \geq 0$; when $q \geq \sqrt[3]{1/2}$, $\Delta(C_2) = C_1 - C_3 =\geq 0$.

Case C) $T(2,4)$ cannot be an optimal tree when $n = 8$. $C_1 = C(q, T(2,4)) = 2(1-q^8)+8(1-q^4)$; while the two alternate trees we consider are $T(8)$ and $T(4,2)$. These have cost$C_2 = C(q, T(8)) = 8(1-q^8)$ and $C_3 = C(q, T(4,2)) = 4(1-q^8) + 8(1-q^2)$. We can see for $q < \sqrt[4]{1/3}$, $\Delta(C_1) = C_1 - C_2 = (3q^4-1)(q^2-1) \geq 0$; on the other hand, when $q \geq \sqrt[4]{1/3}$, $\Delta(C_2) = C_1 - C_3 = 2(q^2-1)^2(3-q^2) \geq 0$.

Case D) $T(3,4)$ cannot be an optimal tree when $n = 12$. In this case, $C_1 = C(q, T(3,4)) = 3(1-q^{12}) + 12(1-q^4)$; while the two alternate trees we consider are $T(12)$ and $T(4,3)$. These have cost $C_2 = C(q, T(12)) = 12(1-q^{12})$ and $C_3 = C(q, T(4,3)) = 4(1-q^{12}) + 12(1-q^3)$. We can see $q < \sqrt[3]{1/3}$, $\Delta(C_1) = C_1 - C_2 = 9q^{12} - 12q^3 + 3 \geq 0$; when $q \geq \sqrt[3]{1/3}$, $\Delta(C_2) = C_1 - C_3 = q^{12} - 12q^4 + 12q^3 - 1 \geq 0$. □

Now, we simplify the dominant tree in the form $T_d(\underbrace{4,4,\ldots,4}_{s},\underbrace{3,3,\ldots,3}_{r})$ as $T_d(4^s3^r)$ and $T_d(\underbrace{4,4,\ldots,4}_{s},\underbrace{3,3,\ldots,3}_{r},2)$ as $T_d(4^s3^r2)$; and simplify the *cost per leaf function* of the dominant tree $CL(T_d(4^s3^r))$ as $CL(s,r)$.

Theorem 3.1.5. *If T_d is an optimal dominant tree and can be written in the form $T_d(4^s3^r)$, then $s = 1$.*

Proof. We prove Lemma 3.1.5 by analyzing the function behavior of $CL(s,r)$. From the definition, we know $CL(s,r) = C(T_d(4^s3^r))/(4^s3^r)$.

First, we consider the dominant tree in the three special forms $T_d(4^03^r)$,

$T_d(4^0 3^{r+1})$ and $T_d(4^1 3^{r-1})$. Their *cost per leaf* values are:

$$\begin{aligned}
CL(0,r) =& (1-q^3) + \frac{1}{3}(1-q^9) + \ \ldots \ + \frac{1}{3^{r-1}}(1-q^{3^r}) + \frac{1}{3^r}(1-q^n) \\
CL(0,r+1) =& (1-q^3) + \frac{1}{3}(1-q^9) + \ \ldots \ + \frac{1}{3^r}(1-q^{3^{r+1}}) + \frac{1}{3^{r+1}}(1-q^n) \\
CL(1,r-1) =& (1-q^3) + \frac{1}{3}(1-q^9) + \ \ldots \ + \frac{1}{3^{r-1}}(1-q^{3^{r-1}4}) + \frac{1}{3^{r-1}4}(1-q^n)
\end{aligned}$$

As $n \to \infty$, we have $q^n \to 0$. We now compare the above three *cost per leaf* functions.

Case A) $CL(0,r) \le CL(0,r+1)$ if and only if $Q_r \le \alpha^{\frac{1}{3}}$, where $Q_r = q^{3^{r-1}}$ and $\alpha = (\frac{1}{3})^{\frac{1}{3}} \approx 0.6934$.

Case B) $CL(0,r) \le CL(1,r-1)$ if and only if $Q_r^4 - Q_r^3 - \frac{1}{12} \le 0$. The function $f(x) = x^4 - x^3 - \frac{1}{12}$ has two positive solutions $\beta \approx 0.8760$ and $\gamma \approx 0.5586$. Since we only need to consider the case $q \ge (\frac{1}{3})^{\frac{1}{3}}$, which implies $Q_r \ge (\frac{1}{3})^{\frac{1}{3}}$, so we can ignore γ. We have $CL(0,r) \le CL(1,r-1)$ if and only if $Q_r \le \beta$.

Case C) $CL(1,r-1) \le CL(0,r+1)$ if and only if $\frac{1}{3}Q_r^9 - Q_r^4 + Q_r^3 - \frac{7}{36} \le 0$. The function $f(x) = \frac{1}{3}x^9 + x^4 - x^3 - \frac{7}{36}$ has only one positive solution $\delta \approx 0.8902$ in the interval $[0,1]$. So, $CL(1,r-1) \le CL(0,r+1)$ if and only if $Q_r \le \delta$.

Comparing the values of α, β and δ, we get

$$\alpha < \beta < \alpha^{\frac{1}{3}} < \delta < \beta^{\frac{1}{3}} < \alpha^{\frac{1}{9}} < \delta^{\frac{1}{3}} < \beta^{\frac{1}{9}} < \alpha^{\frac{1}{27}} < \delta^{\frac{1}{9}} < \cdots$$

If we only consider the trees in the form $T_d(4^0 3^i)$ and $T_d(4^1 3^{i-1})$ as our candidate best trees, where $i \ge 1$. As q increases from $(\frac{1}{3})^{\frac{1}{3}}$ to 1, $T_d(4^0 3^1)$, $T_d(4^1 3^0)$, $T_d(4^0 3^2)$, $T_d(4^1 3^1)$ et al. turn out to be the best trees sequentially:

(i) When $\alpha \le q < \beta$, $T_d(4^0 3^1)$ is the best tree.

(ii) When $(\beta)^{(\frac{1}{3})^i} \leq q < (\delta)^{(\frac{1}{3})^i}$, $T_d(4^1 3^i)$ is the best tree, where $i \geq 0$.

(iii) When $(\delta)^{(\frac{1}{3})^i} \leq q < (\beta)^{(\frac{1}{3})^{i+1}}$, $T_d(4^0 3^{i+2})$ is the best tree, where $i \geq 0$.

Second, we will show other trees in the form $T(4^s 3^r)$ where $s > 1$ cannot be the optimal dominant tree. We compare $CL(s, r)$ and $CL(s+1, r)$ for a fixed r.

$$\begin{aligned} CL(s,r) =&(1-q^3) + \frac{1}{3}(1-q^9) + \ \cdots \ + \frac{1}{3^{r-1}}(1-q^{3^r}) + \frac{1}{3^r}(1-q^{3^r 4}) + \ \cdots \\ &+ \frac{1}{3^r 4^{s-1}}(1-q^{3^r 4^s}) + \frac{1}{3^r 4^s}(1-q^n) \\ CL(s+1,r) =&(1-q^3) + \frac{1}{3}(1-q^9) + \ \cdots \ + \frac{1}{3^{r-1}}(1-q^{3^r}) + \frac{1}{3^r}(1-q^{3^r 4}) + \ \cdots \\ &+ \frac{1}{3^r 4^s}(1-q^{3^r 4^{s+1}}) + \frac{1}{3^r 4^{s+1}}(1-q^n) \end{aligned}$$

As $n \to \infty$, we have $CL(s, r) \leq CL(s+1, r)$ if and only if $q^{3^r 4^{s+1}} \leq \frac{1}{4}$. Now, we can see for a fixed r, as q increases from 0 to 1, among all the dominant trees in the form $T_d(4^x 3^r)$, $T_d(4^0 3^r)$ is the first best dominant tree structure, then $T_d(4^1 3^r)$ becomes best at the point $q = (\frac{1}{4})^{\frac{1}{3^r 4^1}}$, then $T_d(4^2 3^r)$ becomes the best at point $q = (\frac{1}{4})^{\frac{1}{3^r 4^2}}$, and so on. For a fixed r, $T_d(4^{s+1} 3^r)$ is better than $T_d(4^s 3^r)$ if and only if $q > \epsilon^{\frac{1}{4^s 3^r}}$, where $\epsilon = (\frac{1}{4})^{\frac{1}{4}}$.

Now, we will prove the trees in the form $T(4^s 3^r)$ where $s > 1$ cannot be the optimal dominant tree by induction.

We consider the three intervals $\alpha < q < \beta$, $\beta \leq q < \delta$ and $\delta \leq q < \beta^{\frac{1}{3}}$ as the base cases. In the interval $\alpha < q < \beta$, from (i), we know $T_d(4^0 3^1)$ is the best among the trees in the form $T_d(4^0 3^r)$ and $T_d(4^1 3^r)$. From the above analysis, we also get $q^{4^i 3^r} (i \geq 1)$ can only beat $T_d(4^0 3^1)$ as $q \geq \epsilon^{\frac{1}{3}}$. However, $\epsilon^{\frac{1}{3}} \approx 0.8909 > \beta$.

So, $T_d(4^0 3^1)$ remains the best in this interval. Similarly, it is easy to check the statement holds for the other two intervals.

As for the interval $q \in [\beta^{(\frac{1}{3})^i}, \delta^{(\frac{1}{3})^i})$, where $i \geq 1$, we first assume the statement holds for $i = k$. When $i = k+1$, we know $T_d(4^1 3^{k+1})$ is the best among the trees $T_d(4^0 3^r)$ and $T_d(4^1 3^r)$ in the interval $q \in [\beta^{(\frac{1}{3})^{k+1}}, \delta^{(\frac{1}{3})^{k+1}})$. $T_d(4^s 3^{k+1})$ $(s \geq 2)$ can only beat $T_d(4^1 3^{k+1})$ when $q \geq \epsilon^{\frac{1}{4^1 3^{k+1}}}$. However, $(\epsilon^{\frac{1}{4}})^{\frac{1}{3^{k+1}}} \approx 0.917^{\frac{1}{3^{k+1}}} > \delta^{\frac{1}{3^{k+1}}}$. From the above analysis, we see that for any s and $t \geq k+1$, $T_d(4^s 3^t)$ cannot beat $T_d(4^1 3^{k+1})$ in this interval. Now, we want to show the trees in the form $T(4^s 3^t)$, where $n < k+1$, cannot be a better tree. We prove it by contradiction. If there is a tree $T(4^m 3^t)$ with a better cost in the interval, we have

$$\begin{aligned} CL(s,t) &= 1 - q^3 + \frac{1}{3}(1 - q^{3^2}) + \ldots + \frac{1}{3^t}(1 - q^{3^t 4}) + \ldots + \frac{1}{3^t 4^s}(1 - q^n) \\ CL(1,k+1) &= 1 - q^3 + \frac{1}{3}(1 - q^{3^2}) + \ldots + \frac{1}{3^{k+1}}(1 - q^{3^{k+1} 4}) + \frac{1}{3^{k+1} 4}(1 - q^n) \end{aligned} \tag{3.1}$$

We set $q_0 = q^3$, and q_0 is in the interval $[\beta^{(\frac{1}{3})^k}, \delta^{(\frac{1}{3})^k}]$. We can rewrite the above two equations as a function of q_0:

$$\begin{aligned} CL(s,t) &= 1 - q_0 + \frac{1}{3} CL(s, t-1; q_0) \\ CL(1,k+1) &= 1 - q_0 + \frac{1}{3} CL(1, k; q_0). \end{aligned} \tag{3.2}$$

Let $CL(s, t-1; q_0)$ stand for the value of $CL(s, t-1)$ when $q = q_0$. From the assumption $CL(s,t) < CL(1, k+1)$, we get $CL(s, t-1) < CL(1,k)$ when $q = q_0$. This is a contradiction, since we assumed that $CL(1,k)$ is the best tree in the interval $[\beta^{(\frac{1}{3})^k}, \delta^{(\frac{1}{3})^k})$.

A similar induction proof works for the case $q \in [\delta^{\frac{1}{3^i}}, \beta^{(\frac{1}{3})^{i+1}}]$, when $i \geq 1$.

The proof is completed. □

Table 3.1: The best dominant tree structures.

q	$(0,\alpha)$	$[\alpha,\beta)$	$[\beta,\delta)$	$[\delta,\beta^{1/3})$	$[\beta^{1/3},\delta^{1/3})$	$[\delta^{1/3},\beta^{1/3^2})$	$\ldots$
T_d	$T_d(1)$	$T_d(3)$	$T_d(4)$	$T_d(3^2)$	$T_d(4\cdot 3)$	$T_d(3^3)$	$\ldots$

Theorem 3.1.6. *As $n \to \infty$, if $T_d = T_d(4^s3^r)$ is an optimal dominant tree, then we can find the best T_d structure by looking up Table 3.1 based on the value of q:*

(i) When $\alpha \le q < \beta$, $T_d(4^03^1)$ is the best tree.

(ii) When $(\beta)^{(\frac{1}{3})^i} \le q < (\delta)^{(\frac{1}{3})^i}$, $T_d(4^13^i)$ is the best tree, where $i \ge 0$.

(iii) When $(\delta)^{(\frac{1}{3})^i} \le q < (\beta)^{(\frac{1}{3})^{i+1}}$, $T_d(4^03^{i+2})$ is the best tree, where $i \ge 0$.

Here, α, β and δ are the same values as defined in the proof of Theorem 3.1.5.

Proof. This is a direct result from the proof of Theorem 3.1.5. □

Corollary 3.1.7. *If T_d is an optimal dominant tree, then it can expressed in the form $T_d(a_0, 3, \ldots, 3, a_t)$, where $a_0 = 4$ or $\emptyset$; $a_t = 2$ or $\emptyset$.*

Proof. This result follows from Lemma 3.1.4 and Theorem 3.1.5. □

Algorithm 2 *CBDT (Constructing Best Dominant Trees with Uniform Property)*
Input: q

Output: The Best Dominant Tree having the Uniform Property for q

$q' = q^2$

Find the dominant tree structure $T_{d1}(4^{s_1}3^{r_1})$ by looking up Table 3.1 using q'

Find the dominant tree structure $T_{d2}(4^{s_2}3^{r_2})$ by looking up Table 3.1 using q

if $CL(q', T_{d1}) < CL(q, T_{d2})$ **then**

Return $T'_{d1}(4^{s_1}3^{r_1}2)$

else

Return $T_{d2}(4^{s_2}3^{r_2})$

end if

3.2 Approximation Algorithm to Construct GKM Trees

Based on Corollary 3.1.7, we give a constant time algorithm to find the best dominant tree having the Uniform Property.

Due to Theorem 2.1.4 in Chapter Two, we know that a star is the optimal tree when $q \in (0, (\frac{1}{3})^{-\frac{1}{3}})$. Here, for a given value $q \in [(\frac{1}{3})^{-\frac{1}{3}}, 1)$ and a given number of leaves n, we show how to construct an approximate tree $GLR(q, n)$. And we will show the approximate tree performs very well by experimental results.

The idea of the construction is: first, find the dominant tree T_d by algorithm 3 for q; then build a GKM tree using the dominant tree as often as possible and

one "leftover" tree. If the number of leaves (size) of the dominant tree T_d returned by Algorithm 3 is bigger than n, we use the the largest feasible dominant tree in Table 3.1 with size less than n.

Algorithm 3 *GLR (Constructing an Approximate GKM Tree)*

Input: q and n

Output: An approximate GKM Tree

Call Algorithm 3 to find the best dominant tree T_d

The root of $GLR(q, n)$ contains $n_1 + 1$ subtrees: n_1 T_d's and one $GLR(q, L_1)$ where $n_1 = n$ `DIV` $|T_d|$ and $L_1 = n$ `MOD` $|T_d|$

Return $GLR(q, n)$

Take $n = 29$ and $q = 0.9$ as an example. By looking up the Table 3.1, we find the best dominant tree $T_d(3^2)$ with 9 leaves. The root of $GLR(0.9, 29)$ has $\lfloor 29/9 \rfloor + 1 = 4$ subtrees, three of which are $T_d(3^2)$ trees and one of which is a "leftover" su ructure of GLR(0.9,

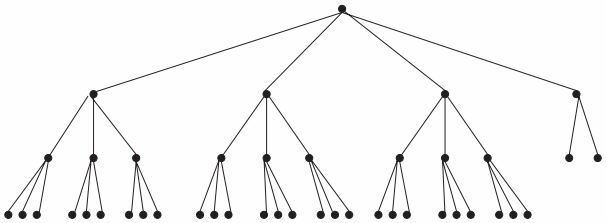


Figure 3.1: The approximate tree generated by GLR when $n = 29$, $q = 0.9$.

We can run simulations on the performance of GLR for various values of

q and n. Define $Ratio(q, n) = \text{GLR}(q, n)/\text{OPT}(q, n)$. Figure 3.2 and Figure 3.3 show $Ratio(q, n)$ as a function of n for $q = 0.9$ and $q = 0.999$ respectively. Within the simulation range, we see that for a fixed value q, the Ratio curve oscillates, but tends to 0 as n gets larger. For $q = 0.1$ the maximum ratio is below 1.02, when n is in the range $[50, 250]$. Notice that each curve has some dips and is not monotonically decreasing with q.

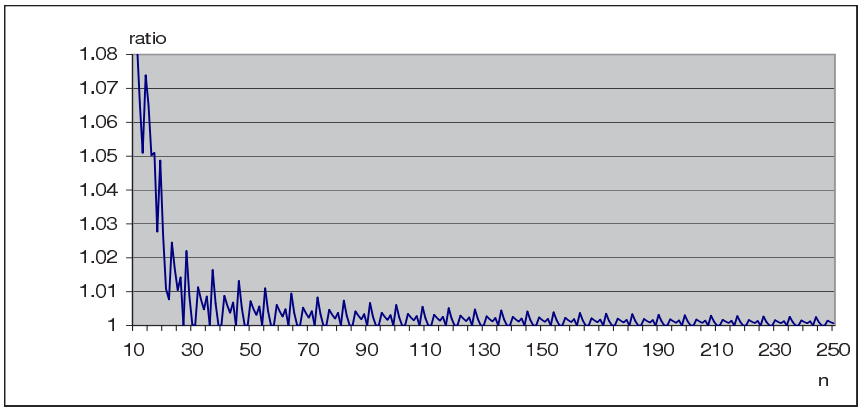


Figure 3.2: Simulation results on $Ratio(q, n)$ for $q = 0.9$

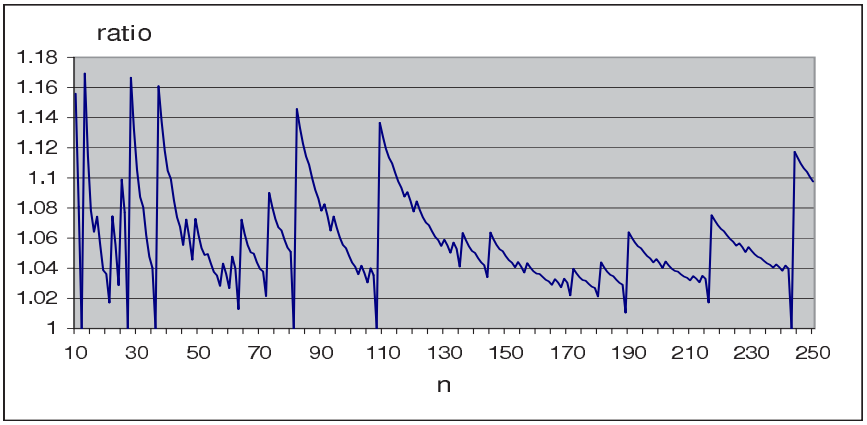


Figure 3.3: Simulation results on $Ratio(q, n)$ for q = 0.999.

Chapter 4

Optimal GKM Trees as $q \to 1$

In this chapter, we will discuss how to build a "good" GKM tree for the limiting case $q \to 1$. Algorithm 1 takes $O(nK + K^4)$ time to build an optimal GKM tree. However, $K \to \infty$ as $q \to 1$, since $K = 4(\log q^{-1})^{-1}$, so the time complexity of Algorithm 1 is unbounded as $q \to 1$.

In this chapter, we will first introduce the properties of optimal GKM trees and the algorithm to construct an optimal GKM tree given by Graham et al. [23]. Later, we will modify the definition of the Jump Sequence Problem for the case $q \to 1$, and analyze the properties of the jumping sequences, which will give a lower bound of the optimal GKM tree. We will conclude this chapter by giving a linear time $2.01(1 + \frac{1}{\lfloor \log_3 n \rfloor})$-approximation algorithm.

4.1 Related Work

We first define a new cost function for GKM trees for the limiting case $q \to 1$. Recall that, given a fixed q and n, the cost of a GKM Tree T is defined as $C(q,T) = \sum_{v \in V} (1 - q^{N(v)}) = \sum_{e \in E} (1 - q^{L(e)})$, where $N(v)$ is the number of leaves under node v and $L(e)$ is the number of leaves underneath the edge e. When $q = 1$, the cost value of any tree T is 0. Since the slope of $C(q,T)$, $C'(q,T)|_{q=1} = -\sum_{e \in E} L(e)$, is negative, we will say the optimal GKM tree as $q \to 1$ is the tree which has the smallest value of $\sum_{e \in E} L(e)$. However, there may be some ties of the function values among the tree structure. That is, two or more tree structures might give the same value of $\sum_{e \in E} L(e)$, so second order derivatives have to be considered. It is shown that we can always find an optimal GKM tree as $q \to 1$ by comparing the second order derivatives.

Definition 4.1.1. Denote the function $-C'_q(q,T)|_{q=1}$ as λ_T,

$$\lambda_T = \sum_{e \in E} L(e).$$

Let $\lambda^*(n)$ be the smallest possible value of λ_T over all the trees having n leaves.

The basic recurrence that $\lambda^*(n)$ satisfies is:

$$\lambda^*(n) = \min_{2 \le m \le n} \{m \cdot n + \sum \lambda^*(i_k)\}$$

where the sum is taken over all $i_k \ge 1$ such that $i_1 + \cdots + i_m = n$ and m denotes the degree of the root r. Based on this recurrence, Graham et al. [23] first determined

the exact value of $\lambda^*(n)$ for all values of n by comparing the value of $\lambda^*(n)$ for different m.

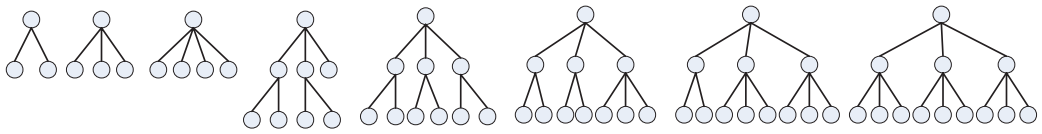


Figure 4.1: The optimal GKM trees for $n = 2, \ldots, 9$.

We first the optimal GKM tree for $n = 2, \ldots, 9$ (Figure 4.1), and calculate the corresponding values of $\lambda^*(n)$ by hand:

Table 4.1: $\lambda^*(n)$ values for $n = 2, \ldots, 9$.

n	1	2	3	4	5	6	7	8	9
$\lambda^*(n)$	0	4	9	16	23	30	38	46	54

For integers $t \geq 0$, define the intervals $I_t = \{3^t, 3^t + 1, \ldots, 2 \cdot 3^t\}$ and $J_t = \{2 \cdot 3^t, 2 \cdot 3^t + 1, \ldots, 3^{t+1}\}$. It is easy to see that $\lambda^*(n)$ is linear on I_0, I_1, J_0 and J_1.

Theorem 4.1.2. *Extend $\lambda^*(n)$ to a real function $\lambda^*(x)$ for all $x \geq 1$ by linear interpolation (See Figure 4.2). Then $\lambda^*(x)$ satisfies*

$$\lambda^*(x) = \begin{cases} (3t+4)x - 4 \cdot 3^t & \text{if} \quad 3^t \leq x \leq 2 \cdot 3^t, \\ (3t+5)x - 6 \cdot 3^t & \text{if} \quad 2 \cdot 3^t \leq x \leq 3^{t+1}. \end{cases}$$

As mentioned earlier, if for any number of leaves n, there is a unique value m^* such that only the tree structures with root degree m^* can reach the minimum

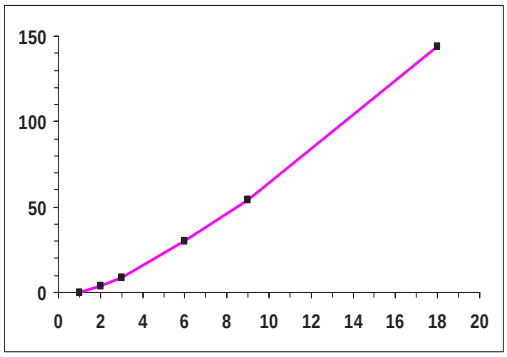


Figure 4.2: Graph of $\lambda^*(n)$.

cost $\lambda^*(n)$. We can construct the optimal GKM trees recursively by only considering the trees with root degree m^*. However, one can show, in the following cases, the trees having the minimum cost value $\lambda^*(n)$ can have different root degrees.

Case A) When $4 \cdot 3^{t-1} < n \leq 6 \cdot 3^{t-1}$, the root degree m could be 2 or 3 as the tree structure reaches the minimum cost $\lambda^*(n)$.

Case B) When $n = 4 \cdot 3^{t-1}$, the root degree m could be 2, 3 or 4 as the tree structure reaches the minimum cost $\lambda^*(n)$.

Let $T^*(n)$ be the optimal tree structure with n leaves as $q \to 1$. Although the slope value $\lambda^*(n)$ of $C(q, T^*)$ is determined by n, the root degree of the optimal GKM tree cannot be decided. For the above two "ambiguous" cases, the second derivative of $C(q, T)$ has to be considered.

Definition 4.1.3. Denote the function $C''_q(q, T)|_{q=1}$ as μ_T:

$$\mu_T = 2 \sum_{e \in E} \binom{L(e)}{2}.$$

Let $\mu^*(n)$ denote the largest possible value of this sum over all trees $T(n)$ for which $\lambda_{T(n)} = \lambda^*(n)$.

By comparing the values of $\mu_T(n)$ in Case A) and Case B), it can be shown the root degree of $T^*(n)$ is always 3 except for $n = 4 \cdot 3^t$ and $n = 2$.

Theorem 4.1.4. *As $q \to 1$, the (q, n)-optimal tree $T^*(n)$ always has root degree 3 except for $n = 4 \cdot 3^t$, in which case $T^*(n)$ has root 4, and for $n = 2$, when $T^*(2)$ has root degree 2.*

4.2 Jump Sequences as $q \to 1$

In this section, we will consider the relaxed version of the GKM tree problem introduced in Chapter Two, and will provide one lower bound and one upper bound for the jumping sequences from 1 to n. Based on the result given in this section, we will give a linear time approximation algorithm to build an approximate GKM tree as $q \to 1$.

As $q \to 1$, we know $F(q, n) \to 0$, i.e., all the jumps have cost 0. So for this limiting case we should consider a modified cost function. To determine which cost function, we start by recalling $q = 1 - p$ and note that $1 - q^x = px + O(p^2)$ by the Binomial Theorem. Thus, $F(q, n)$ proposed in Chapter Two becomes

$$F(q, n; a_1 a_2, \ldots, a_{k-1}) = \sum_{i=0}^{k-1} \frac{1 - q^{a_{i+1}}}{a_i} = p \sum_{i=0}^{k-1} \frac{a_{i+1}}{a_i} + O(p^2).$$

The obvious candidate is to use the first order term as our new cost function, i.e., the cost of a jump from a to b will be given by b/a. Note this is consistent with the cost function of trees as $q \to 1$. Given a sequence of jumps, the total cost of moving from 1 to n will be given by

$$\begin{aligned} G(n; a_1, a_2, \ldots, a_{k-1}) &= \frac{a_1}{1} + \frac{a_2}{a_1} + \cdots + \frac{a_{k-1}}{a_{k-2}} + \frac{n}{a_{k-1}} \\ &= \sum_{i=0}^{k-1} \frac{a_{i+1}}{a_i}, \end{aligned}$$

and we will denote the minimal cost of jumping from 1 to n by

$$G(n) = \min_k \min_{1<a_1<a_2<\cdots<a_{k-1}<a_n} G(n; a_1, a_2, \ldots, a_{k-1}).$$

In the next two subsections, we will analyze properties of these jumping sequences as $q \to 1$.

4.2.1 Jumping along the Reals

We begin by noting that by the arithmetic-geometric mean inequality, for any jump sequence we have

$$G(n; a_1, a_2, \ldots, a_{k-1}) \geq k \sqrt[k]{\frac{a_1}{1}\frac{a_2}{a_1} \cdots \frac{a_{k-1}}{a_{k-2}}\frac{n}{a_{k-1}}} = k\sqrt[k]{n},$$

with equality holding if and only if $a_1/1 = a_2/a_1 = a_3/a_2 = \cdots$. Note that for any $n > 1$ the function $f(x) = xn^{1/x}$ is concave up for $x > 0$ and so has a unique minimum which occurs at $f(\ln n) = e \ln n$.

Lemma 4.2.1. *When jumping along the reals we have that*

$$G(n) = \min\left\{\lfloor \ln n\rfloor n^{1/\lfloor \ln n\rfloor}, \lceil \ln n\rceil n^{1/\lceil \ln n\rceil}\right\} = e\ln n + O\big(\frac{1}{\ln n}\big).$$

While the optimal jumping patterns are formed by geometric sequences with ratio $e^{1+o(1)}$.

Proof. The form of $G(n)$ and the formation of a geometric series with a ratio of either $n^{1/\lfloor \ln n\rfloor} = e^{\ln n/\lfloor \ln n\rfloor} = e^{1+o(1)}$ or $n^{1/\lceil \ln n\rceil} = e^{\ln n/\lceil \ln n\rceil} = e^{1+o(1)}$ follows from the statements preceding the proposition. It remains to verify the asymptotic behavior. So suppose that $\lfloor \ln n\rfloor = \ln n - \alpha$ then we have

$$\begin{aligned}
\lfloor \ln n\rfloor n^{1/\lfloor \ln n\rfloor} - e\ln n &= \big(\ln n - \alpha\big)e^{\ln n/(\ln n-\alpha)} - e\ln n \\
&= e\ln n\big(e^{\alpha/(\ln n-\alpha)} - 1\big) - \alpha e\, e^{\alpha/(\ln n-\alpha)} \\
&= e\ln n\left(\frac{\alpha}{\ln n-\alpha} + O\big(\frac{1}{\ln n^2}\big)\right) - \alpha e + O\big(\frac{1}{\ln n}\big) = O\big(\frac{1}{\ln n}\big).
\end{aligned}$$

Now suppose that $\lceil \ln n\rceil = \ln n + \alpha$ then we have

$$\begin{aligned}
\lceil \ln n\rceil n^{1/\lceil \ln n\rceil} - e\ln n &= \big(\ln n + \alpha\big)e^{\ln n/(\ln n+\alpha)} - e\ln n \\
&= e\ln n\big(e^{-\alpha/(\ln n+\alpha)} - 1\big) + \alpha e\, e^{-\alpha/(\ln n+\alpha)} \\
&= e\ln n\left(\frac{-\alpha}{\ln n+\alpha} + O\big(\frac{1}{\ln n^2}\big)\right) + \alpha e - O\big(\frac{1}{\ln n}\big) = O\big(\frac{1}{\ln n}\big).
\end{aligned}$$

The proof is completed. □

4.2.2 Jumping along the Integers

We will let $G_{\mathbb{Z}}(n)$ denote the minimal cost for the case of when we restrict jumps to the integers. Clearly, $G_{\mathbb{Z}}(n) \geq G(n) = e\ln n + O(1/\ln n)$. However, with

the restriction of jumping only on the integers we should expect the cost to go up. In this section will look at the asymptotic behavior of $G_{\mathbb{Z}}(n)$. First though we will introduce an important integer sequence, and some of its properties, that will play a role in the behavior of the function.

Lemma 4.2.2. *Let $a(n)$ be the integer sequence such that $a(0) = 1$ and $a(n) = \lfloor e{\cdot}a(n-1) + 0.5 \rfloor$ for $n \geq 1$ (i.e., to get the next term multiply by e and round). Then the following holds:*

- $a(n) = \lfloor \gamma e^n + 0.5 \rfloor$ *for* $\gamma = 1.098002099366832827899136351\ldots$
- $\lim\limits_{n\to\infty} \left(\sum\limits_{i=1}^{n} \dfrac{a(i)}{a(i-1)} - e \ln\big(a(n)\big) \right) = \alpha = 0.014357537447198206167909857\ldots$

This sequence, which starts $\{1, 3, 8, 22, 60, 163, 443, 1204, 3273, 8897, \ldots\}$, is `A024581` in the `OEIS` [24]. Essentially what it does is try to best approximate jump lengths of e where after every jump forward we reset (i.e., as compared to taking the nearest integer to powers of e which would be the sequence $\{1, 3, 7, 20, 55, 148, 403, 1097\ldots\}$).

Proof. For the first part we let $b(n) = a(n)/e^n$. We first show that this sequence converges. For $n \geq 1$,

$$\begin{aligned} |b(n) - b(n-1)| &= \left| \frac{a(n)}{e^n} - \frac{a(n-1)}{e^{n-1}} \right| \\ &= \left| \frac{\lfloor e{\cdot}a(n-1) + 0.5 \rfloor}{e^n} - \frac{e{\cdot}a(n-1)}{e^n} \right| \\ &= e^{-n} \big| \lfloor e{\cdot}a(n-1) + 0.5 \rfloor - e{\cdot}a(n-1) \big| \;\leq\; \frac{1}{2} e^{-n}, \end{aligned}$$

where the last step follows from noting that no number is more than $1/2$ away from the nearest integer (this is what the inside of the absolute value is expressing). This implies the sequence is Cauchy and so must converge. Let $\gamma = \lim_{n\to\infty} b(n)$. Then note that for any n

$$\big|b(n) - \gamma\big| \le \sum_{k=n+1}^{\infty} \big|b(k) - b(k-1)\big| \le \sum_{k=n+1}^{\infty} \frac{1}{2}e^{-k} = \frac{\frac{1}{2}e^{-(n+1)}}{1-\frac{1}{e}} < \frac{0.3}{e^n}.$$

Multiplying both sides by e^n, this implies that $\big|a(n) - \gamma e^n\big| < 0.3$. Since $a(n)$ is an integer and γe^n is less than 0.3 away it must be that $a(n)$ is the nearest integer to γe^n. From this it can be shown that $1.05e^n < a(n) < 1.15e^n$.

For the second part let

$$c(n) = \sum_{i=1}^{n} \frac{a(i)}{a(i-1)} - e\ln\big(a(n)\big),$$

and consider the following

$$\begin{aligned} c(n) - c(n-1) &= \frac{a(n)}{a(n-1)} - e\ln\left(\frac{a(n)}{a(n-1)}\right) \\ &\le \frac{e{\cdot}a(n-1)+0.5}{a(n-1)} - e\ln\left(\frac{e{\cdot}a(n-1)-0.5}{a(n-1)}\right) \\ &= \frac{1}{2a(n-1)} - e\ln\left(1 - \frac{1}{2e{\cdot}a(n-1)}\right) \\ &\le \frac{1}{2a(n-1)} + \frac{1}{a(n-1)} \;\le\; \frac{3e}{2e^n}. \end{aligned}$$

(Here we used the fact that for $0 \le x \le 1/2e$ that $-\ln(1-x) \le 2x$.) On the other hand it is easy to see that $c(n)$ is increasing, i.e., since $x - e\ln x$ has a minimum value of 0 at $x = e$, so $c(n) - c(n-1) \ge 0$. Combining we have $\big|c(n) - c(n-1)\big| \le 3e/2e^n$ from which the convergence of $c(n)$ easily follows. This shows that $c(n)$ converges

to α with an error of order e^{-n}, a little more careful analysis can show that the error is at most $0.05e^{-2n}$.

The numerical approximation for γ and α can be found by the computing the corresponding terms for sufficiently large n. □

Theorem 4.2.3. *When jumping along the integers we have that*

$$G_{\mathbb{Z}}(n) \leq e \ln n + \alpha + O\big(\frac{1}{\ln n}\big),$$

where α is the constant introduced in Lemma 4.2.2.

Proof. We construct an approximate optimal sequence for n by letting $\ell = \lfloor \ln \ln n \rfloor$ and then take the first ℓ terms from the integer sequence from Lemma 4.2.2 to form the initial part of the sequence, $a(1), a(2), \ldots, a(\ell)$. From the proof of Lemma 4.2.2 we have that $0.38 \ln n \leq a(\ell) \leq 1.15 \ln n$, i.e., we have that $a(\ell) = \theta(\ln n)$. Now starting at $a(\ell)$ find the optimal jump sequence jumping to n along the *real* numbers and then round each term to the nearest integer, to form the rest of the sequence. We denote this remaining part of the sequence by $b(0) = a(\ell), b(1), b(2), \ldots, b(k), b(k+1) = n$.

We now need to bound the cost of the jump sequence $a(1)$, ..., $a(\ell)$, $b(1)$, ..., $b(k)$.

From Lemma 4.2.2, the first part of the jump sequence has cost bounded by $e \ln(a(\ell)) + \alpha - o(1)$. It is also easy to adapt the proof of Proposition 4.2.1 to see that the cost of the optimal jump sequence along the reals from $a(\ell)$ to N

is $e \ln\big(n/a(\ell)\big) + O(1/\ln n)$. It remains to show that the error from rounding the terms to the nearest integer is of order $O(1/\ln n)$.

By Proposition 4.2.1 we know that the sequence $b(i) = \lfloor a(\ell)\beta^i + 0.5 \rfloor$ where $\beta \approx e$. A simple calculation shows that

$$\left|\beta - \frac{b(i+1)}{b(i)}\right| \leq \frac{\beta+1}{2b(i)} \leq \frac{\beta+1}{a(\ell)\beta^i}.$$

From this it follows that the error we get from rounding to the nearest integer is bounded by

$$\sum_{i=0}^{k} \left|\beta - \frac{b(i+1)}{b(i)}\right| \leq \sum_{i=0}^{\infty} \frac{\beta+1}{a(\ell)\beta^i} = \frac{\beta(\beta+1)}{a(\ell)(\beta-1)} = O\big(\frac{1}{\ln n}\big).$$

□

4.3 An Approximation Algorithm to Build GKM Trees

In this section, we design a approximation algorithm, namely the LR algorithm, to build GKM trees. LR maintains an almost balanced ternary tree (i.e., the depth of any two leaves differ by at most 1) in which at most one internal node has degree two. Moreover, LR adds new leaves incrementally in a left to right order. Figure 4.3 shows the tree we get by using LR for $n = 2, \ldots, 9$. We can also recursively build a key tree using LR in the following way. For a tree with $n \geq 3$ leaves, the number of leaves in the root's three subtrees is decided by the table

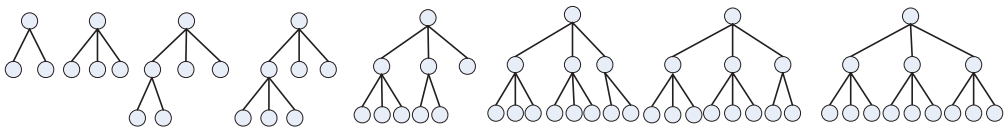


Figure 4.3: Trees generated by LR for $n = 2, \ldots, 9$.

below; while for a tree with 2 leaves, the tree structure is a root with two children (a star).

Table 4.2: LR Algorithm.

	Leaves (Left)	Leaves(Middle)	Leaves (Right)
$3^t \leq n < 5 \cdot 3^{t-1}$	$n - 2 \cdot 3^{t-1}$	3^{t-1}	3^{t-1}
$5 \cdot 3^{t-1} \leq n < 7 \cdot 3^{t-1}$	3^t	$n - 4 \cdot 3^{t-1}$	3^{t-1}
$7 \cdot 3^{t-1} \leq n < 3^{t+1}$	3^t	3^t	$n - 2 \cdot 3^t$

We denote the tree with n leaves constructed by LR as T_n. Note that the structure of the ternary tree can be decided in $\log n$ time because every time we go down the tree, there is at most one subtree whose number of leaves is not a power of 3 and needs further calculation.

Let $\mathrm{LR}(q, n)$ denote the cost of the ternary tree constructed by LR for given n and q. To obtain an upper bound for $\mathrm{LR}(q, n)$, we first prove the following lemmas.

Lemma 4.3.1. *The inequality* $LR(q, n) < LR(q, n + 1)$ *holds for all* $n > 0$ *and* $0 < q < 1$.

Proof. We view $C(T_n)$ as the cost of T_n for given q, which is defined in Chapter

One and compare the cost of the corresponding nodes $w(u)$ and $w(u')$ in T_n and T_{n+1} respectively. Assuming $w(u) = d_u \cdot (1 - q^{N_u})$, we have $w(u') = w(u)$, $w(u') = d_u \cdot (1 - q^{N_u+1})$ or $w(u') = (d_u + 1) \cdot (1 - q^{N_u+1})$. Therefore, for any node u of T and its corresponding node u' of T_{n+1}, we have $w(u) \leq w(u')$. This proves the lemma. □

Lemma 4.3.2. *For any integer $t > 0$ and $0 < q < 1$, we have $\frac{1-q^{3^t}}{3^{t-1}} > \frac{1-q^{3^{t+1}}}{3^t}$.*

Proof. Note that $3\sum_{i=1}^{3^t} q^{i-1} > (1+q^{3^t}+q^{2\cdot 3^t})\sum_{i=1}^{3^t} q^{i-1} = \sum_{i=1}^{3^{t+1}} q^{i-1}$. The lemma is proved by multiplying $\frac{1-q}{3^t}$ on both sides. □

Lemma 4.3.3. *For any integer $t > 0$ and $0 < q < 1$, we have*

$$LR(q, 3^{t+1}) < 3(1 + \frac{1}{t})LR(q, 3^t).$$

Proof. From the description of the LR algorithm, we know LR is a complete ternary tree when $n = 3^t$ and $n = 3^{t+1}$.

$$\begin{aligned} LR(q, 3^{t+1}) &= 3^{t+1}(1 - q^3 + \frac{1}{3}(1 - q^{3^2}) + \cdots + \frac{1}{3^{t-1}}(1 - q^{3^t}) + \frac{1}{3^t}(1 - q^{3^{t+1}})) \\ &< 3^{t+1}(1 + \frac{1}{t})(1 - q^3 + \frac{1}{3}(1 - q^{3^2}) + \cdots + \frac{1}{3^{t-1}}(1 - q^{3^t})) \\ &= 3(1 + \frac{1}{t})LR(q, 3^t). \end{aligned}$$

The above inequality holds, due to Lemma 4.3.2. □

Lemma 4.3.4. *When $n = 3^t$, $G_{\mathbb{Z}}(n) \geq 0.996 \cdot G_{\mathbb{Z}}(S_n)$. Let S_n be the jumping sequence $(1, 3, 3^2, \ldots, 3^t)$.*

Proof. Recall that $G_{\mathbb{Z}}(n)$ denotes the minimal cost for the case of when we restrict jumps to the integers, so we have $G_{\mathbb{Z}}(n) \geq G(n) \geq k\sqrt[k]{n}$. For any $n > 1$ the

function $f(x) = xn^{1/x}$ is concave up for $x > 0$ and so has a unique minimum which occurs at $f(\ln n) = e \ln n$. So, when $n = 3^t$, we have $G_{\mathbb{Z}}(n) \geq et \ln 3$. On the other hand, we know that $G_{\mathbb{Z}}(S_n) = 3t$. Hence, we have

$$\frac{G_{\mathbb{Z}}(n)}{G_{\mathbb{Z}}(S_n)} = \frac{et \ln 3}{3t} \geq 0.996.$$

□

The following Theorem will give a lower bound of the cost of the (q, n)-optimal tree as $q \to 1$.

Theorem 4.3.5. *For $3^t \leq n < 3^{t+1}$, we have $OPT(q, n) \geq 0.996 \cdot n \cdot G(S_{3^t})$ when $q \to 1$.*

Proof. This is a direct consequence of Lemma 4.3.1, Lemma 4.3.4. □

Now we are ready to prove the approximation ratio of the LR algorithm.

Theorem 4.3.6. *When $q \to 0$, we have* $\mathrm{LR}(q, n) < 3.015(1 + \frac{1}{\lfloor \log_3 n \rfloor})\mathrm{OPT}(q, n)$.

Proof. Suppose $3^t \leq n < 3^{t+1}$. We claim the following

$$\begin{aligned} LR(q, n) &< LR(q, 3^{t+1}) \\ &< 3(1 + \frac{1}{t})LR(q, 3^t) \\ &= 3(1 + \frac{1}{t})3^t \cdot G_{\mathbb{Z}}(S_{3^t}) \\ &\leq 3.015(1 + \frac{1}{t})\mathrm{OPT}(q, n). \end{aligned}$$

The first inequality is implied by Lemma 4.3.1 and the second one by Lemma 4.3.3. The last inequality holds due to Theorem 4.3.5. □

In the above discussion, we use the smallest balanced ternary tree with no less than n leaves as an upper bound for $\mathrm{LR}(q, n)$. By adding a small number of leaves instead of filling the whole level, we can obtain a better approximation ratio which is shown below.

We divide the integers in the range $(3^t, 3^{t+1}]$ into three consecutive subsets of equal size $H = \frac{3^{t+1}-3^t}{3}$ as follows:

$$P_1 = (3^t, 3^t + H], P_2 = (3^t + H, 3^t + 2H], P_3 = (3^t + 2H, 3^{t+1}].$$

For any $n \in P_i$, we can use $\mathrm{LR}(q, n')$ where $n' = \max P_i$ to upper bound the value of $\mathrm{LR}(q, n)$ by Lemma 4.3.1. Let $\Delta_t = \mathrm{LR}(q, 3^t) - \mathrm{LR}(q, 3^{t-1})$ and define $a = 1 - q^{3^{t+1}}$. Notice that

$$\mathrm{LR}(q, 3^{t+1}) = 3a + 3 \cdot \mathrm{LR}(q, 3^t) = 3a + 3\Delta_t + 3 \cdot \mathrm{LR}(q, 3^{t-1}).$$

It's not hard to verify the following inequalities based on the definition of the tree cost:

$$\mathrm{LR}(q, 7 \cdot 3^{t-1}) < \mathrm{LR}(q, 3^{t+1}) - \Delta_t,$$

$$\mathrm{LR}(q, 5 \cdot 3^{t-1}) < \mathrm{LR}(q, 3^{t+1}) - 2\Delta_t.$$

We now derive a lower bound for the value of Δ_t.

Lemma 4.3.7. *For $0 < q < 1$, we have $\Delta_t \geq \frac{1}{6} \cdot \mathrm{LR}(q, 3^{t+1})$.*

Proof. We only need to prove $\Delta_t > a + \mathrm{LR}(q, 3^{t-1})$. By the definition of Δ_t, we know that $\Delta_t = 2 \cdot \mathrm{LR}(q, 3^{t-1}) + 3(1 - q^{3^t})$. Then by using Lemma 4.3.2, we

have $3(1-q^{3^t}) \geq (1-q^{3^{t+1}})$, which implies $\mathrm{LR}(q, 3^{t-1}) + 3(1-q^{3^t}) \geq (1-q^{3^{t+1}})$. Therefore, we have $\Delta = 2 \cdot \mathrm{LR}(q, 3^{t-1}) + 3(1-q^{3^t}) > \mathrm{LR}(q, 3^{t-1}) + a$. □

By making use of Lemma 4.3.7, we can obtain the following theorem on the performance of LR.

Theorem 4.3.8. *When $q \to 1$, we have* $\mathrm{LR}(q,n) < 2.01(1 + \frac{1}{\lfloor \log_3 n \rfloor})\mathrm{OPT}(q,n)$.

Proof. We prove the theorem using Lemma 4.3.7 and similar arguments used in Theorem 4.3.6. The discussion below is divided into three cases according to the value of n.

Case A) $3^t < n \leq 5 \cdot 3^{t-1}$.

$$\begin{aligned}
\mathrm{LR}(q,n) &< \mathrm{LR}(q, 5 \cdot 3^{t-1}) \\
&< \frac{2}{3}\mathrm{LR}(q, 3^{t+1}) \\
&< \frac{2}{3} \cdot 3.015(1+\frac{1}{t}) \cdot \frac{3^t}{n} \cdot \mathrm{OPT}(q,n) \\
&\leq 2.01(1+\frac{1}{\lfloor \log_3 n \rfloor}) \cdot \mathrm{OPT}(q,n).
\end{aligned}$$

Case B) $5 \cdot 3^{t-1} < n \leq 7 \cdot 3^{t-1}$.

$$\begin{aligned}
\mathrm{LR}(q,n) &< \mathrm{LR}(q, 7 \cdot 3^{t-1}) \\
&< \frac{5}{6}\mathrm{LR}(q, 3^{t+1}) \\
&< \frac{5}{6} \cdot 3.015(1+\frac{1}{t}) \cdot \frac{3^t}{n} \cdot \mathrm{OPT}(q,n) \\
&< \frac{5}{6} \cdot 3.015(1+\frac{1}{t}) \cdot \frac{3}{5} \cdot \mathrm{OPT}(q,n) \\
&< 2.01(1+\frac{1}{\lfloor \log_3 n \rfloor}) \cdot \mathrm{OPT}(q,n).
\end{aligned}$$

Case C) $7 \cdot 3^{t-1} < n \leq 3^{t+1}$.

$$\begin{aligned} \mathrm{LR}(q,n) &< \mathrm{LR}(q, 3^{t+1}) \\ &< 3.015(1+\frac{1}{t}) \cdot \frac{3^t}{n} \cdot \mathrm{OPT}(q,n) \\ &< 3.015(1+\frac{1}{t}) \cdot \frac{3}{7} \cdot \mathrm{OPT}(q,n) \\ &< 2.01(1+\frac{1}{\lfloor \log_3 n \rfloor}) \cdot \mathrm{OPT}(q,n). \end{aligned}$$

□

As $q \to 1$, both the LR algorithm and GLR algorithm maintain almost balanced trees, but they deal with the "leftover" leaves in two different ways: the LR puts all the "leftover" leaves at the bottom level of the complete ternary tree; the GLR bundles all the "leftover" leaves as a "leftover" tree and puts it as a sibling of the complete balanced trees. In Figure 4.4, We compare the approximation ratios of LR and GLR when $q = 0.99$ and $n = 1 \cdots 1000$. From the simulation results, we see that when n is large, GLR appears to perform better.

4.4 Acknowledgements

The text of Chapter Four, in part, is adapted from the paper, "Approximately Optimal Trees for Group Key Management with Batch Updates", which is going to appear in Special Issue of Theoretical Computer Science. I would like to thank my co-authors, Minming Li, Ze Feng, Ronald L. Graham and F. F. Yao for allowing me to include our joint work in this thesis.

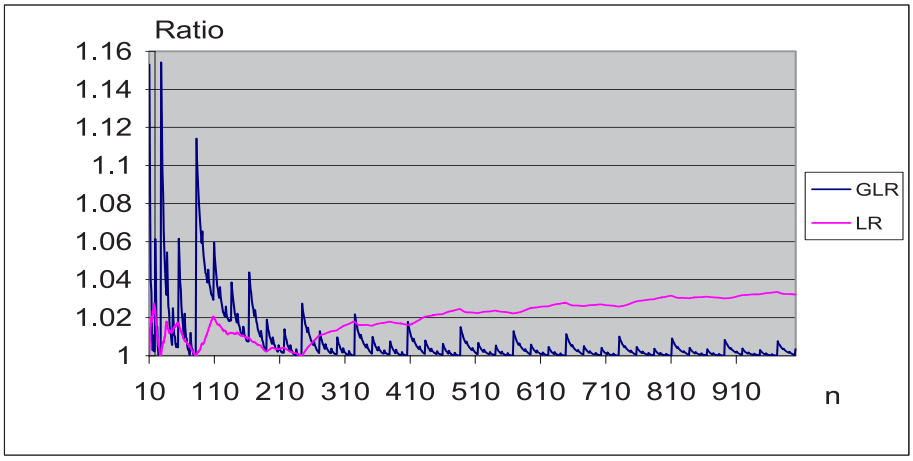


Figure 4.4: The Ratio of LR and GLR as $q = 0.99$.

Chapter 5

Conclusions and Future Work

This dissertation addressed the group key management problem for broadcasting applications. We analyzed in detail on one of the popular network models, in which group members have the same probability p of changing membership in a batch period. We focused on the problem of constructing "good" trees and analyzing the properties of GKM optimal trees, given the number of group members n and the probability p of changing membership. In the first part of the dissertation we focused on finding the properties of GKM optimal trees, proposing a lower bound for the cost of optimal GKM trees. In the second part, we looked at the limiting case when $n \to \infty$. As $n \to \infty$, we showed that almost all the subtrees of the root are the same (Majority Property), and proved such subtree structure can be determined by the value of q. Then, we proposed an efficient approximation algorithm to construct an approximate GKM tree by using such subtree structure.

Computer simulations showed the algorithm is very effective. In the last part, we saw another limiting case when $q \to 1$, i.e. $p \to 0$. A new cost function of the GKM tree was defined for this special case. As $q \to 1$, we designed an $O(\log n)$ heuristic algorithm and showed it produces a nearly 2-approximation to the optimal GKM trees. In this dissertation, we also defined a new mathematical problem, called the Jumping Sequence Problem, which is a real number analog of the GKM problem. We analyze the jumping sequence behavior from 1 to n, and show some important properties of the jumping sequences. All these properties agree well with the known facts about optimal GKM trees.

There are many interesting problems and directions of future research arising from the work presented here.

1. In this dissertation, we gave an approximation algorithm, called GLR, to build approximate GKM trees. The GLR algorithm can be used on the Group Key Management model in which the number of group members changes. We saw this algorithm was quite good on GKM trees by practical experiments. However, in this dissertation, we didn't provide the numerical bound of the approximation ratio for either the case when n is stable or the case when n is changing.

2. We proposed a new mathematical problem, called the Jumping Sequence Problem. In this dissertation, we have focused on two different cost functions involved with jumping from a to b, i.e., $(1 - q^b)/a$ and b/a, which are both related to the GKM tree problems. In Chapter Three, we have proved if the cost

of jumping from a to b is b/a, the optimal jumping sequence which starts with $(1, 3, 8, 22, 60, 163, 443, 1204, \ldots)$, matches with the first hundreds of terms of the sequence `A024581` in the `OEIS` [24]. We have also noticed if the cost of jumping from a to b is set to be $(a + b)/a^2$, the jumping sequence which starts with $(1, 2, 5, 12, 29, 70, 169, 408, 985, \ldots)$ seems to be the optimal sequence to jump from 1 to ∞ and appears to agree with the Pell numbers. There are many different possible cost functions and it would be very interesting to see what combinatorial properties different cost functions have.

3. In this dissertation, we considered the model in which each group member has the same probability p of changing membership. A much harder problem is to find an efficient tree to arrange the group members, where each member has a different probability p_i of changing membership. Under this new model, the cost functions of trees become very complicated, and little work has been done in the literature. A good way to attack this problem may be to consider a simplified version first: there are two different probabilities p_1, p_2 of changing and each member has either probability p_1 or p_2 of changing.

There clearly remain many interesting and important research directions to be pursued.

Bibliography

[1] H. Harney and C. Muckenhirn. Group key management protocol (GKMP) architecture. *RFC 2094*, 1997.

[2] H. Harney and C. Muckenhirn. Group key management protocol (GKMP) specification. *RFC 2093*, 1997.

[3] T. Ballardie and J. Crowcroft. Multicast-specific security threats and countermeasures. *Network and Distributed System Security, 1995., Proceedings of the Symposium on*, pages 2–16, 1995.

[4] Klaus Becker and Uta Wille. Communication complexity of group key distribution. In *CCS '98: Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 1–6, 1998.

[5] Colin Boyd. On key agreement and conference key agreement. In *ACISP '97: Proceedings of the Second Australasian Conference on Information Security and Privacy*, pages 294–302, 1997.

[6] Bob Briscoe. MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences. In *NGC '99: Proceedings of the First International COST264 Workshop on Networked Group Communication*, pages 301–320, 1999.

[7] Refik Molva and Alain Pannetrat. Scalable multicast security in dynamic groups. In *CCS '99: Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 101–112, 1999.

[8] Adrian Perrig, Dawn Song, and J. D. Tygar. ELK, a new protocol for efficient large-group key distribution. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 247–262, 2001.

[9] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman key distribution extended to group communication. In *CCS '96: Proceedings of the*

3rd ACM Conference on Computer and Communications Security, pages 31–37, 1996.

[10] Chun-Ying Huang, Yun-Peng Chiu, Kuan-Ta Chen, and Chin-Laung Lei. Secure multicast in dynamic environments. *Computer Netwworks*, 51(10):2805–2817, 2007.

[11] Gianluca Dini and Ida Maria Savino. Scalable and secure group rekeying in wireless sensor networks. In *PDCN'06: Proceedings of the 24th IASTED International Conference on Parallel and Distributed Computing and Networks*, pages 84–88, 2006.

[12] H. Ragab Hassen, A. Bouabdallah, H. Bettahar, and Y. Challal. Key management for content access control in a hierarchy. *Computer Netwworks*, 51(11):3197–3219, 2007.

[13] Xinliang Zheng, Chin-Tser Huang, and Manton Matthews. Chinese remainder theorem based group key management. In *ACM-SE 45: Proceedings of the 45th Annual Southeast Regional Conference*, pages 266–271, 2007.

[14] Yacine Challal, Hatem Bettahar, and Abdelmadjid Bouabdallah. SAKM: a scalable and adaptive key management approach for multicast communications. *SIGCOMM Comput. Commun. Rev.*, 34(2):55–70, 2004.

[15] Olivier Chevassut Emmanuel Bresson and David Pointcheval. Provably secure authenticated group Diffie-Hellman key exchange. *ACM Trans. Inf. Syst. Secur.*, 10(3):10, 2007.

[16] Dijiang Huang and Deep Medhi. A key-chain-based keying scheme for many-to-many secure group communication. *ACM Trans. Inf. Syst. Secur.*, 7(4):523–552, 2004.

[17] S. Zhu, S. Setia, and S. Jajodia. Performance optimizations for group key management schemes. In *ICDCS'03: Proccedings of the 23rd International Conference on Distributed Computing Systems*, pages 163–171, 2003.

[18] Xiaozhou Steve Li, Yang Richard Yang, Mohamed G. Gouda, and Simon S. Lam. Batch rekeying for secure group communications. In *Proceedings of the 10th International World Wide Web Conference on World Wide Web*, pages 525–534, 2001.

[19] Suvo Mittra. Iolus: A framework for scalable secure multicasting. In *SIGCOMM*, pages 277–288, 1997.

[20] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1):16–30, 2000.

[21] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Trans. Netw.*, 7(4):502–513, 1999.

[22] F. Zhu, A. Chan, and G. Noubir. Optimal tree structure for key management of simultaneous join/leave in secure multicast. In *Military Communications Conference*, volume 2, pages 773–778, 2003.

[23] Ronald L. Graham, Minming Li, and Frances F. Yao. Optimal tree structures for group key management with batch updates. *SIAM J. Discret. Math.*, 21(2):532–547, 2007.

[24] N. J. A. Sloane. On-line enyclopedia of integer sequences, http://www.research.att.com/ njas/sequences.