Representing Shortest Paths in Graphs Using Bloom Filters without False Positives and Applications to Routing in Computer Networks

Gökçe Çaylak Kayaturan

A Thesis Submitted for the Degree of
Doctor of Philosophy

Department of Mathematical Sciences

University of Essex

October 2017

SUMMARY

A Bloom filter is data structure for representing sets in a compressed form, which has many applications. Bloom filters save time and space, but produce errors known as false positives.

Recently it has been suggested that Bloom filters can be used for encoding paths in graphs, for the purpose of delivering messages in computer networks. False positives mean that a message will be delivered not only to its destination, but also to some nodes to which it should not have been delivered. Some ways of reducing the probability of such errors have been discussed in the literature.

In this thesis, a new approach is suggested. Instead of choosing labels for edges at random (as is done in the standard Bloom filter approach), labels for edges are chosen based on the graph and the position of an edge in the graph. It is shown that under some assumptions (the graph is known, and only shortest paths are encoded), there will be no false positives leading to a message being delivered to a wrong node.

Chapter 1 introduces the routing scenario, the Standard Bloom Filters and the encoding methods that are investigated in the following chapters. Different types of assumptions concerning the graphs are applied in further chapters. Shortest paths in rectangular grids are encoded in Chapter 2; then king's graphs, hexagonal grids and triangular grids are studied in Chapters $3, 4$ and 5, respectively. Another realistic networks is considered in Chapter 6. Finally, a way to generalise these encoding methods to arbitrary graphs is discussed in Chapter 7, which concludes the thesis.

DECLARATION

The work in this thesis is based on research carried out at the Department of Mathematical Sciences, University of Essex, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification, and it is all my own work, unless referenced, to the contrary, in the text.

ACKNOWLEDGEMENTS

Firstly, I would like to thank Turkish Ministry of Education who financially support me during my studies in the UK. This scholarship opens many doors to me, so in particular, I am grateful to who made this law which provides a scholarship for students to get a degree in foreign countries for giving me a chance to get a high quality education and invaluable experience.

I would like to express my sincere gratitude to my supervisor Dr. Alexei Vernitski for the continuous support of my PhD study, for his patience, immense knowledge and encouragement. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study.

Besides, I would like to thank my examiners: Prof. Tomasz Radzik and Prof. Peter Higgins for their insightful comments, but also for the hard questions which encourage me to widen my research from various perspectives.

My sincere thanks also goes to Prof. Abdellah Salhi for his guidance, advices and sharing his experiences during my study, and to the Department of Mathematics and administrative staff without their precious support it would not be possible to conduct this research.

Also, I would like to thank my family: my mother, Melek, and my father, Hasan, for their care and support over the years and believing in me; and to my siblings Eleni and Mehmet(Hayran) and friends Dilek, Mehmet (Unver), Tuba for supporting me spiritually throughout writing this thesis and my life in general; and to Irem for her friendship, support and all the fun we have had in the last four years.

Last but not the least, my special thanks goes to my husband Metin who gives meaning to my life for his patience through many changing years. I owe my success to him.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

A Bloom filter, named after its inventor Burton Bloom ([7]), is a way to compress the data. Accordingly, it is useful to prefer the Bloom filter for variety of applications on computer networks.

In other words, a Bloom filter is a way to represent a set. Consider a universal set U of all elements and a subset S of U. Suppose each element in the set U is labeled by a binary array of length m where a certain number of bits are set to 1. The subset S has n elements. A Bloom filter represents the set S using a bit-vector of length m. Each element is stored and it can be determined that an element x is whether in the set S or not by comparing the representative binary arrays of the set and the element in all bits. If the bits 1 in the binary array of the element x matches with the bits 1 in the Bloom filter of the set S, we may say $x \in S$. Yet, there might be some elements, which are not really in the set S, in the set U, their binary arrays may be less than

or equal to the Bloom filter of the set S in all bits positions. These elements which are not really in the set S might be thought of as elements in the set S. This kind of elements are called false positives of the Bloom filter of the set S.

This way of presenting sets is very space and time efficient, since representing the sets with Bloom filters saves space compared to other methods which are naive representations of sets and it is easy to query the elements whether they are in the set or not (see the Sections 1.3.1 and 1.3.2 for details). These are the opportunities of the Bloom filters.

However, a shortcoming needed to be dealt with is the false positives that are produced by the Bloom filter. The main purpose of the researches conducted on the Bloom filter is to minimize the number of false positives when saving time and space. A wide range of research for network applications of Bloom filter has been surveyed by Broder and Mitzenmacher in [10]. Indeed, some studies on the networks contribute the new applications of Bloom filter. In the paper [10], it is said that the applications of Bloom filters should be categorized, since, while some applications use the Bloom filter on the problems that are especially data mining, reducing data traffic, security of data transmission between parties, others might use it to sort combination of these problems out.

The standard Bloom filter is a random data structure and represents sets using small memory. It may use relatively short binary vectors so it may be impossible to have a unique representation for each set. Therefore reducing the randomness might be thought of as a way to decrease the number of false positives. Another way to construct a Bloom filter for less randomness has been described in [34].

The Bloom filter is a way to represent a set with the bits 0s and 1s. Adding an element to the set or deleting an element from the set is another challenge for the applications of the Bloom filter. While adding elements to the set increases the size of the space, deleting an element from the set in reasonable time is not easy for users. To deal with these problems, some of the network applications of Bloom filters [8] [21] [22] [42] [63] have been proposed in literature.

One might think to delete some elements from the set in order to reduce the false positives. Yet, this is not as easy as to swap the corresponding bits from 1 to 0 in the Bloom filter of the set. Since, the bit 1 is placed in Bloom filters of the elements randomly, that is why the bit 1 in the Bloom filter of the set might come from not only the Bloom filter of deleted element, but also multiple number of the Bloom filters of other elements in the set. In this case, some elements, which exactly belong to the set, might be thought of as nonexistent in the set. These kind of elements are called false negatives (we have given a concrete example of false negatives in the Section 1.3.2).

By imposing some additional restrictions on the structure of the Bloom filter to avoid the false negatives from the set, a generalisation of Bloom filter called in-packet Bloom filter [57] basically has been designed as the standard Bloom filter with requirement of same parameters. The method proposed by [57] computes the ratio of deletable elements from the system securely as well as yields optimum ratio of false positives in order to increase the performance of routing in networks.

Another generalisation of Bloom filter called compressed Bloom filter [46] focuses on decreasing the size of the network messages transmitted in order to save space.

The idea behind compressed Bloom filters is that reformulate the size of Bloom filter via fixing the optimum probability of false positives of standard Bloom filter.

The Bloom filter may be used for the transaction of the messages between computers in a network without disruption. This is related with another generalisation called cryptographic Bloom filters [6], and also another generalisation of Bloom filter has been introduced by [47] for transmission of the big data.

1.2 Bloom Filters in Graphs

As indicated in previous section, the main purpose of the applications of the Bloom filters is to reduce the number of false positives while saving space and time. In this thesis, we aim to avoid false positives by choosing positions for individual elements in graphs not at random, but using some rules. Since, the problem we consider in this thesis is the false positives which may cause extra network traffic for some routing scenarios (see the Sections 1.5 and 1.6).

In the literature, a network may have various forms (see [44] for a survey). The paths have an important role to provide an efficient communication between computers in these networks. One of the shape for a network is a grid which is a family of a graph that can be built via cartesian product. Specifying the paths, where the information is shared between parties, in grids is another problem and another domain in graph theory [28]. One solution, that is producing algorithms, have been examined by [65] for any path-finding in variety number of graphs. This is another research field that we do not consider in this thesis. In the following chapters we have found

the shortest paths in graphs by using our approach.

For less computational requirements by using Bloom filters, another algorithm, that increases the security of network traffic flow, on paths has been established by [64].

In order to reduce the number of false positives in networks, the idea of encoding the links between nodes in a way similar with our work has been studied in another paper [12]. Even though the coding requires slightly more processing than the standard Bloom filter, the most current method called yes-no Bloom filter [12] yields efficient results of false positives.

In a network model proposed in [40], the Bloom filter has a role as a storage of large documents which might be shared by the servers. Also, the Bloom filter is used in an algorithm, which aims to find the location of the documents whose size is reduced by using the basic principles behind the Bloom filter, [55]. Yet, the methods proposed in both [40] and [55] require to update the Bloom filters when a new information is added to the stored documents. Yet, this property of applications of Bloom filter requires more time for queries the documents.

According to the routing model in [41] the main point with the distribution of the messages is the number of nodes in the network, therefore the problem with delivering a message between nodes is wasting time and resources. When the number of nodes increases in the network, the message is delivered to the receiver, but the resources are wasted.

Throughout the thesis we label the links (edges) between nodes (vertices) and send the code through a route chosen in advance, similar model were considered in [13].

In order to optimize the length of the Bloom filter, the encoding method proposed in [13] offers to control the probability of false positives with respect to the size of the space. Since, sometimes it may not be easy to remove the false positives from the networks. In that case, the method introduced in [13] allows the users to choose the best possible rate of the false positives.

1.3 Arbitrary Encoding Methods in Networks

One model of message delivery in a computer network is based on labelling each edge by a subset of a universal set, and then encoding a path as the union of the labels of its edges. There might be several methods to label the edges. We will show two widely used methods in this section. One can summarise these two methods as a construction of an appropriate Boolean array, which is a sequence of bits 0 and 1, for each link in the network.

1.3.1 Bit-per-edge Labelling

Consider a subset S of a universal set U of all elements. The bit-per-edge labelling method is essentially the same as representing subsets as characteristic vectors. This method is the way to encode each element in the set U with a Boolean array which is a sequence of bits 0 and 1. Each element is represented by a bit 1 in a specified position in this Boolean array, hence these bits take certain places in the array. The length of the Boolean array is obviously the number of the elements in the universal set U. It is easy to query whether an element is in the set S or not with this labelling

method.

For instance; consider an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The edges of this graph are the elements of a universal set U. Each edge has a designated position in bit arrays of size m where m is the number of edges in the graph. In this case, we call this encoding method *bit-per-edge labelling*. The length of the Boolean array is the number of edges in the graph. A path in the graph is viewed as a subset S of $U = E$. The representative Boolean array of this path contains the bits 1 in the bit position of corresponding edges which are in the path (see the Section 6.3.2 for a detailed bit-per-edge labelling in star graphs). To query whether an edge is on the path or not, the representative binary arrays of the set S and an edge e are compared in all bits positions. We definitely say that $e \notin S$, if the bit 0 is in the bit position which is reserved for the edge e in the representative binary array of the set S.

It may be obvious to answer whether an element is in the set S when the elements are encoded by using bit-per-edge labelling, we prove that if any element is a false positive of the representative binary array of the set S by the following theorem. Note that a false positive is an element in the set U and this element, which is not an element of the set S, might be counted as an element of the set S in some cases.

Theorem 1.3.1. *The labelling method we refer as the bit-per-edge does not produce a false positive.*

Proof. Consider a universal set U with n elements and a subset S of U with k elements. The elements in the set U are represented by one bit in the Boolean array which is

a sequence of bits 0 and 1. The elements in set S are denoted by the bit 1 in the Boolean array of the set S. Obviously, the Boolean array of the set S contains k number of bit 1 and other bits are 0 whose number is $n - k$.

Suppose an element f in the set U is a false positive of the Boolean array of the set S. The Boolean array of the element f contains the bit 1 in its ith bit position among n bits. The bits 1s in the Boolean array of the set S are placed in k distinct bit positions corresponding to the elements of the set S. If the Boolean array of the set S contains the bit 1 in its ith bit position, then we can conclude that the element f is an element of the set S. Otherwise, there must be $k + 1$ number of bit 1 in the Boolean array of the set S. This contradicts the number of elements in the set S. Therefore, the element f is not a false positives of the Boolean array of the set S.

□

The bit-per-edge labelling method of representing sets is very time-efficient, since it is easy to query the elements whether they are in the subset S or not; but not very space-efficient, since the universal set U might contain large number of elements. Hence, as a more advanced approach we consider is the Bloom filters.

1.3.2 Standard Bloom Filter

Other encoding method called standard Bloom filter has been invented by Burton Bloom [7].

Suppose a subset S of a universal set U has n elements. The set S is represented by a binary array of m bits. In order to make this possible, each element which

potentially can be an element of S is represented by an m bit array in which k bits are set to 1. These bits 1 in the array are distributed to random bit positions without replacement. All other bits are set to zero.

We assume that the Bloom filter of the set S is constructed by applying the OR operation ([62]) in bitwise to all Bloom filters of the elements of S. Note that binary OR operation takes the bits 0 and 1 as inputs and produces a bit 1, if at least one of the input is 1. Otherwise, it produces a bit 0, when all inputs are 0. Also, it is possible to consider other binary operations in order to build the Bloom filter for the set S. These operations are bitwise AND operation [45], bitwise XOR operation [59] and bitwise NOT operation [43]. Bitwise AND operation produces the bit 1 when all inputs are 1, otherwise it generates the bit 0. Besides, the bitwise XOR operation yields the bit 0 when all inputs are 0 or 1, otherwise it produces the bit 1. Finally, the bitwise NOT operation takes the bit 0 and outputs the bit 1, and when it takes the bit 1, then it outputs the bit 0. The Bloom filter of the set S can be obtained by applying any of these operations in bitwise to the all elements in the set S.

After the Bloom filter of a set S has been constructed, one can query whether an element x in U belongs to the set S or not by comparing the Bloom filter of the set S with the Bloom filter of the element x in all bits positions. If at least one of these positions in the Bloom filter of x is greater than the corresponding bit positions of the Bloom filter of S, then we say that the tested element is definitely not in the set S. However, when the Bloom filter of the tested element is less than or equal to the Bloom filter of the set in all bits positions, then this element probably be in the set S. The comparison of the bits of Bloom filters of both the element and the set may

conclude that some elements might be incorrectly counted as an element of the set. These kind of errors are called *false positives* which are seemingly in the set S, but they might not be really in the set S.

A set might be defined as a collection of elements which are well defined and related with each other. Any objects might construct a set, as a concrete example, an attendee list of a conference represents a set such as $S = \{Claire, David, Rod, Tori\}$. Each attendee can be represented by a fixed length Bloom filter with randomly allocated bit 1 such that $[1, 0, 1, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 1, 0, 0], [0, 0, 1, 0, 0, 1, 0, 0]$, $[1, 0, 0, 0, 1, 0, 0, 0]$, respectively. By applying OR operation in bitwise to all Bloom filters of the elements in the set S, the Bloom filter of the set denoted by $\beta(S)$ is obtained as $[1, 0, 1, 0, 1, 1, 0, 0]$. If one called Albert represented by a Bloom filter $[1, 1, 0, 0, 0, 0, 0, 0]$ claims that he is an attendee of the conference, in second bit position, the Bloom filter of the set is less than the Bloom filter of Albert. In conclusion, Albert definitely is not an attendee of the conference. On the other hand, another person Kate represented by a Bloom filter $[0, 0, 1, 0, 1, 0, 0, 0]$ might claim another attendee of the conference. In all bits positions the Bloom filter of Kate is less than or equal to the Bloom filter of the set $\beta(S)$. Hence, one might think Kate is a possible attendee. Yet, she is a false positive.

On the other hand, a person Mary with a Bloom filter $[0, 1, 0, 0, 1, 0, 0, 0]$ may want to register for the conference, then it is easy to update the Bloom filter of the set. The Bloom filter of the set including the new attendee Mary can be obtained by applying a bitwise OR operation to the Bloom filter of the set and the Bloom filter of Mary in all bits positions. Hence, the updated Bloom filter of the set is obtained

as $[1, 1, 1, 0, 1, 1, 0, 0]$.

Yet, if someone (say Rod) wants to leave the conference, then one might think to swap the bits 1 with the bit 0 in the corresponding bit positions of Bloom filter of the set in order to update the Bloom filter of the set. By replacing the bits 1 with the bit 0 in the third and sixth bit positions, which correspond to the bit 1 in the Bloom filter of Rod, of the Bloom filter of the set, then the Bloom filter of the set S is found as $[1, 0, 0, 0, 1, 0, 0, 0]$ after leaving *Rod*. However, in this case, the attendees Claire and David look like the false negatives of the set S. Since, they are already members of the set, yet the Bloom filters of them are not less than or equal to the Bloom filter of the set in all bits positions. In brief, it is not easy to update the Bloom filter of the set, when some elements are deleted from the set. This is because the bit 1 in some bit positions of the Bloom filter of the set comes from the Bloom filter of multiple elements in the set. Note that, we examine this example to clarify the meaning of false negatives with a concrete example. Some elements called false negatives are outside the scope of this thesis. Note that false negatives are the elements in the set S yet in some cases they might be thought of not being in the set S.

While the Bloom filter provides fast answers the membership queries of a set, it potentially yields large number of false positives. Also, it is not easy to eliminate the false positives from the set. In the very small example above, the Bloom filters of elements consist of fixed number of bits with fixed number of bit 1. The worst case of our example is that the Bloom filter of the set contains many 1s and the length of Bloom filter is too short. Depending on the size of the domain the Bloom filter of the set in the example above may produce many false positives.

1.3.2.1 Rate of False Positives of Standard Bloom Filter

A Bloom filter of an element is a binary array with m bits and k bit positions are set to 1 in this Bloom filter. These k random distinct bit positions are placed in any bit positions in the array without replacement. The bits 1s in the Bloom filter of some elements may coincidently match with the bits 1s in the Bloom filter of the set S where these elements are not really in the set S. The Bloom filters represent sets using small space, so it may be impossible to have a unique representation of each set. Thus, this membership query makes the false positives possible. The probability of false positive is the probability of the positive answer to this membership query. The probability of false positive is a function which depends on the number of 1s in the Bloom filters of elements in the set S, length of the Bloom filters of these elements and number of the elements in the set S.

The further study of Bloom filter [7] shows that the probability of false positive p can be computed as in the (1.1) which is obtained as a result of simple probability where m is the number of bits in the Bloom filter, n is the number of elements in the set and k is the number of bit 1 in the Bloom filter of an element.

$$p = \left(1 - (1 - \frac{1}{m})^{kn}\right)^k \tag{1.1}$$

It is assumed that the bit 1 is placed on any bit position in a Bloom filter of an element with an equal probability. When the number of bits in the Bloom filter

is m, the probability that a given bit position is chosen when a random position is selected is $\frac{1}{m}$. Accordingly, a certain bit is not 1 in m length binary string with the probability $1 - \frac{1}{m}$. The number of bit 1 in the Bloom filter of an element is exactly k, and there are n elements in the set S. Note that, the bits 1s are placed in the Bloom filter of an element without replacement. The probability that a specific bit is not set to 1 in the Bloom filter of the set S is $(1 - \frac{1}{m})^{kn}$. Yet, we are looking for the probability of certain places containing the bit 1 in the Bloom filter of the set. Hence, a given bit position in the Bloom filter of the set is set to 1 with probability $1-(1-\frac{1}{m})^{kn}$ and finally there are k random 1 placed in the Bloom filter of an element without replacement. We might check any element whether being in the set or not by comparing the Bloom filter of the set and the Bloom filter of the element in all bit positions with the probability of false positive obtained as:

$$p = \left(1 - (1 - \frac{1}{m})^{kn}\right)^k \tag{1.2}$$

which is approximately:

$$\left(1 - e^{\frac{-kn}{m}}\right)^k \tag{1.3}$$

By taking the derivative of (1.3) with respect to the parameter k and equalizing it to 0, the optimum number of the bit 1 denoted by k has been obtained as $\ln 2 \times \frac{m}{n}$, [10] [22] [46]. Yet the parameter k must be an integer so $k = \lceil \ln 2 \times \frac{m}{n} \rceil$ which is the

smallest integer that is not less than $\ln 2 \times \frac{m}{n}$, while the parameters m and n are given. Hence, the optimum number of 1 in the Bloom filter minimizes the probability of false positives. Obviously, when the value of $\frac{m}{n}$ increases, then the optimum number of k increases, yet the probability of false positives decreases, [22]. Note that, the number of the bit 1 in the Bloom filter of an element must be an integer.

Although, the formula indicated with (1.3) which is approximation of the formula (1.1) has been accepted as the probability of false positives by the vast majority of applications of Bloom filter. In the literature there has been some new approaches to the probability of false positives of standard Bloom filter, e.g.[48]. It has been considered in [9] that the probability of false positives is greater than the formula (1.1) and this claim has been shown in [16]. The formula (1.1) is for the experiment as indicated above, the formula (1.4) which has been obtained by using the proof of balls and bins model on three parameters k, m, n, [16] expresses the expected rate of false positives for a random n-element set S. Hence, the probability of false positives p_{false} has been computed as the following formula.

$$p_{false} = \frac{m!}{m^{k(n+1)}} \sum_{i=1}^{m} \sum_{j=1}^{i} (-1)^{i-j} \frac{j^{kn} i^k}{(m-i)!j!(i-j)!} \tag{1.4}$$

This formula might yield more realistic results for small size of parameters, where for instance; m is 32 bits or 64 bits, [16]. Yet it is difficult to compute the formula (1.4) for large values of the parameters k, m, n. Nevertheless, the sizes of both m and k are not expected to be too large, some applications of Bloom filter may require large amount of space. The parameters m and k should be chosen large enough since the

aim is to keep both membership query time and space as small as possible. Namely, the number of false positives might be small enough to be ignored.

1.3.2.2 An Approach to the Probability of False Positives

The probability of false positives of Bloom filters has been studied in engineering papers. In this section, we analyse the probability of the false positives with other approach than above. Note that, in the following chapters of this thesis we do not consider the results that we have obtained in this section. We still work on this approach as a new direction of the false positives.

We are interested in the formula (1.3). The question of our interest is how much we might ensure that any chosen element in the set U is not a false positive of the Bloom filter of the set S with the probability of at least 95%?

We restrict the probability of no false positives to 95% or more. This percentage is a confidence interval of our model. A confidence interval is a percentage that makes us to be sure for the true value of the false positives in a set. We could be more certain if the percentage increases. Our consideration is to give a new approach to the probability problem of false positives. In this section we aim to show that the value of the parameters m, n, and k of the Bloom filters can be controlled when the value of the probability of the false positives is assumed to be in a confidence interval.

Additionally, it is assumed that all elements in U are tested whether they are an element of the set S or not with an equal probability. The elements are independent which means that there is no influence on each other. Hence, they are called independent and identically distributed. Practically, the probability of false positive of

the Bloom filter of each element is different from each other, even if the values of the parameters m, n, and k are fixed. Since, the bits 1 in the Bloom filter of each element and the Bloom filter of the set S matches with different number during the membership query of an element. Nevertheless, there are not big differences between the probabilities of false positives of elements.

We recall the classic formula to show that it is possible to specify the parameters m, n and k when the rate of false positives is assumed to be in a confidence interval.

$$\left(1 - (1 - \frac{1}{m})^{kn}\right)^k \approx \left(1 - e^{\frac{-kn}{m}}\right)^k \tag{1.5}$$

Optimum values for the parameters can be determined experimentally for some applications of Bloom filters [12], on the other hand we need to make it more general instead of trying all possible values on the formula (1.3) and the formula (1.4). For this purpose a parameter a is defined as the area left outside confidence level, [37]. More specifically a represents an upper bound on the probability of false positives. For instance: if the confidence level is 95%, then the probability of false positives is 0.05 on the outside of the confidence zone. Following this example it is generalised that the confidence level is determined with the formula $100(1-a)\%$ when a is given, [37].

We assume the probability of false positives of the standard Bloom filter is obtained with at most 5% and we wish to increase the certainty. If the probability of success of any element being not a false positive is at most 95%, then the formula of probability of false positive of that element should be taken less than or equal to 0.05.

For more certainty it is possible to take the value of the probability of false positives less than 0.05, for instance 0.01 for the confidence interval 99%. All parameters k, m and n are integers and the functions exp and ln are increasing functions. Therefore, we are able to restrict the number of elements in a set as a result of the following analysis.

$$\left(1 - e^{\frac{-kn}{m}}\right)^k \leq a \tag{1.6}$$

$$1 - e^{\frac{-kn}{m}} \leq (a)^{\frac{1}{k}} \tag{1.7}$$

$$e^{\frac{-kn}{m}} \geq 1 - (a)^{\frac{1}{k}} \tag{1.8}$$

$$-\frac{kn}{m} \geq \ln\left(1 - (a)^{\frac{1}{k}}\right) \tag{1.9}$$

$$n \leq -\frac{m}{k}\ln\left(1 - (a)^{\frac{1}{k}}\right) \tag{1.10}$$

As an example; let the parameters k and m be 6 and 256, respectively. If these sizes are put in the inequality (1.10), the number of elements n is obtained as at most 42 for 95% confidence. An element might be a false positive with 5% probability for these sizes of a set. If we increase the confidence interval for more certainty to 97%, then the the probability of false positive will be 0.03. If the value 0.03 is put into the inequality (1.10), then we should reach the maximum number of elements in the set with the probability of false positives 0.03. For the same parameters $k = 6$ and $m = 256$ the number of elements n can be found as at most 35 for the 97% confidence level. Similarly, for $k = 6$ and $m = 256$ number of elements is at most 23 with 0.01

false positives rate for the 99% confidence level.

1.4 Representing the Edges in Graphs with Bloom Filters

A network might be represented by an undirected graph denoted by $G = (V, E)$ where V and E are the sets of vertices and edges, respectively. Also a universal set U models the header of a message sent from one computer in G to another. A set S, whose elements are the edges of a path in the graph, is a subset of E.

Suppose each edge $e \in E$ is labeled by a subset of U; we shall denote the Bloom filter of e by $\beta(e) \subseteq U$. Hence, the Bloom filter of a subset S of edges is defined as $\beta(S) = \bigcup_{e \in S} \beta(e)$. An edge $e \in E$ is recognised as an edge on the path represented by S, if $\beta(e) \subseteq \beta(S)$. Obviously, if $e \in S$ then e is recognised by $\beta(S)$. Yet, it is also possible that $e \notin S$ and e is recognised by $\beta(S)$; one refers to this situation as a *false positive*.

We shall say that a set of edges S is represented by its label $\beta(S)$ if $e \notin S$ which are adjacent to the shortest path are not recognised by $\beta(S)$. A particular scenario we have in mind is when S is a path connecting vertices u and v, and $\beta(S)$ is used for routing a message from u to v (or from v to u); thus, $\beta(S)$ is sent along with the message as its header. We assume that each vertex $v \in V$ is a computer which cannot access information about the general structure for the network when it is used for routing messages, but can access the labels of the edges which are incidental to

v; accordingly, v can compare the header of the message $\beta(S)$ with these labels and decide along which edge the message should be sent next. In this case, it is aimed to prevent the messages to be wasted and to guarantee the delivery. Since, a computer tends to send the message throughout all connections to it. Thus, we encode the links in the network.

If S is represented faithfully by $\beta(S)$ then at each vertex on S it is clear from inspecting $\beta(S)$ along which edge the message should be sent next. However, if S is not represented faithfully by $\beta(S)$, that is, there is a false positive $f \in E$ adjacent to the path S then it will be impossible to find out from inspecting $\beta(S)$ whether the message should be sent along f or not.

In practice, a subset $\beta(S) \subseteq U$ would be represented as a binary array of length $|U|$, in which each position corresponds to one fixed element of U; in the array representing $\beta(S)$, a bit at a certain position equal to 0 (or 1) means that this element of U does not belong to S (or belongs to S). Thus, the header attached to the message to describe where and by what route it should be delivered has size $|U|$. When a computer at a vertex v decides where to forward the message, it considers each edge e incidental to v and checks whether $\beta(e) \subseteq \beta(S)$; in practice, this comparison is implemented as a bitwise comparison of two binary arrays of length $|U|$ which represent $\beta(e)$ and $\beta(S)$; this operation can be performed very fast; in fact, it can be performed while the header is passing through v (for example, as an optical signal), without the need to store the bits of the header at v and then perform any arithmetic operations on them. Such fast performance makes this model an attractive possibility for routing in computer networks [13] [27].

In our research we assume that S is a shortest path between a vertex u and a vertex v. Our research concentrates on looking for the ways of labelling edges in a given graph so that, on the one hand, each shortest path is represented faithfully, and, on the other hand, the size $|U|$ is reasonably small.

We describe how for such a graph, a labelling can be defined which represents each shortest path faithfully; that is, our labellings do not produce false positives which are edges adjacent to the shortest paths, providing that they are used only for labelling shortest paths and not other sets of edges.

1.5 A Possible Routing Scenario

We consider a routing scenario from one computer to other computer in several networks in each chapter of this thesis. We have chosen a shape from graphs for a network in each chapter. These shapes have been used in literature as a realistic network model in modern computer science.

We assume that there are computers on each vertex of given graph. A computer might send a message to another computer in the graph. The sender computer chooses the shortest path to forward the message to the receiver computer in advance. The messages are forwarded via edges that are the links between computers.

We construct separate encoding methods for all edges in corresponding graph in each chapter. The Bloom filter of a shortest path is obtained by applying bitwise OR operation to all edges on the shortest path. The Bloom filter of the shortest path and the message are sent together by the computers, like in [13]. Once a computer

on the shortest path receives the message, it does not send the message back and forwards it along the edges whose Bloom filter is recognised by the Bloom filter of the shortest path. If the Bloom filter of the shortest path confirms that an edge is on the shortest path, then the computer on the shortest path sends the message to the next computer on the shortest path via this edge.

A computer might send the message along all edges linked with itself. Hence, in this routing model, the possible false positives are the adjacent edges to the shortest path that is chosen by the sender computer in advance.

The encoding methods we have constructed might prevent a possible interruption of messages called false positives from the path during the transaction between nodes. Hence, this might make the Bloom filter reduce the network traffic between parties as well as to optimize the size of the data.

Note that, in some cases, there might be found some edges, which are neither on the chosen shortest paths in the graph nor adjacent edges to the chosen shortest path, but the Bloom filters of these edges might be less than or equal to the Bloom filter of the shortest path. Hence, this kind of edges might be thought of as a false positive (we have given examples of this kind of edges at the end of the Sections 2.4 in Chapter 2, 3.5 in Chapter 3, etc.). Yet, it is not possible that the messages follow the edges which are distinct from the chosen shortest paths. Therefore, only the adjacent edges to the shortest paths are potential false positives in our routing model of networks.

1.6 Our Approach to Bloom Filters in Graphs

We consider the same routing scenario we described as above for a generalisation of Bloom filters in some special graphs throughout the thesis. We assume all the graphs we consider in this thesis are undirected. Because of the differences between the graph-theoretical properties of the graphs, the Bloom filters of the edges alter in each graph.

We assume that subsets to be represented by Bloom filters are not arbitrary subsets but satisfy certain fixed properties. Also, we assume that the objects that can be checked for being elements of the subsets are not arbitrary but also satisfy certain properties. This is the way that removing the randomness from the Bloom filter of the set. By considering this, we aim to both save space and avoid false positives by choosing positions for individual elements not at random, but using some rules.

We assume that there is a computer at each node of the network. We want to develop a system for sending messages from one computer to another within the graph. For each message, the sender computer chooses a specific path which the message must follow. We assume that the path chosen is always one of the shortest paths between the sender and the receiver. This is a special property of the subsets to be presented: they are not arbitrary sets of edges, but only shortest paths. Each computer receiving the message (unless it is the one where the message terminates) forwards it on. It does not send it back, but sends it along all other edges whose description is recognised by the Bloom filter describing the path. This is a special

property of edges to be checked for being on the path: they are not arbitrary edges, but only the edges incidental with the nodes of the path.

Our methodology in each chapter is as follows: by using an appropriate method encoding the edges, it is theoretically proved that the Bloom filters do not yield a false positive in a graph which we have chosen; then we compare our approach with other encoding methods which use other common labelling methods.

CHAPTER 2

RECTANGULAR GRIDS

2.1 Introduction

The results we obtained in this chapter has been published as a paper [29].

In this chapter and all other chapters, we have chosen a graph which has been used in modern computer networks. We build reasonably small Bloom filters invented by Bloom [7] for all edges in these graphs on a particular routing scenario.

In this chapter, we consider a routing scenario from one computer to one other computer in a graph called a rectangular grid which is a graph made of squares lying in rows horizontally and in columns vertically (see Figure 2.1).

Definition 2.1.1. *Rectangular grid: A rectangular grid is a graph* $G_R = (V_R, E_R)$ *with a set of vertices* V_R *and a set of edges* E_R *where* $V_R = \{(i,j) | i \in [0,M], j \in [0,N], M, N \in \mathbb{Z}\}$. *The vertices* (i,j) *and* (p,q) *are connected in a rectangular grid by an edge if and only if* $i = p$ *and* $j = q \pm 1$ *or* $i = p \pm 1$ *and* $j = q$.

The size of a rectangular grid is $M \times N$, that is, M links horizontally in each row

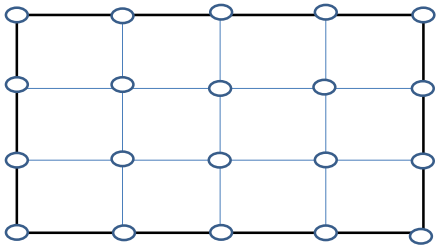


Figure 2.1: A rectangular grid contains a computer on each corner of each square.

and N links vertically in each column.

2.1.1 A Possible Routing Scenario

We assume that there are computers on every corners of squares and the messages are forwarded within it between computers via the edges.

We assume that from time to time a computer in the network may need to send a message to another computer in the network; then the sender computer encloses a header with the message, which describes exactly what path the message must follow. For this purpose, each edge is allocated its own Bloom filter in advance, and the path is represented as the set of its edges, that is, the Bloom filter of the path is the bitwise OR of the Bloom filters of the edges constitution the path. We assume that only shortest paths in the grid are used to deliver messages between the computers. The Bloom filter of the shortest path and message are forwarded to the receiver together, like in [13]. Once a computer, which is located on the shortest path between the sender and receiver computers, receives the message, it does not send it

back and forwards it along all other edges whose Bloom filter is less than or equal to the Bloom filter of the shortest path. Therefore, before sending the message to the next computer on the shortest path, the edge between computers is queried whether it is on the shortest path or not. Since, a computer might send the messages through all edges linked with itself. In this case, the adjacent edges to the shortest path are the potential false positives. In this routing scenario, false positives adjacent to the shortest paths produce extra network traffic.

2.2 Encoding Paths in a Rectangular Grid

We always assume the edges and paths in rectangular grid network are undirected. However, we sometimes mention the directions for the edges (e.g. Lemma 2.2.1). Hence, it is convenient to treat paths as directed paths. The edges in a rectangular grid lie either horizontally or vertically. Accordingly, in total, there are four directions for the edges in a rectangular grid, these are north ($\uparrow$), south ($\downarrow$), east ($\rightarrow$) and west ($\leftarrow$).

Lemma 2.2.1. *A directed shortest path between two distinct vertices in a rectangular grid consists of directed edges with at most two different directions which are not opposite of each other.*

Proof. Suppose the edges in a directed shortest path P have three different directions. More precisely, we may assume that the edges have directions of north, east and west in a directed shortest path (see Figure 2.2). Obviously, the directions of the edges in a path might be chosen within different combinations, yet the proof will be the

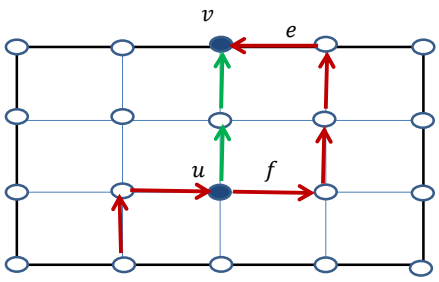


Figure 2.2: The red path is a directed path consisting of directed edges with more than two directions.

same for all possibilities. We pick a minimal fragment F, which is placed between two vertices u and v, of the path and contains the edges with the directions of $\rightarrow$, $\uparrow$ and $\leftarrow$ (e.g. a fragment starts with the edge f and ends with the edge e in the Figure 2.2). As known the directions $\rightarrow$ and $\leftarrow$ are the opposites of each others. When the first edge having direction of $\rightarrow$ and last edge having direction of $\leftarrow$ of the fragment have opposite directions, then all the rest of edges between them are oriented to the direction of $\uparrow$ (see for example the Figure 2.2). Hence, there is another path F' between the vertices u and v, which consists of the edges oriented to only the direction of $\uparrow$. Obviously the path F' is shorter than the fragment F between the vertices u and v. If we replace the fragment F with the path F' in the path P, this new path P' will be two edges shorter than the path P. This contradicts the assumption that P is a shortest path.

□

We aim to encode all edges in a rectangular grid. For this purpose, we introduce

two systems of coordinates for the grid: the one starting from the bottom-left corner and the second starting from the top-left corner of the grid. Accordingly, we introduce two notations for each vertex, with the letter u in the former system of coordinates and with the letter v in the latter. The vertex at the bottom left corner of the grid is denoted by the point $u_{(0,0)}$ which is the origin of the x/y coordinate system and the coordinates increase on the direction of north-east. The point $v_{(0,0)}$ is the origin of another x/y coordinate which is placed on the top left corner of the grid and the coordinates increase on the direction of south-east.

Each edge is represented by a Bloom filter with the length $m = 4 \times (M + N)$ bits. The Bloom filter consists of two equal length parts, where each half is called projection, which correspond to the two systems of coordinates and each projection is divided into two-bit length blocks.

Encoding an edge can be interpreted as the coordinates of the end vertices of the edges and the orientations that the edge is placed on horizontally or vertically in the grid. In other words, when an edge lies on the grid horizontally, then the first and the second halves of the Bloom filter of the edge will be based on the vertices $u_{(i,j)}$ and $v_{(i,N-j)}$ where $0 \leq i \leq M$ and $0 \leq j \leq N$. Both vertices $u_{(i,j)}$ and $v_{(i,N-j)}$ are placed on the left endpoint of the encoded edge, namely $u_{(i,j)} = v_{(i,N-j)}$.

However, if an edge is a vertical edge, the represented vertices will be $u_{(i,j)}$ which is on bottom endpoint of the edge and $v_{(i,N-j-1)}$ which is top endpoint of the same edge. Predictably, the first half of the Bloom filter of a vertical edge will be specified with the vertex $u_{(i,j)}$ and the second half will be constructed with $v_{(i,N-j-1)}$.

It is useful to imagine each half of the Bloom filter encoding an edge as a sequence

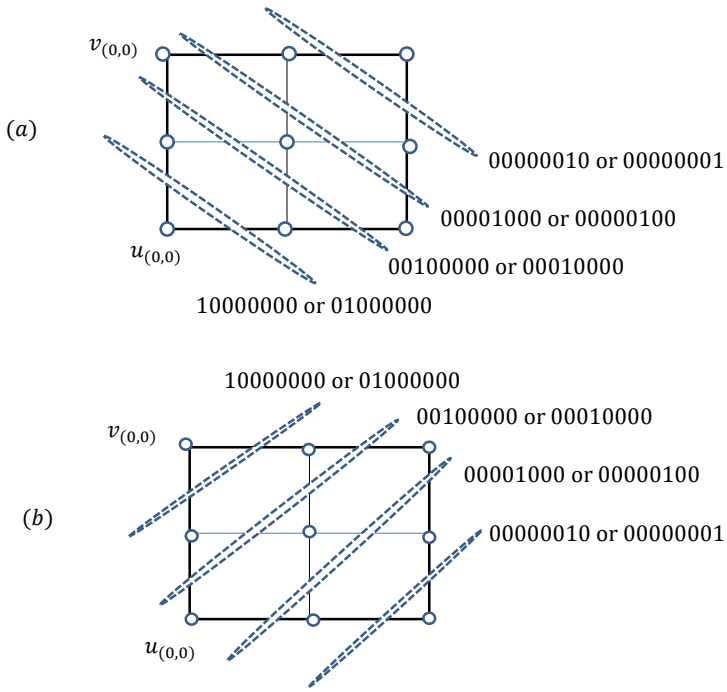


Figure 2.3: The graphs (a) and (b) show the first and second projections of the Bloom filters of the edges in a 2×2 sized rectangular grid, respectively.

of two-bit fragments which we call blocks. So there are $2(M + N)$ two-bit blocks in total and each half of the encoded edges has $(M + N)$ blocks. Exactly one block of each half contains 1, and this block is either 01 or 10. All other blocks are 00. To encode a vertical edge, the bit 1 will be situated in the first places of both $(i+j+1)$th and $(M+N+i+N-j-1+1)$th blocks in the first and the second half of the Bloom filter of the edge, respectively. To encode a horizontal edge, the bit 1 is placed in the second bit position in $(i+j+1)$th and $(M+N+i+N-j+1)$th blocks in the first and the second halves of Bloom filter of edge, respectively.

Hence, by adding 1 and the points of the corresponding vertices of the edges together the places of the blocks including 1 are specified in the Bloom filter of the edges. For instance: consider both graphs in the Figure 2.3 as a rectangular grid

with size 2×2, the first half of the Bloom filters of the edges are given with the corresponding edges in the Figure 2.3 (a) and the second half of the Bloom filters of the edges are given with the corresponding edges in the Figure 2.3 (b). When the Bloom filters of horizontal edges contain the block 01 on the corresponding block position, the Bloom filters of vertical edges include the block 10 on the corresponding block position.

A horizontal edge might be represented by the points $u_{(0,0)}$ and $v_{(0,2)}$, which are the left endpoint of the edge, in a rectangular grid with size 2×2. Then the block 01 will be placed in the $0 + 0 + 1 = 1$st and $2 + 2 + 0 + 2 + 1 = 7$th block positions among $2(2 + 2) = 8$ blocks. Note that the length of the Bloom filter of each edge is $4(2 + 2) = 16$ bits. So the Bloom filter of this horizontal edge will look like 0100000000000100. Now, a vertical edge with the points $u_{(0,0)}$ and $v_{(0,1)}$ which are the bottom and top endpoints of the vertical edge, respectively, in the grid with size 2×2. The block 10 will be situated in the $0 + 0 + 1 = 1$st and $2 + 2 + 0 + 1 + 1 = 6$th block positions. The Bloom filter of this vertical edge will be as 1000000000100000.

2.3 An Analysis of Bloom Filters of Paths

2.3.1 Positions of 1s in the Bloom filters

All possible shortest paths consist of $n \geq 1$ edges lying between two distinct vertices consecutively. Besides, all Bloom filters of edges include two 1s and the Bloom filter of a path is obtained by applying OR operation to the encoded adjacent edges lying

on the path together.

Theorem 2.3.1. *All Bloom filters of edges are unique in a rectangular grid.*

Proof. All edges of a grid sized $M \times N$ are encoded by $2(M + N)$ blocks and each block consists of two bits.

Suppose two horizontal edges e and f are two distinct edges in a rectangular grid. Hence the 1s of these edges take place in the second bit positions in corresponding blocks. The vertex of the corresponding horizontal edge e is $u_{(i,j)}$ ($= v_{(i,N-j)}$), and the vertex of the horizontal edge f is given with $u_{(k,l)}$ ($= v_{(k,N-l)}$), when we accept the left top and bottom corners of the grid as the origins of two different x/y coordinate systems. Both halves of the Bloom filter of horizontal edges are encoded with the left endpoints of the edges. The two main points behind the encoding edges are with regard to the corresponding vertex of the edge and the orientations of the edge. The blocks containing 1s are placed in $(i + j + 1)$th and $(M + N + i + N - j + 1)$th block positions of the Bloom filter of edge e. Similarly, $(k + l + 1)$th and $(M + N + k + N - l + 1)$th blocks of the Bloom filter of edge f contain 1s.

Suppose the horizontal edges f and e are represented by exactly the same Bloom filters. That means both $i + j + 1 = k + l + 1$ and $M + N + i + N - j + 1 = M + N + k + N - l + 1$ occur at the same time. Hence,

$$i + j = k + l \tag{2.1}$$

$$i - j = k - l \tag{2.2}$$

This implies that $i = k$ and $j = l$. This contradicts the assumption that the edges e and f are situated in two distinct locations in the grid.

Now, we suppose that both e and f are vertical edges, then the Bloom filters of both edges contain 1s on the first positions in the corresponding blocks. When we assume these two edges are distinct and encoded by the same Bloom filter, then the same equations between the points of the corresponding vertices $u_{(i,j)}$, $v_{(i,N-j-1)}$ and $u_{(k,l)}$, $v_{(k,N-l-1)}$ above can be constructed and obviously the same contradiction, which is found for the horizontal edges, is obtained.

Finally, suppose two distinct edges e and f, where one is vertical edge and other one is horizontal edge, are represented by exactly the same Bloom filter. Yet, while the bit 1 in the Bloom filter of a vertical edge takes place in the first bit positions of the corresponding block, the bit 1 in the Bloom filter of a horizontal edge is placed in the second bit positions of the corresponding blocks. Hence, even if the blocks including 1s of these two distinct edges are placed on the same block positions, these 1s are never placed on the same bit positions. This contradicts that the edges e and f are represented by the same Bloom filter.

As a result of these three cases, all edges of a grid are encoded uniquely in a given size grid. □

Lemma 2.3.2. *Consider an undirected shortest path whose direction is it towards north-east (or, equivalently, south-west) between two distinct nodes, then the first half of the Bloom filter of the path contains a consecutive subsequence of blocks having the form* 01 *and* 10*.*

Likewise, if we consider an undirected shortest path whose direction is it towards south-east (or, equivalently, north-west) between two distinct nodes, then the second half of the Bloom filter of the path contains a consecutive subsequence of blocks having the form 01 *and* 10*.*

Proof. Consider an undirected shortest path P whose direction is it towards north-east between two distinct nodes denoted by $u_{(i,j)}$ and $u_{(x,y)}$ where $x \geq i$ and $y \geq j$ in a $M \times N$ sized rectangular grid. Let the edges directed to the north be denoted by N' and the edges directed to the east be denoted by E'. The sequence of the edges in the path P may have a form of $E', E', N', N', E', N', N', \ldots$. Hence, the sequence of the vertices of these edges has a form of $u_{(i,j)}, u_{(i+1,j)}, u_{(i+2,j)}, u_{(i+2,j+1)}, u_{(i+2,j+2)}, u_{(i+3,j+2)}$, $u_{(i+3,j+3)}, u_{(i+3,j+4)}, \ldots, u_{(x,y)}$. As seen, the first or the second component of ordered pairs of the consecutive vertices in the shortest path whose direction is it towards north-east changes 1 point. The places of the corresponding blocks in the first half of the Bloom filters of the edges are specified by the points of u and the values of all vertices u increase on the way of north-east. By the encoding method we introduced in the Section 2.2, the block positions of the non-zero blocks in the first half of the Bloom filter of the path P are obtained as $i+j+1, i+1+j+1, i+2+j+1, i+2+$ $j+1+1, i+2+j+2+1, i+3+j+2+1, i+3+j+3+1, i+3+j+4+1, \ldots, x+y+1$. Hence, these block positions of the non-zero blocks lie consecutively in the first half of the Bloom filter of the shortest path P.

Similarly, consider another shortest path P' whose direction is it towards south-east between two distinct nodes denoted by $v_{(i,N-j)}$ and $v_{(p,N-q)}$ where $p \geq i$ and

$q > j$ in a $M \times N$ sized rectangular grid. Let the edges directed to the south be denoted by S' and the edges directed to the east be denoted by E'. The sequence of the edges in the path P' may have a form of $E', S', E', S', E', \ldots$. The sequence of the vertices in the path P' has a form of $v_{(i,N-j)}, v_{(i+1,N-j)}, v_{(i+1,N-j+1)}, v_{(i+2,N-j+1)}$, $v_{(i+2,N-j+2)}, v_{(i+3,N-j+2)}, \ldots, v_{(p,N-q)}$. The first or the second component of ordered pairs of the consecutive vertices in a shortest path whose direction is it towards south-east changes 1 point. The places of the corresponding blocks in the second half of the Bloom filters of the edges are specified by the points of v and the values of all vertices v increase on the way of south-east. By the encoding method we introduced in the Section 2.2 the block positions of the non-zero blocks in the second half of the Bloom filter of the path P' are obtained as $i + N - j + M + N + 1, i + 1 + N - j + M + N + 1, i + 1 + N - j + 1 + M + N + 1, i + 2 + N - j + 1 + M + N + 1, i + 2 + N - j + 2 + M + N + 1, i + 3 + N - j + 2 + M + N + 1, \ldots, p + N - q + M + N + 1$. Hence, these block positions of the non-zero blocks lie consecutively in the second half of the Bloom filter of the shortest path whose edges are oriented to south and east (or equivalently north and west).

□

As an expected result of this Lemma, some single blocks in the second half of the Bloom filter of path consisting of the edges lying on the way of north-east or south-west contain two bits 1 at the same time. Similarly, the first half of Bloom filter of the paths on the way of south-east or north-west might include the blocks 11 between the other blocks.

It is interesting to observe that if all the edges of a specific path lie horizontally or vertically, then all blocks which include one bit 1 of the Bloom filter of the path come after each other without an interruption of a block such as 00 or 11.

2.3.2 The number of 1s in the Bloom filters

In this model the number of 1s in the Bloom filter of each edge is exactly 2 and they are not placed arbitrarily. Therefore, there is limited number of 1s in the Bloom filters of shortest paths. It is easy to count the number of the bits 1 in a Bloom filter of a shortest path, we will give the following Lemma without proof.

Lemma 2.3.3. *The number of* 1*s of the Bloom filter of a shortest path in rectangular grid is not greater than* $2n$*, where the number of edges of the path is* n*.*

By the lemma 2.3.2 the blocks 10 or 01 lie together in corresponding halves without an interruption of the blocks 00 or 11 in the Bloom filter of the path as relevant to the orientation of the path. So these blocks 10 or 01 represents the number of edges passed and there are n edges in total in a path. Hence, when the number of the bits 1 in one half of the Bloom filter of the path is n, the number of bits 1 will be $\leq n$ in the other half of the Bloom filter. Any half of Bloom filter of path does not necessarily have n 1s, since the 1s of corresponding half of an edge might be occupied with the 1s on the same bit of the same block of the Bloom filter of following edges in the path. More precisely, the path might contain the edges that are encoded on the same block position with the same pair bits including 1. Therefore, the bits 1s of the Bloom filter of these paths are $\leq 2n$.

For instance: if the path on the direction of north- east or south - west yields the Bloom filter with the edges represented in the first half of the string, then the bits 1 in the first half of Bloom filter gives the number of elements n. Namely, the first half of the Bloom filter of the path illustrates all edges passed, when the edges of the path are lying on the way of north-east or south-west. But this result is not necessarily valid for the path on the direction of south-west or north-east, since the blocks including one 1 lie together on the second half of Bloom filter of the path. Furthermore, the edges of a path are on the only one way, then both halves of the Bloom filter of the path will contain n 1s. As a consequence, the number of 1 of the Bloom filter of our model is at most $2n$.

2.4 Avoiding False Positives Adjacent to the Paths

Theorem 2.4.1. *The Bloom filter of a shortest path consisting of $n \geq 1$ edges in a rectangular grid model does not yield any false positives when links adjacent to the path are queried.*

Proof. In the proof we shall concentrate on one fixed node of a path and demonstrate that no more than one link will be recognised by its Bloom filter as the next link of the path.

The Bloom filter of a path in a given size $M \times N$ rectangular grid is obtained by following the $n \geq 1$ edges on the way of one of the shortest distance between two

distinct vertices. When a message directed to a specific way on an edge e, then it moves on one of the next three adjacent edges of e with the purpose of reaching its destination on a completed distinct path. When we show the path contains one of these adjacent edges of the edge e and other edges are adjacent to the path, then we will reach the result that the Bloom filter of the path, which is denoted by $\beta(P)$, including edge e does not yield any false positives which are adjacent to the chosen shortest path.

An edge e encoded with $4(M + N)$ binary bits is represented by $\beta(e)$ that is divided into two bits length blocks denoted by $\beta_1(e), \beta_2(e), \ldots, \beta_{2(M+N)}(e)$. The two individual bits constituting a block $\beta_p(e)$ will be denoted by $\beta_p^1(e)$ and $\beta_p^2(e)$. Note that the Bloom filters of the edges are divided into two equal parts and the positions of the blocks containing 1s of all Bloom filters of edges are specified with the points represented vertex u for the first half and vertex v for the second half. When an edge is horizontal in the grid, the bit 1 is placed in the second bit positions in the corresponding two distinct blocks of the edge and as known the remaining blocks are constructed by the bit 0. These 1s are shown for an edge given with the points $u_{(i,j)}$ and $v_{(i,N-j)}$ by $\beta^2_{(i+j+1)}(e)$ and $\beta^2_{(M+N+i+N-j+1)}(e)$ where $(i + j + 1)$ and $(M + N + i + N - j + 1)$ represent the order of the blocks of the edge. Note that we replace $(i + j + 1)$ with k and $(M + N + i + N - j + 1)$ with l to make the indices simpler.

If the edge e is vertical in the grid, then 1s of the blocks are in the first positions of the corresponding blocks. These bits are shown as $\beta_k^1(e)$ and $\beta_{l-1}^1(e)$, since the represented vertices of vertical edge e are $v_{(i,j)}$ and $u_{(i,N-j-1)}$. As seen each Bloom filter of edge includes only two 1s and they are not placed arbitrarily in the relevant

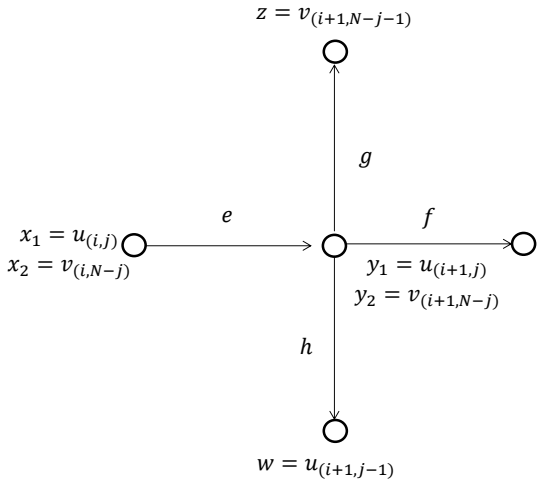


Figure 2.4: An edge e is from the shortest path and the other edges f, g and h are adjacent edges to the edge e. The edges e, f, g and h are encoded by the vertices x_1 and x_2; y_1 and y_2; y_1 and z; w and y_2, respectively.

blocks.

Suppose the next three adjacent edges of the edge e are called f, g and h (see Figure 2.4). We already know that the Bloom filters of edges $\beta(e), \beta(f), \beta(g)$ and $\beta(h)$ are distinct by Theorem 2.3.1. There are four possible ways of north, east, south or west for a message moving on an edge which lies on a specific path. Now we look for the false positives of the Bloom filters of the paths for these four possible directions.

Firstly, we suppose that the edge e is on the path where the message moves on the direction of east and the represented vertices of e are $u_{(i,j)}$ and $v_{(i,N-j)}$ (see Figure 2.4).

The next ways followed by the message after the edge e are east, north and south on the edges f, g, h which are represented by the vertices $u_{(i+1,j)}$ and $v_{(i+1,N-j)}$, $u_{(i+1,j)}$

and $v_{(i+1,N-j-1)}$, $u_{(i+1,j-1)}$ and $v_{(i+1,N-j)}$, respectively. By the encoding method we add the values of the points of represented vertices and 1 together to find the block positions of the pairs including 1s of the edges. Hence the bits $\beta^2_k(e)$ and $\beta^2_l(e)$ are 1. As a result of the encoding method the bits $\beta^2_{(k+1)}(f)$ and $\beta^2_{(l+1)}(f)$, $\beta^1_{(k+1)}(g)$ and $\beta^1_{(l)}(g)$, $\beta^1_{(k)}(h)$ and $\beta^1_{(l+1)}(h)$ are 1.

Note that when the edge e lies on the path, then the bits of corresponding blocks of the Bloom filter of the path are $\beta^2_{(k)}(P) = 1$ and $\beta^2_l(P) = 1$. That means the places of the 1s of the Bloom filter of the path depends on the locations of the 1s of the edges on the path.

As known from the standard Bloom filter when an element is less than or equal to the Bloom filter of the set in all bit positions, then the tested element is either an element of the set or a false positive.

The proof proceeds by considering several cases: that f is on the path, that g is on the path and that h is on the path.

If both $\beta^2_{(k+1)}(f) \leq \beta^2_{(k+1)}(P)$ and $\beta^2_{(l+1)}(f) \leq \beta^2_{(l+1)}(P)$ occur at the same time, then we say whether the path includes the edge f or f is a false positive of the path. But the bit of the path $\beta^1_{(k+1)}(P)$ is 0. If the path constructed by the edges on the direction of north or east, then the first half of the Bloom filter of path contains the blocks which consist of one 0 and one 1 consecutively (see Lemma 2.3.2). That means $\beta^1_{(k+1)}(g) = 1 > \beta^1_{(k+1)}(P) = 0$. Namely, the Bloom filter of path is not greater than or equal to the Bloom filter of the edge g in all bit positions. As a result when the Bloom filter of the edge f is smaller than or equal to the Bloom filter of the path P in all bits positions, more precisely $\beta(f) \leq \beta(P)$, then the edge g is definitely not on

the path.

When both Bloom filters of e and f are smaller than or equal to the Bloom filter of path in all bits positions at the same time, then the both consecutive blocks $\beta_k(P)\beta_{(k+1)}(P)$ of the Bloom filter of the path looks like 0101. This is because of the encoding method we use. Namely, the values of the points, which specify the position of the block including the bit 1 in the first half of the Bloom filter of the edges, of the corresponding edges are increasing on the way of north-east and the edges e and f are placed on the directions of east or north. Under these circumstances, when the edge e lies on the path and $\beta(f) \leq \beta(P)$, then $\beta^1_{(k)}(P) = 0$. But, $\beta^1_{(k)}(h) = 1 > \beta^1_{(k)}(P) = 0$, then we say that the edge h is not on path. The Bloom filter of edge h is not smaller than or equal to the Bloom filter of the path in all bits positions.

Now let us consider another case, if $\beta(g) \leq \beta(P)$ in all bits positions, then we say the edge g lies on the path after the edge e or a false positive of the Bloom filter of the path. Again, when both Bloom filters of e and g are smaller than or equal to the Bloom filter of path in all bits positions and take places on the orientation of east and north which are the ways that make the blocks including one 1 and one 0 in the first part of the Bloom filter of the path to lie consecutively. So $\beta^2_{(k+1)}(P) = 0$, but we obtain that $\beta^2_{(k+1)}(f) > \beta^2_{(k+1)}(P)$, and hence the edge f is not on the path. Subsequently, by comparing the first bits of the (k)th blocks both Bloom filter of the path and the Bloom filter of the edge h, we reach the result that h is definitely not an edge of the path.

Hence, both $\beta(e) \leq \beta(P)$ and $\beta(g) \leq \beta(P)$ occur at the same time, then both $\beta(f)$ and $\beta(h)$ are not smaller than or equal to the Bloom filter in all bits positions.

Finally, if both $\beta(e) \leq \beta(P)$ and $\beta(h) \leq \beta(P)$ in all bits positions, then the second part of the Bloom filter of the path will contain the blocks consisting of one 0 and one 1, consecutively. Since the values of the point y of the vertices v increase on the way of south-east and both e and h are on the direction of east and south, respectively. But $\beta^2_{(l+1)}(f) > \beta^2_{(l+1)}(P)$ and $\beta^1_{(l)}(g) > \beta^1_{(l)}(P)$ are obtained. Hence, when $\beta(h) \leq \beta(P)$, the edges f and g are definitely not on the path.

As a result, the edge e is followed by one of the adjacent adges f or g or h in order to have a path with the number of edges $n > 1$. As shown above when e lies on the path and any adjacent edges of e is smaller than or equal to the Bloom filter in all bits positions at the same time, then the other adjacent edges are not definitely on the path. So the Bloom filter of an adjacent edge of e which is $\leq \beta(P)$ lies on the path after the edge e. Consequently, we are able to make a certain decision for an edge of grid whether in the path or not. As seen a distinct path does not yield a false positive.

Consequently, we searched for the results, when the edge e is on the way of the message directed to east. When the edge e is supposed to be lying on the path that the message is directed to north or west or south, the same results above are obtained. The same method using comparisons between the Bloom filter and the edges should be applied.

□

Note that, it is still possible that some Bloom filters of edges, which are actually not on the path, might be less than or equal to the Bloom filter of the path in all bits

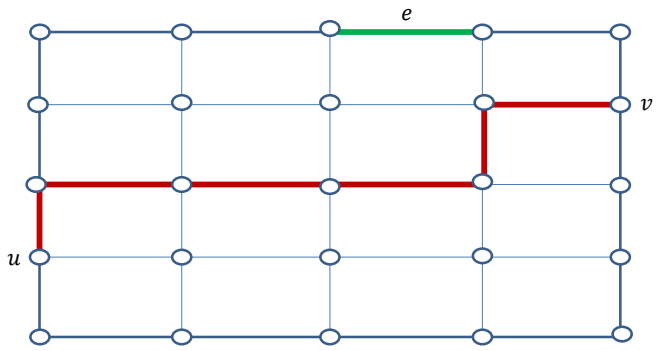


Figure 2.5: The Bloom filter of the edge e is less than or equal to the Bloom filter of the shortest path between the nodes u and v in all bits positions.

positions (see Figure 2.5). Then these edges might be thought that they are lying on the path, in other words they are false positives of the Bloom filter of the path. As seen in the Figure 2.5, the edge e is not an adjacent edge of the shortest path between the vertices u and v, yet the Bloom filter of the edge e is less than or equal to the Bloom filter of the path. Hence, one might think the edge e is a false positive of this path. Yet a message starts its journey in a specific path and the path is constructed by consecutive edges between two distinct vertices. As it is shown above that we are able to say that a message follow the edges lying on the path, since the Bloom filter of the path specifies all the edges both on the path and the adjacent edges of the path. Hence the possible false positives of the set are eliminated with the Bloom filter of the specific path. The edge e is not an adjacent edge of the path, therefore it does not affect additional message traffic.

2.5 Comparing our Approach with Arbitrary Coding Methods

In this section, we consider two possible encoding models that we compare with our model.

One encoding model is that each edge is represented by one bit. In that model, the bits 1s in the Bloom filter of the path shows the edges on the path. If one bit per edge was used in rectangular grid, the length of the Bloom filter would be $2MN + N + M$ which is the total number of the edges of $M \times N$ sized grid. Contrary to this, our approach for the length of Bloom filter is $4(M+N)$, and $4(M+N) < 2MN+N+M$ for the sufficiently large values of M and N where $M \geq 4$ and $N \geq 3$ or $N \geq 4$ and $M \geq 3$. This encoding method does not produce a false positive by the Theorem 1.3.1, yet it generates longer binary arrays than our approach.

The other encoding model is the standard Bloom filter. If our encoding method would had been constructed by putting the bits 1s in arbitrary bit positions in the Bloom filters of edges, we would certainly obtain false positives. Our model can be treated as an application of Bloom filters (indeed, we use disjunction and comparison of binary arrays precisely in the way it is done when the Bloom filters are used). The possible false positives of our model are the adjacent edges of the chosen path that the messages must follow. Since as a possible scenario the messages would be sent along these adjacent edges instead of following only the specified path.

To observe the probability of false positives, we use our parameters which are the

length of Bloom filter of edges m, number of edges in the shortest path n, and the number of 1s k in the Bloom filter of the edges, in the standard Bloom filter. In a rectangular grid, the number of edges lying on a path takes its maximum value with $(M + N)$, when the two distinct vertices stand on the corners, which are opposite each other, in the rectangular grid. The length of the Bloom filter of each edge and the length of the Bloom filter of any path are $4(M + N)$ (see Section 2.2).

The formula for the optimum value of the number of the bits 1 of an element is obtained as $k = \lceil \ln 2 \times \frac{m}{n} \rceil$ when the probability of the false positives is minimized, [46]. The ratio of the length of the Bloom filter and the number of edges of the path of our model is $\frac{m}{n} = \frac{4(M+N)}{(M+N)} = 4$ where n takes its maximum value. Hence the number of 1 of an edge is $\ln 2 \times 4 \approx 2.772$. Yet, the number of 1 in the Bloom filter must be an integer. One might take 2 or 3 as optimal value of k. This approximation is the recommendation, which is 2, for the number of 1 of an encoded edge in our model. Hence, in order to obtain the minimum probability of the false positive of the Bloom filter of the path, the number of 1 of the all Bloom filters of the edges k can be an integer 2.

When the number of edges on the path is maximum then the probability of false positives of the Bloom filter of the path would have been obtained as (2.3).

$$\left(1 - e^{\frac{-2(M+N)}{4(M+N)}}\right)^2 \approx 0.1548 \tag{2.3}$$

If the number of the bit 1 would have been taken as 3, then the probability of the false positives would have been obtained as the following inequality.

$$\left(1 - e^{\frac{-3(M+N)}{4(M+N)}}\right)^3 \approx 0.1468 \tag{2.4}$$

Obviously, the number of edges in a shortest path might be less than $M+N$. When m and k are fixed and n decreases, then the probability of false positives decreases (see table (b) in the Figure 2.6). Hence, the approximation 0.15 is the maximum value of the probability. If the bits 1, whose number is 2, would have taken place in the Bloom filter of the edges randomly, the probability of having no false positives in a path would be approximately $0.85^{2(M+N)}$ where $2(M+N)$ is the number of adjacent edges of the shortest path with maximum number of edges in the grid. When $k = 3$, then the probability of having no false positives in a path would be approximately $0.86^{2(M+N)}$.

2.5.1 Some Examples of Probability of False Positives

We observe an example to show what changes when the number of edges and the length of the Bloom filter take the variety of values where the number of the bit 1 is fixed in the Bloom filters of edges in a rectangular grid (see table (a) in the Figure 2.6). Note that, we obtain the results in the Figure 2.6 when we assume the bits 1 in the Bloom filter of the edges are placed randomly. As seen in the table (a) of

(a)

$M+N$	n	$m=4(M+N)$	*probability of false positives*
10	10	40	≈ 0.15%
10	9	40	≈ 0.13%
25	20	100	≈ 0.10%
63	32	252	≈ 0.05%
500	80	2000	≈ 0.005%
20000	500	80000	≈ 0.0001%

(b)

$M+N$	n	$m=4(M+N)$	*probability of false positives*
2000	100	8000	≈ 0.0006%
2000	200	8000	≈ 0.002%
2000	320	8000	≈ 0.005%
2000	500	8000	≈ 0.013%
2000	900	8000	≈ 0.04%
2000	1500	8000	≈ 0.09%

Figure 2.6: The false positive rates are given when the number of edges in a shortest path and the length of the Bloom filter take variety values.

the Figure 2.6, the probability of false positives takes its maximum value when the number of edges in the shortest path is maximum. When we consider the formula (2.3), the probability of false positives depends on the ratio $\frac{n}{m}$ where k is fixed. In the table (b), we observed the ratio of false positives when the length of the Bloom filter and the number of the bits 1 are fixed. In that case, when the number of edges in the shortest path increases, then the probability of false positives increases.

CHAPTER 3

KING'S MOVE GRIDS

3.1 Introduction

A realistic network might have a complicated structure which has large number of nodes linked with large number of edges. Building a reliable communication between nodes while eliminating the errors from this kind of network, we consider another graph as a network. This graph looks like the graph presented on the Figure 3.1 that is called a king's graph.

Definition 3.1.1. *King's graph: A king's graph is a graph $G_K = (V_K, E_K)$ with a set of vertices V_K and a set of edges E_K where $V_K = \{(i,j)|i \in [0,M], j \in [0,N], M, N \in \mathbb{Z}\}$. The verices (i,j) and (p,q) are connected in a king's graph by an edge if and only if $i = p$ and $j = q \pm 1$ or $i = p \pm 1$ and $j = q$ or $i = p \pm 1$ and $j = q \pm 1$.*

By definition of a king's graph, there are three types of edges which are vertical edges with endpoints (p,q) and $(p,q-1)$ or (p,q) and $(p,q+1)$, horizontal edges with endpoints (p,q) and $(p-1,q)$ or (p,q) and $(p+1,q)$ and diagonal edges with endpoints (p,q) and $(p+1,q+1)$ or (p,q) and $(p-1,q-1)$ or (p,q) and $(p+1,q-1)$ or

(p, q) and $(p-1, q+1)$ where $p, q \in \mathbb{Z}$ in a king's graph. A king's graph is formed as squares and additional two diagonal edges to every square in a rectangular grid (see Figure 3.1). It can be concluded that a rectangular grid is a subgraph of king's graph. Unlike the orientations of the edges in a rectangular grid, the straight segments of the edges are vertical, horizontal and diagonal in a king's graph. The king's graph offers more links than a rectangular gird for supporting the communication between nodes. The way to build Bloom filters that we introduce in this chapter prevents the false positives in order to reduce network traffic.

A king's graph is basically inspired by the movement of king on a chess board. According to the chess board rules, the king is able to move one square horizontally, vertically or diagonally. Therefore, any vertex is connected with at most 8 links in a king's graph.

Graph-theoretical properties and the applications of king's graph has already been studied by [4] [23] [24] [26]. As a model, the king's graph has been chosen by [53] for monitoring vehicles, since the graph-theoretical properties of king's graph increase the connections between target vehicles.

3.1.1 A Routing Scenario in a King's Graph

The computers are placed on the corners of each single square in a king's graph (see Figure 3.1). Note that, the diagonal edges do not intersect each other. Hence, there is not a node on the intersection of the diagonal edges (see Figure 3.1). A message is forwarded through the shortest paths between two distinct nodes.

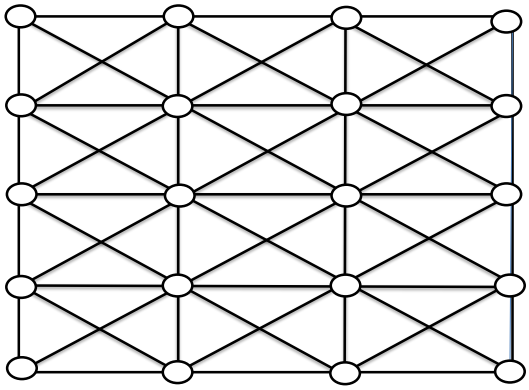


Figure 3.1: A king's graph of size 3×4 with a computer on each vertex of squares.

The routing is provided via shortest paths and the path is chosen by the sender in advance. In order to prevent a message from distribution to every node, the code of the path and the message are forwarded together, [13]. The Bloom filter of the shortest path is found by applying bitwise OR operation to the Bloom filters of all edges on the path. When a computer on the shortest path receives the message, it does not send the message back and forwards it to next computer on the shortest path. Hence, as usual, we aim to build an appropriate encoding method for all edges in king's graph to obtain zero false positives.

3.2 Coding Edges in a King's Graph

In order to prevent confusion, we denote the diagonal edges directed to north-west (or south-east) with $\nwarrow$ and north-east (or south-west) with $\nearrow$.

The size of a king's graph is $M \times N$ where M and N are the number of edges on

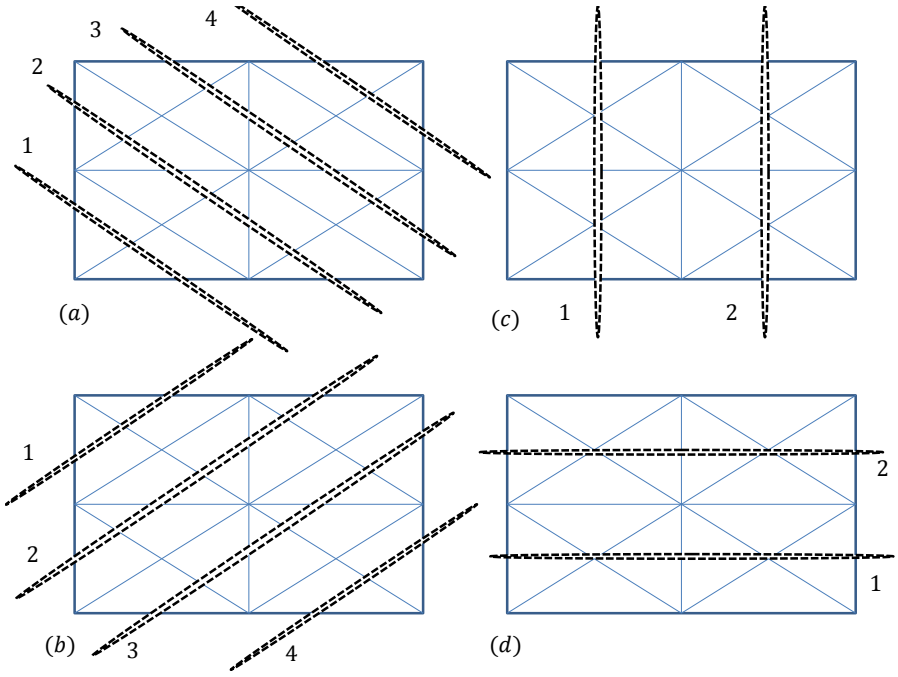


Figure 3.2: The rows in the graphs $(a), (b), (c)$ and (d) encode the first, second, third and fourth projections of Bloom filters of edges, respectively.

the horizontal and the vertical lines, respectively.

In this encoding method, we use imaginary rows intersecting edges like in the Figure 3.2. Because of the orientations of the edges in a king's graph, there are four different directions of rows which are necessary for obtaining sufficiently large Bloom filters of all edges. These rows intersect the edges on directions of north-east (south-west), north-west (south-east), north (south) and east (west) (see Figure 3.2). Obviously, the number of rows on each of two directions of north-east and north-west is $M + N$, the number of vertical rows is M, which is the number of horizontal edges on a side of king's graph, and the number of horizontal rows is N that is the number of vertical edges on a side of king's graph.

The Bloom filter of an edge consists of four parts called projections and each

projection is divided into four-bit fragments which we call blocks. In king's graph, all horizontal, vertical and diagonal edges ($\nwarrow$ and $\nearrow$) are represented by four-bit blocks $1000, 0100, 0010$ and 0001, respectively.

The positions of representative blocks of the edges in the Bloom filters depend on the rows that intersect the corresponding edge. The rows oriented to north-west encode the first projection of Bloom filter, they are numbered starting from the bottom left corner (see Figure 3.2). The rows oriented to north-east specify the block positions for the representative blocks in second projections of the Bloom filters of the edges, these rows are numbered starting from top left corner (see Figure 3.2). The third and the fourth projections of the Bloom filters of the edges are encoded by vertical rows, which are numbered starting from the left side of king's graph, and horizontal rows, which are numbered starting from the bottom side of king's graph, respectively (see Figure 3.2).

For instance, a horizontal edge is intersected by three rows, which have orientation of north-east, north-west and north, the representative block 1000 of the horizontal edge is placed in first, second and third projections of the Bloom filter with respect to the number of rows. The representative blocks of edges are placed on the Bloom filter on the block position which is the number of the row intersecting the corresponding edge.

The message distribution is provided by only shortest paths. Hence, after all edges are labeled by a Bloom filter, the Bloom filter of a path is obtained by applying bitwise OR operation to all Bloom filters of edges.

As expected, the length of the Bloom filter of an edge in a king's graph is $4 \times$

$3 \times (M + N)$, since there are four projections where the first two projections contain $(M + N)$ four-bit blocks, the third projection consists of M four-bit blocks and the fourth projection contain N four-bit blocks. The projections of the Bloom filters contain different number of blocks, since the total number of rows oriented to any direction is different from each other.

As an example, a horizontal edge e from a 3×4 sized king's graph given in the Figure 3.8 is encoded by $4 \times 3 \times (3 + 4) = 84$ bit length Bloom filter denoted by $\beta(e)$. Note that, the projections of the Bloom filter of the edge e should be denoted by $\beta^i(e)$ where $i \in \{1, 2, 3, 4\}$. The edge e is intersected by three rows which are second, fourth and first rows among the rows on the orientation of north-east, north-west and east, respectively. These three rows encode first three projections of the Bloom filter of the edge e, respectively. Therefore, the representative block 1000 of the edge e is placed on second block position in first projection which is $\beta^1(e) = 0000 - 1000 - 0000 - 0000 - 0000 - 0000 - 0000$, fourth block position in second projection which is $\beta^2(e) = 0000-0000-0000-1000-0000-0000-0000$ and first block position in third projection which is $\beta^3(e) = 1000-0000-0000$ (remember that the edge e is chosen from the graph given in the Figure 3.8). Yet, the fourth projection consists of all bits 0 which is $\beta^4(e) = 0000 - 0000 - 0000 - 0000$, since the edge e is not intersected by any horizontal row which encodes the fourth projections of Bloom filters of edges. The edge e is intersected by three rows, hence three blocks of Bloom filter of the edge e contain the bit 1.

3.3 Shortest Paths in a King's Graph

A diagonal edge is an edge with endpoints (i, j) and $(i+1, j+1)$ or (i, j) and $(i-1, j-1)$ or (i, j) and $(i+1, j-1)$ or (i, j) and $(i-1, j+1)$ where $i, j \in \mathbb{Z}$ in a king's graph. There can be found a path P which consists of one horizontal edge and one vertical edge between these endpoints which are (i, j) and $(i+1, j+1)$ or (i, j) and $(i-1, j-1)$ or (i, j) and $(i+1, j-1)$ or (i, j) and $(i-1, j+1)$. Obviously, the path P contains two edges that is longer than one diagonal edge between these endpoints. Hence, a shortest path in a king's graph consists of diagonal edges as much as possible along with vertical or horizontal edges (see Lemma 3.3.1 below). On the contrary of this, there is no diagonal edges in a rectangular grid. That is why, by comparison with the same size rectangular grid, there can be fewer edges in a shortest path between two distinct vertices in a king's graph than a shortest path in the same size rectangular grid.

Suppose that all vertices in a king's graph are represented by coordinates of a point in two dimensional space. We assume that the vertex at the left bottom corner of a king's graph is the origin of an x/y coordinate system. Note that the coordinates of endpoints differ one point depending on the orientation of edges. For instance; a horizontal edge lies on the x-axis, hence the endpoints of the horizontal edge differ one point on its first component of pair of form (x, y). Similarly, the ordered pair of endpoints of a vertical edge differ one point on its second component, and the ordered pair of endpoints of a diagonal edge differ one point of both first and second

components.

Lemma 3.3.1. *A shortest path in a king's graph does not contain both vertical and horizontal edges.*

Proof. Suppose a path $P_{(a)}$ contains both horizontal and vertical edges. There might be several parts that contain both kind of edges in this shortest path. Consider a fragment $F_{(a)}$, which contains one vertical and one horizontal edges, is situated in the shortest path $P_{(a)}$ (see the graph (a) in the Figure 3.3).

Suppose the fragment $F_{(a)}$ lies between two vertices given with the points (i, j) and (k, l) where $i \leq k - 1$ and $j \leq l - 1$ (see Figure 3.3 (a)). Suppose that the fragment $F_{(a)}$ starts with a vertical edge with the endpoints (i, j) and $(i, j + 1)$ and ends with a horizontal edge with the endpoints $(k - 1, l)$ and (k, l). The $n \geq 0$ edges between them are diagonal edges. Therefore, there are in total $n + 2$ edges between the vertices (i, j) and (k, l) in the fragment $F_{(a)}$.

The sequence of the endpoints of n diagonal edges is $(i, j + 1), (i + 1, j + 2), (i + 2, j + 3), \ldots, (i + n, j + n + 1)$. Hence, it is obtained that the points $(k - 1, l)$ and $(i + n, j + n + 1)$ represent the same vertex. Therefore, the path lies between the vertices (i, j) and $(i + n + 1, j + n + 1) = (k, l)$. Note that, both first and second components of the ordered pair of endpoints of a diagonal edge differ 1. It is obvious that these endpoints (i, j) and $(i+n+1, j+n+1)$ have a form of a sequence of several diagonal edges, therefore there can be found another path $F'_{(a)}$ with all diagonal edges between the vertices (i, j) and (k, l). In other words, there is another path, which is shorter than the fragment $F_{(a)}$ including both vertical and horizontal edges, between

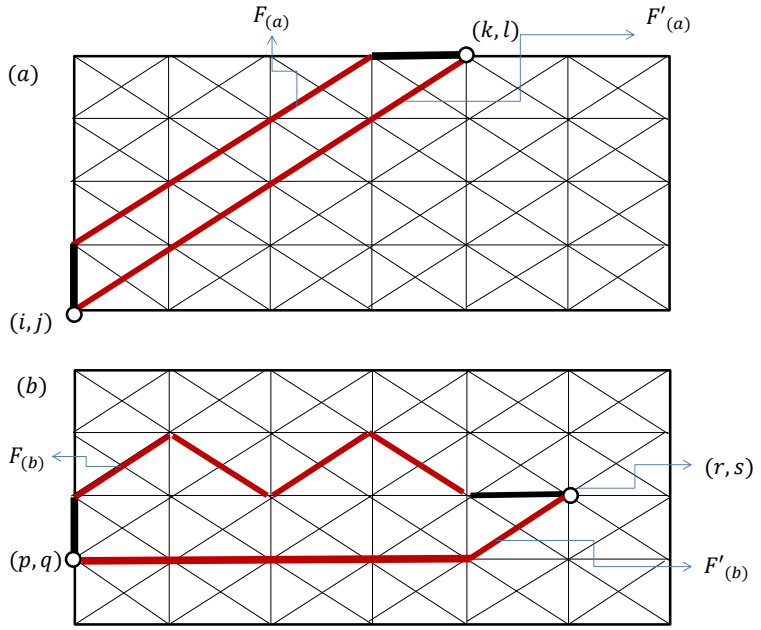


Figure 3.3: The fragments containing one horizontal and one vertical edges and the paths between the given endpoints.

vertices (i,j) and $(k,l) = (i+n+1, j+n+1)$ with $n+1$ diagonal edges. When we replace the $F_{(a)}$ with $F'_{(a)}$ in the path $P_{(a)}$, then we obtain another path $P'_{(a)}$ which is shorter than the path $P_{(a)}$.

Suppose another path $P_{(b)}$ and a fragment $F_{(b)}$ of this path contain both horizontal and vertical edges (given in the Figure 3.3 (b)). The fragment $F_{(b)}$ of the path $P_{(b)}$ lies between the vertices (p,q) and (r,s) where $p \leq r-1$ and $q \leq s-1$, and consists of one vertical, one horizontal and $m \geq 0$ diagonal edges.

Suppose the endpoints of the vertical edge are (p,q) and $(p,q+1)$, and the endpoints of the horizontal edge are $(r-1,s)$ and (r,s). All $m \geq 0$ edges between these two edges are diagonal and the sequence of endpoints of m diagonal edges is $(p,q+1),(p+1,q+2),(p+2,q+1),\ldots,(p+m,q+1)$. Hence, $(p+m,q+1) = (r-1,s)$,

this implies that $(p+m+1, q+1) = (r, s)$.

A path between the endpoints of (p, q) and $(p+m, q)$ lies on x-coordinate with m number of horizontal edges, and there can be a diagonal edge between the points $(p+m, q)$ and $(p+m+1, q+1)$. Hence, there can be found at least a path $F'_{(b)}$ which is one edge shorter than the fragment $F_{(b)}$ between the endpoints (p, q) and $(p+m+1, q+1) = (r, s)$. When we replace the $F_{(b)}$ with $F'_{(b)}$ in the path $P_{(b)}$, then another path $P'_{(b)}$ which is shorter than $P_{(b)}$ can be obtained.

□

For the clarification of the expressions of following lemmas, we define ladders for a king's graph (see Figure 3.4).

Definition 3.3.2. *A ladder in a king's graph is a set of edges which satisfies the following properties.*

- *A ladder is bounded by two lines consisting of concatenated edges which have same orientations.*
- *The edges having same orientations in a ladder are parallel to each other.*
- *The width of a ladder* $|w|$ *is* 1 *which is the length of one edge.*

There are three types of ladders depending on the orientations of the edges on the boundary lines. The ladders are named after the edges constituting boundary lines. For instance, if the boundary lines consist of diagonal edges, then the ladder is called *diagonal ladder*. Similarly, when boundary lines consist of vertical or horizontal edges, then the ladder is called vertical ladder or horizontal ladder, respectively.

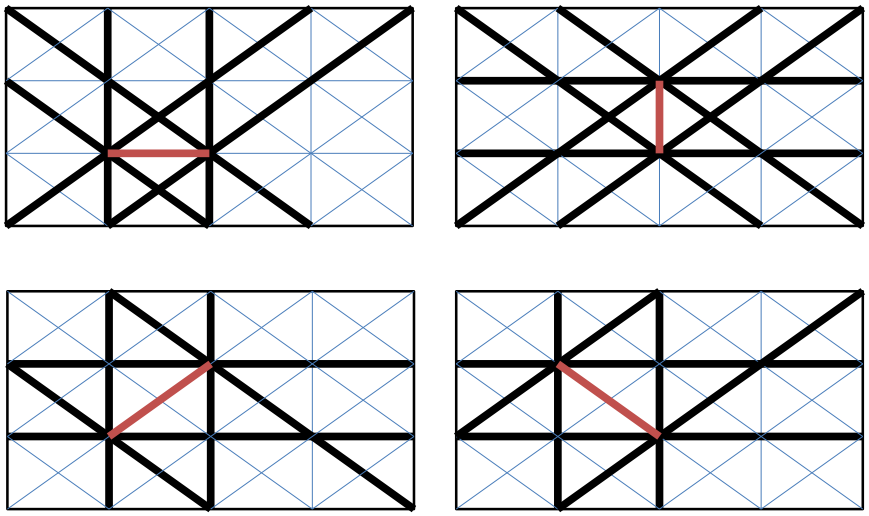


Figure 3.4: The black lines are the boundary lines of the ladders of the red edges in each graph.

Lemma 3.3.3. (1) *If a shortest path consists of diagonal and horizontal edges, with at least one horizontal edge every vertical ladder contain only one edge of this path.*

(2) *If a shortest path consists of diagonal and vertical edges, with at least one vertical edge every horizontal ladder contain only one edge of this path.*

Proof. A shortest path does not contain vertical and horizontal edges by Lemma 3.3.1, hence a shortest path may consist of horizontal and diagonal edges or vertical and diagonal edges, or a shortest path might consists of horizontal edges or vertical edges or diagonal edges.

Case 1: Suppose the shortest path lies between two distinct points given with ordered pairs (x_1, y_1) and (x_n, y_m) where $n, m \in \mathbb{Z}$ and $x_1 \leq x_n$. The boundary lines of vertical ladders are vertical, hence they are numbered by a point from x-coordinate. The sequence of the coordinates of boundary lines of vertical ladders is

$\{x_1, x_2, x_3, \ldots, x_{n-1}, x_n\}$ between the given points. Hence, each edge in the shortest path consisting of horizontal and diagonal edges takes place between the ordered pair of two consecutive lines. Suppose at least one vertical ladder contains more than one edges on the shortest path consisting of horizontal and diagonal edges.

Suppose a vertical ladder between the lines numbered $x_{(i-1)}$ and x_i where $i \in \{1, \ldots, n\}$ contains more than one edge on a shortest path P consisting of horizontal and diagonal edges. One edge in this ladder has endpoints on the lines $x_{(i-1)}$ and x_i, we assumed there is one more edge whose endpoints belongs to the lines x_i and $x_{(i-1)}$ in this ladder. In order to connect all edges on the shortest path P this vertical ladder must contain at least one more edge whose endpoints lie on the lines $x_{(i-1)}$ and x_i. The following edge of the shortest path P belongs to the next vertical ladder. Suppose there is one edge all other vertical ladders. There are three edges in one vertical ladder and one edge between all other pairs of the vertical lines. There are n vertical lines and $n-1$ ladders containing the edges from the shortest path P, hence the number of edges in the path P is $(n-1)-1+3 = n+1$. However, if each vertical ladder contains one edge from another path P' between the vertices (x_1, y_1) and (x_n, y_m), then the path P' has $n-1$ edges that is two edges shorter than the path P.

Suppose the last vertical ladder, which is bounded by the lines numbered x_{n-1} and x_n, contains two edges. Also, all edges are connected in a path. Hence, these two edges on the last ladder have a common vertex (x_n, y_m) (see Figure 3.5 (a)). In this case, the path ends with any of ordered pair (x_{n-1}, y_{m-1}) or (x_{n-1}, y_{m+1}) or (x_{n-1}, y_m). Yet, these vertices belongs to the vertical line numbered x_{n-1}. Hence,

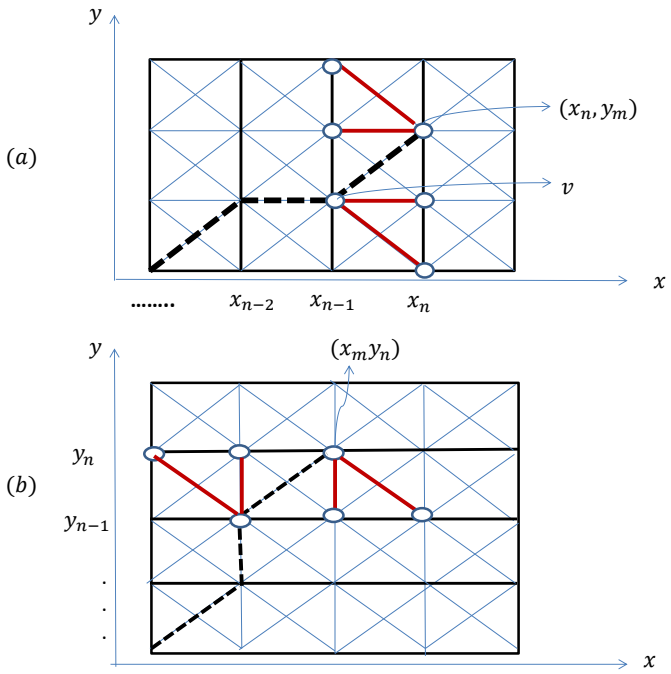


Figure 3.5: Dashed lines in both graphs are paths consisting of horizontal (vertical) and diagonal edges, and the red edges lie on the same vertical (horizontal) ladder with an edge from the shortest path.

there can be found another path between the lines x_1 and x_{n-1} and the last ladder is bounded with the lines x_{n-2} and x_{n-1}. Hence, the new path ending with the line x_{n-1} is shorter than the path which contains two edges between the ordered lines x_{n-1} and x_n, when each vertical ladder between lines is assumed to contain an edge.

Also, if the last two edges have a common vertex v belonging to the boundary line numbered x_{n-1}, then there is one edge linking with the endpoint (x_n, y_m), which is the endpoint of the path, from the vertex v. Otherwise the path ends with two distinct vertices and obviously it does not represent a shortest path.

Case 2: Likewise, the horizontal ladders are bounded by horizontal lines numbered by the points on y-coordinate. When a shortest path consisting of vertical and diagonal edges lies between the points (x_1, y_1) and (x_m, y_n) where $y_1 \leq y_n$, the sequence

of horizontal boundary lines of the shortest path is $\{y_1, y_2, y_3, \ldots, y_{n-1}, y_n\}$.

Suppose a horizontal ladder denoted by L between the lines numbered $y_{(j-1)}$ and y_j where $j \in \{1, \ldots, n\}$ contains more than one edge on a shortest path P'' consisting of vertical and diagonal edges, and all other horizontal ladders contain one edge from the shortest path P''. The consecutive edges in the ladder L has horizontal border lines $y_{(j-1)}$ and y_j, y_j and $y_{(j-1)}$, $y_{(j-1)}$ and y_j, respectively. In order to connect all edges on the shortest path P'' this horizontal ladder L must contain at least three edges. There are three edges in one horizontal ladder and one edge all other horizontal ladders. There are n horizontal lines and $n-1$ ladders, hence the number of edges in the path P'' is $(n-1)-1+3 = n+1$. However, when each horizontal ladder contains one edge between the vertices (x_1, y_1) and (x_m, y_n), then there can be found another path P''' which has $n-1$ edges that is two edges shorter than the path P''.

Suppose the last horizontal ladder contains two edges. Thus, the last two edges lie between the boundary lines y_{n-1} and y_n of a horizontal ladder by definition of the ladder given in this chapter, and these two edges must have common vertex. If the common vertex of last two edges belongs to the boundary line numbered y_n, then the path ends with any ordered pairs (x_{m-1}, y_{n-1}) or (x_m, y_{n-1}) or (x_{m+1}, y_{n-1}) (see Figure 3.5 (b)). As seen, the endpoints of the last edge lie on the horizontal line numbered y_{n-1}. Therefore, there can be found one other path, which is shorter than the path containing two edges in the last ladder, and it ends on boundary line numbered y_{n-1}.

Also, if the common vertex of the last two edges belongs to the boundary line numbered y_{n-1}, then there is only one edge linking with the endpoint (x_m, y_n), otherwise

it does not represent a shortest path.

□

Although, by Lemma 3.3.3 horizontal and vertical ladders contain at most one edge from a shortest path between two distinct vertices, in some cases they might contain more than one edge in the shortest path. We are going to summarise it with the following Remark.

One might think of a king's graph as a set of small squares formed of two horizontal, two vertical and two diagonal edges. For example, the number of n small squares are concatenated on a horizontal band, then these squares constitute a horizontal ladder. Yet, all the n squares on the horizontal band belongs to n vertical ladders where each ladder contains one square. This graph-theoretical property of of king's graph produces the following Remark.

Remark: When a shortest path consists of vertical and diagonal edges, then vertical ladders might contain more than one diagonal edge belonging to the shortest path. Similarly, when a shortest path consists of horizontal and diagonal edges, then horizontal ladders might contain more than one diagonal edges in the shortest path.

By Lemma 3.3.3, if a shortest path consists of horizontal and diagonal edges, then every vertical ladder contains only one edge of the path; and if a shortest path consists of vertical and diagonal edges, then every horizontal ladder contains only one edge of the path. Yet, because of that the individual squares containing an edge from the shortest path in the ladders might generate another horizontal or vertical ladder, hence some diagonal edges belonging the same shortest path might lie on the same

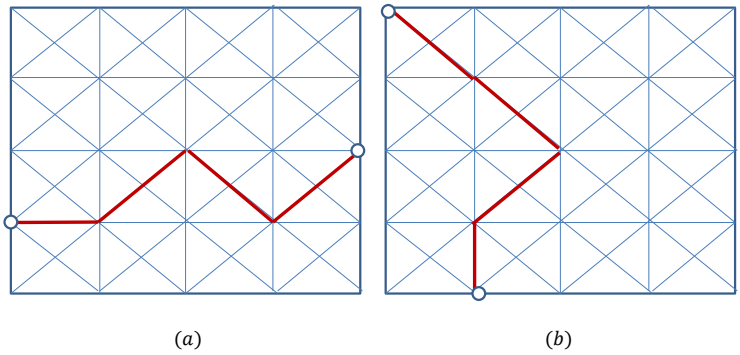


Figure 3.6: The vertical and horizontal ladders might contain more than one diagonal edges in the shortest paths.

ladder.

For example, although, the paths in the graphs given in the Figure 3.6 are shortest between two distinct nodes, the vertical and horizontal ladders contain more than one diagonal edges. Some of the squares containing one edge from the shortest path consisting of horizontal and diagonal edges might constitute a part of a horizontal ladder. In this case a horizontal ladder contains more than one diagonal edge from the shortest path, since the consecutive horizontal edges represent the boundary lines of a horizontal ladder (see Figure 3.6 (a)). Likewise, the squares containing an edge of the shortest path might constitute a vertical ladder, when the shortest path consists of vertical and diagonal edges. Hence, the vertical ladder might contain more than one diagonal edge, since the boundary lines of a vertical ladder are constituted by vertical edges (see Figure 3.6 (b)).

Lemma 3.3.4. *Any diagonal row intersects at most one edge belonging to a given shortest path.*

Proof. Suppose a diagonal row intersects two edges in a shortest path. Note that a diagonal row lies through a diagonal ladder whose boundary lines consists of diagonal edges. It is already known from the Lemma 3.3.1 that a shortest path do not contain both vertical and horizontal edges, therefore a single row do not intersect these two types of edges. If these two edges have the same orientation such as both vertical or both horizontal, then all edges in the shortest path between these two edges are diagonal edges which belong to the boundary diagonal line of diagonal ladders (see Figure 3.7 (a)).

Besides, the edges oriented to the same way are parallel in diagonal ladders. Suppose these two edges denoted by h_1 and h_2 are horizontal. The endpoints of the horizontal edges lying on these rows have a form of $h_1 = \{(i,j),(i+1,j)\}$ and $h_2 = \{(i+n,j+n),(i+1+n,j+n)\}$ (see Figure 3.7 (a)). The endpoints (i,j) and $(i+n,j+n)$, and $(i+1,j)$ and $(i+n+1,j+n)$ are directly connected by n diagonal edges. Therefore, while one of the horizontal edges h_1 and h_2 belongs to a path, then the other one cannot belong to the same shortest path. Otherwise, the path is longer than the shortest one with one additional edge.

Similarly, suppose a diagonal row intersects more than one vertical edge v_1 and v_2 belonging to the same shortest path (see Figure 3.7 (a)). The endpoints of the vertical edges have a form of $v_1 = \{(k,l),(k,l+1)\}$ and $v_2 = \{(k+n,l+n),(k+n,j+1+n)\}$. All edges between the endpoints of these vertical edges are diagonal and lie on the boundary lines of a diagonal ladder. Therefore, the shortest distance between the endpoints of vertical edges lying on the same diagonal ladder is provided by all diagonal edges. When these two same oriented edges are assumed to be in the

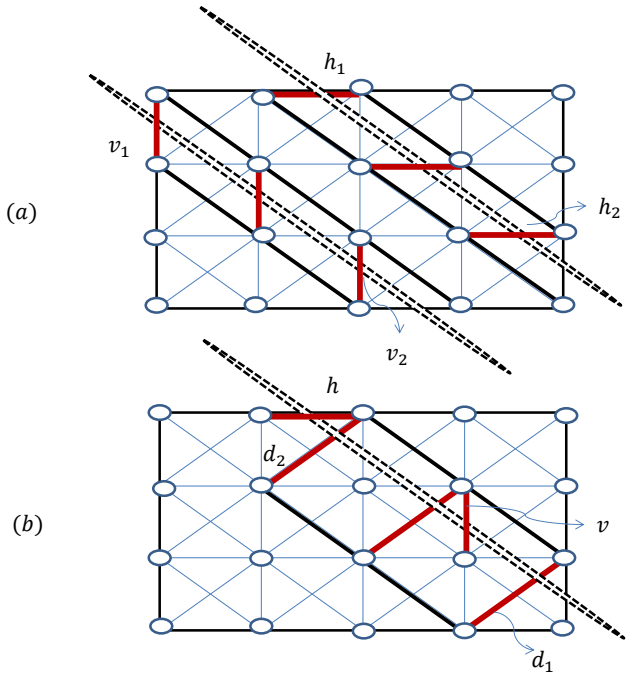


Figure 3.7: In the graph (a), parallel edges belonging to diagonal ladders are intersected by diagonal rows, and in the graph (b) the edges belonging to a diagonal ladder are intersected by a diagonal row

same path, then the path is not one of the shortest. It is two edges longer than the path which does not contain more than one vertical edge on a diagonal ladder.

Suppose a diagonal row intersects two diagonal edges d_1 and d_2 which belong to the same shortest path. The endpoints of diagonal edges belong to the diagonal boundary lines that are the shortest distance between nodes, and the edges having the same orientations in a ladder are parallel by the definition of the ladder (see Figure 3.7 (b)). Therefore, the number of edges in the path between the endpoints of the diagonal edges contradicts the assumption that this path is shortest.

Suppose a diagonal row intersects one diagonal and one horizontal (or vertical) edge where the edges belong to the same shortest path. We examine the existence of horizontal and diagonal edges intersected by the same diagonal row in the same

shortest path. The same approach is applicable to the existence of vertical and diagonal edges intersected by the same diagonal row in the same shortest path.

Now, we suppose the endpoints of a horizontal edge h are given with (i, j) and $(i + 1, j)$, and the endpoints of the diagonal edge d_1 are $(i + n, j - m - 1)$ and $(i+n+1, j-m)$. The endpoints $(i+1, j)$ and $(i+n+1, j-m)$ belong to the same diagonal line of diagonal ladder (see Figure 3.7 (b)). Since the differences the first and second components of the endpoints are $|i+1-(i+1+n)| = n$ and $|j-(j-m)| = m$. This is the distance between the edges, since all edges between these endpoints are diagonal and the shortest path between edges consists of as many as possible diagonal edges. The number of diagonal edges between the horizontal and diagonal edges is n which is equal to m. In total, there are $n + 2$ edges in this path containing these diagonal and horizontal edges.

On the other hand, while one of the endpoints of a diagonal edge belongs to the boundary diagonal line with the endpoint of horizontal (or vertical) edge, the other endpoint of the diagonal edge belongs to the another diagonal line (e.g. the endpoint $(i + n, jm - 1)$ of the diagonal edge d_1 belongs to the boundary line of a diagonal ladder containing the edge d_1 in the Figure 3.7 (b)). There are $|j-(j-m-1)| = m+1$ edges between the other endpoints of these two edges. Yet, $m + 1 < n + 2$ where $m = n$, this contradicts the shortest distance between the edges intersected by the same diagonal row.

□

Although, a diagonal row does not intersect more than one edge of the shortest

path, the horizontal and vertical rows might intersect more than one edge lying on a shortest path. This is because that horizontal and vertical rows lie through the vertical and horizontal ladders and by the Remark, vertical and horizontal ladders may contain more than one diagonal edge of a shortest path.

3.4 Bloom Filters of Shortest Paths

The graph-theoretical properties of a king's graph generates specific Bloom filters for edges. The following lemmas show which forms the Bloom filters of edges have in the king's graph.

Lemma 3.4.1. *All non-zero blocks, which include at least one bit* 1*, form a continuous sequence in corresponding projection of the Bloom filter of the shortest path in a king's graph.*

Proof. Suppose a block with all bits 0 appears between non-zero blocks in corresponding projection of the Bloom filter of the shortest path. Note that all edges are connected in a path, and they are intersected by numbered rows which lie consecutively on any direction. Hence, the representative blocks of the edges take place on the corresponding projections of the Bloom filter of the shortest path with respect to the number of rows. Therefore, the block 0000 represents a gap between the edges on the path. Yet, this is a contradiction with the fact that the path continuous. □

Lemma 3.4.2. *Only one projection, which is either third when the shortest path consists of vertical and diagonal edges or fourth when the shortest path consists of*

horizontal and diagonal edges, of a Bloom filter of shortest path may contain non-zero blocks including more than one bit 1.

Proof. A block contains more than one bit 1 when different representative blocks of edges coincide on the same block position in the Bloom filter of the shortest path. Namely, when the edges whose orientations are different from each other are intersected by the same row and lie on the same shortest path. By Lemma 3.3.4, a diagonal row, which encodes the first and second projections of the edges, does not intersect more than one edge lying on the shortest path. Therefore, the first and second projections of the Bloom filter of shortest path do not contain a block with more than one bit 1.

The vertical rows intersect edges through vertical ladders, and they encode third projections. By the Remark, a vertical ladder might contain more than one diagonal edges, when the shortest path consists of vertical and diagonal edges. Besides, horizontal rows intersect more than one edge, which belong to shortest paths consisting of horizontal and diagonal edges, in the horizontal ladders, and encode the fourth projections of Bloom filters of edges. Therefore, the representative blocks of the edges intersected by the same row coincide on the corresponding block positions in the either third or fourth projections of the Bloom filter of the shortest path.

□

3.5 Routing without False Positives

Theorem 3.5.1. *The Bloom filter of the shortest path in a king's graph does not produce a false positive adjacent to the path.*

Proof. A node in a king's graph has up to 8 links incidental with it. Once a message arrives at a node, then the node might send it through all links. As always, we assume a node does not send the message back. Once a message arrives from one node to another, then there are 7 possible destinations to move through. Possible false positives of the Bloom filter of the shortest paths are the adjacent edges to the shortest path. Accordingly, we need to show that while one of the next edges is on the shortest path, other 6 edges are adjacent edges to the shortest path.

Consider an edge e that the message passes through on the shortest path, and one of the next seven edges denoted by f, g, h, i, j, k and l is possible edge that the message might follow on the shortest path (see Figure 3.8).

The edges k and g are vertical edges coming after a horizontal edge e (see Figure 3.8). Yet, vertical and horizontal edges do not belong to a shortest path together by Lemma 3.3.1. Therefore, these two edges might be false positives. The rows encoding the first and second projections of the edge e intersect the edges g and k, respectively. The representative block of vertical and horizontal edges are 0100 and 1000, respectively, and these edges lie on the same diagonal rows. If the edge g is on the shortest path with the edge e, then the first projection of the Bloom filter of the shortest path contains a block with two bit 1. Similarly, if the edge k lies on the

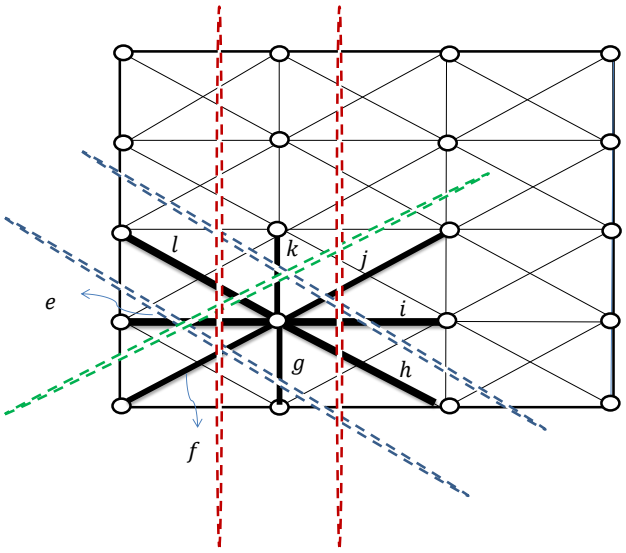


Figure 3.8: A horizontal edge e is one of the edge in the shortest path, and the edges f, h, i, j, k, l are the adjacent edges to the edge e.

same shortest path with the edge e, then the second projection of the Bloom filter of the shortest path includes the block with more than one bit 1. Yet, these contradict Lemma 3.4.2. Hence, while the edge e is on the path, the edges k and g are not false positives of a Bloom filter of the shortest path.

The edge e is a horizontal edge, and a shortest path contains one edge from every vertical ladder, while the shortest path consists of horizontal and diagonal edges by Lemma 3.3.3. Hence, the diagonal edges l and f might be false positives, since they belong to the same vertical ladder with the horizontal edge e. If the edges e and f or e and l are on the same shortest path, then the Bloom filter of the path contain the block with more than one bit 1 in its third projection. Since the representative blocks of these three edges are different from each other and take place in the same block position in the third projection of the Bloom filter of the shortest path. Yet,

this contradicts Lemma 3.4.2. Therefore, the edges f and l are neither on the path nor false positives.

On the other hand, the edges h, i or j might be the next edges after the edge e on the shortest path. These three next edges h, i or j belong to the same vertical ladder. By Lemma 3.3.3, a vertical ladder contain only one edge from the shortest path, while a shortest path consists of horizontal and diagonal edges. Hence, while one of these three edges is on the path, then the other two edges are possible false positives of the Bloom filter of the shortest path. The vertical row intersecting all edges h, i and j encodes the third projections of Bloom filters of edges. Since all these three edges lie on the same row, the representative blocks of the edges appear on the same block positions of Bloom filters. However, the block with more than one bit 1 does not appear in the third projection of Bloom filter of the shortest path, when the edges on the shortest path have orientations of horizontal and diagonal by Lemma 3.4.2. Therefore, while the edge i is the next edge after the edge e, then the edges j and h do not belong to the shortest path and they are not false positives (see Figure 3.8). Similarly, if the edges j or h is on the shortest path after the edge e, then the shortest path do not contain the edges i and h or j and i. Also, they are not false positives.

Besides, the edge e might be vertical edge, then the following edges can be examined by following the similar approaches above. If the edge e, which is assumed to be an edge on the shortest path, is a vertical edge, then the shortest path consists of vertical and diagonal edges. Therefore, the existence of the false positives are checked by taking the consideration into the horizontal ladders. As a consequence, by using

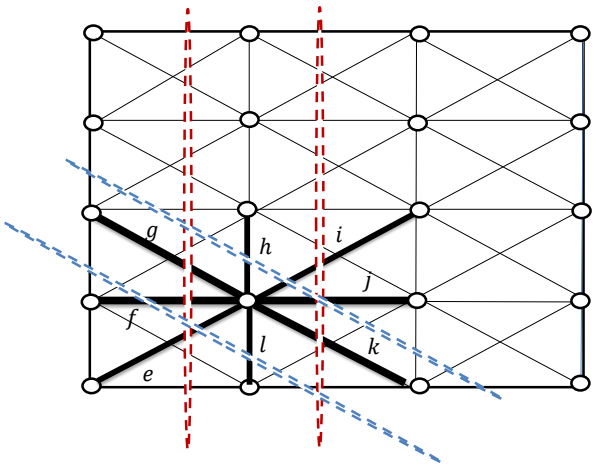


Figure 3.9: A diagonal edge e is one of the edge in the shortest path, and the edges f, h, i, j, k, l are the adjacent edges to the edge e.

the lemmas, we do not obtain false positives which are adjacent the shortest paths, when the shortest path consists of vertical and diagonal edges.

On the other hand, the edge e assumed to be an edge from the shortest path might be a diagonal edge (shown in the Figure 3.9), then the possible false positives of the Bloom filter of the shortest path are the edges f, g, h, i, j, k and l in the Figure 3.9.

The horizontal edge f and vertical edge l are coming after the diagonal edge e. The edge f might be a false positive, when the shortest path consists of diagonal and horizontal edges. Since, a vertical ladder does not contain more than one edge from the shortest path by Lemma 3.3.3. Also, the edge l is another potential false positive, when the shortest path consists of diagonal and vertical edges. Since, a horizontal ladder does not contain more than one edge from the shortest path by Lemma 3.3.3. Suppose the edge f is on the shortest path with the edge e, then the Bloom filter of the shortest path contains a block with more than one bit 1 in its

third projection, since both edges e and f are intersected by the same vertical row. Yet, this contradicts the Lemma 3.4.2. Hence, the edge f is not on the shortest path. Likewise, the edge l and e lie on the same row encoding fourth projection of Bloom filter of edges. Yet, the Bloom filter of the shortest path does not contain the block with more than one 1 in its fourth projection, since the shortest path is assumed to consist of vertical and diagonal edges with the edges e and l. By Lemma 3.4.2, the edge l is not a false positive.

On the other hand, any edges g, h, i, j, k might follow the edge e on the shortest path (see Figure 3.9). Suppose the edge k is on the shortest path with the edge e. The edges e and k belong to the same horizontal ladder and both are assumed to be the edges on the same shortest path. This happens when the shortest path consists of horizontal and diagonal edges by Lemma 3.3.3. Since, vertical ladders does not contain more than one edge from this kind of shortest paths, the edges i, j and g might be false positives. When we suppose any of the edges i or j or g is on the shortest path containing the edges e and k, then the Bloom filter of the shortest path contain a block with more than one bit 1 in its third projection. This contradicts the Lemma 3.4.2. Therefore, when both edges e and k are on the same shortest path, then the edges i, j and g are not on the same shortest path and they are not false positives.

The path containing the edges e and k consists of horizontal and diagonal edges. Therefore, the vertical edge h might be a false positive by Lemma 3.3.1. Suppose the edge h is on the shortest path. The Bloom filter of the path including the edges e and k might contain all the blocks 1000, 0010 and 0001, but it never contains the block

0100 which is the representative block of vertical edges. Therefore, the edge h is not a false positive of the Bloom filter of the shortest path when the edges e and k are on the shortest path.

The edges g, h, i, j might be assumed as an edge coming after the edge e on the shortest path, then following the same argument as we examined for the edge k, it is obtained that false positives do not exist in the king's graph.

As a possible scenario, the edge e (say in the Figure 3.8) might be the first edge of the shortest path. There are another 7 adjacent edges coming before the edge e. If the previous edges lie on the same vertical ladder with the horizontal edge e or they are vertical edges, then they obviously do not belong to the same shortest path with the edge e by Lemma 3.3.3 and Lemma 3.3.1, respectively. As expected, the Bloom filter of the path with these edges does not represent a shortest path, since the Bloom filter of the shortest path contain a block including more than one bit only in its fourth projection, when the shortest path consists of horizontal and diagonal edges by Lemma 3.4.2.

Although one of the edges belonging to the previous vertical ladder of the ladder containing the horizontal edge e probably be an edge of the shortest path, this edge is intersected by the rows that are numbered before the rows encoding the edge e. If there is an edge coming before the edge e from the previous vertical ladder on the shortest path, then the representative block of the previous edge appears on the previous or next block position of Bloom filter of the edge e on the the corresponding projection. Yet, when the edge e is assumed to be the first edge of the shortest path, the representative block of it is either the first or last block of the corresponding

projection of the Bloom filter of the shortest path. Hence, the previous or next blocks of the representative block of the edge e is 0000. Since, by Lemma 3.4.1, non-zero blocks in the Bloom filter of the shortest path have a form of consecutive sequence.

We examined that the edge e is horizontal as a first edge of the shortest path. Obviously, when as a first edge, the edge e is assumed to be a vertical edge or diagonal edge, then the same process as above can be repeated and similar result can be obtained.

□

Similarly in the rectangular grids, the king's graph has some edges, which are not adjacent to the shortest paths, that might be thought of as false positives of the Bloom filters of the shortest paths. As an example; in the Figure 3.10, the edge e satisfies this property. The Bloom filter of the edge e is less than or equal to the Bloom filter of the shortest path between the vertices u and v in all bits positions. This kind of edges does not affect the message delivery in our routing model.

3.6 Comparing Arbitrary Encoding Methods with our Approach in King's Graph

In this section, we compare two encoding models using arbitrary methods with our approach in a king's graph.

One method is to encode each edge with one bit. When the size of a king's graph

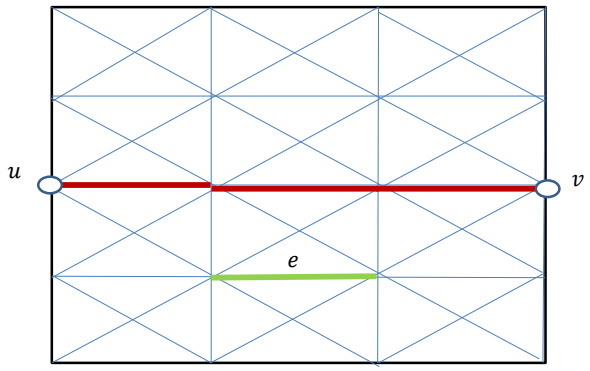


Figure 3.10: An edge e is not adjacent to the shortest path between the vertices u and v, but its Bloom filter is less than or equal to the Bloom filter of the shortest path between the vertices u and v in all bits positions.

is given with $M \times N$ where M and N are the number of horizontal and vertical edges on one side of a king's graph, there are $(M+1)(N+1)$ nodes and $(M+1)N + (N+1)M + 2MN = 4MN + M + N$ edges in total. If each edge was represented by one bit in the binary array, then the length of the Bloom filter of an edge would be $(M+1)N + (N+1)M + 2MN$ in a king's graph where the size of the king's graph is $M \times N$. We already consider the length of the Bloom filter in a king's graph as $12 \times (M+N)$. It is true that $12 \times (M+N) < (M+1)N + (N+1)M + 2MN$ for the big size of king's graph where $M \geq 7$ and $N \geq 5$ or $M \geq 5$ and $N \geq 7$. Therefore, the space is saved with the parameters used for coding edges in our method.

Another method to encode the edges is to use a standard Bloom filter. The standard Bloom filter is a random data structure, hence the false positives might be obtained in king's graphs. We compute the false positives ratio when the edges

in a king's graph are encoded by the method of the standard Bloom filter with the parameters that we have shown in Section 3.2.

The optimum number of bits 1 has been obtained as $k = \lceil \ln 2 \times \frac{m}{n} \rceil$, when the probability of false positives is assumed to be minimum, [46], where n is the maximum number of edges in a shortest path in our model and m is the length of the Bloom filter. To obtain the optimum number of k for the Bloom filters of edges in king's graph, we consider the maximum number of edges in a shortest path and the length of Bloom filters of the edges where they make the probability of false positives minimal. In this case, the maximum number of the edges in a shortest path in this model is M when $N < M$ or N when $M < N$, so $\max(M, M)$. Since, the shortest distance between nodes is provided by as many as possible diagonal edges and the both component of endpoints of diagonal edges differ one in each movement; and the points in two opposite corner of king's graph are $(0, 0)$ and (M, N) when a coordinate system is centered on one corner of king's graph. Hence, the optimum value of k is $\lceil \ln(2) \times \frac{m}{n} \rceil = \lceil \ln(2) \times \frac{12(M+M)}{M} \rceil = 16$ when $M = N$. Yet, this is higher than the value that we attain the number of bits 1 in the Bloom filter of an edge. The number of bit 1 in the Bloom filter of edges is maximum 4 in our encoding method.

When the edges were encoded by using the parameters m, k, n as above where we assume that $M = N$, then the probability of false positives in a king's graph would be calculated from the following formula. The probability of false positives (left hand side of the equality below) has been given in [7].

$$\left(1 - e^{\frac{-kn}{m}}\right)^k = \left(1 - e^{\frac{-16(M)}{12(M+M)}}\right)^{16} \approx 0.000009 \tag{3.1}$$

There are $6M$ or $6N$ adjacent edges to the shortest path with maximum number of edges. These are possible false positives of Bloom filter of the shortest path. Therefore, when the bits 1 in the Bloom filter of an edge in the king's graph are placed arbitrarily, then the probability of non-existent false positives would be $(0.999991)^{6M}$ when $M = N$.

Though the length of the Bloom filter of an edge in a king's graph $(12 \times (M + N))$ is three times longer than the Bloom filter of an edge in the same size rectangular grid $(4 \times (M + N))$; as well as the optimum number of the bits 1 in the Bloom filter of an edge in the king's graph (which is $\lceil \ln(2) \times \frac{m}{n} \rceil = \lceil \ln(2) \times \frac{12(M+M)}{M} \rceil = 16$, see Section 3.6) is eight times higher than the number of the optimum number of the bits 1 in the Bloom filter of an edge (which is $\lceil \ln(2) \times \frac{m}{n} \rceil = \lceil \ln(2) \times \frac{4(M+M)}{(M+M)} \rceil = 2$, see Section 2.5) in the rectangular grid, when the bits 1s are situated random places in the Bloom filters in both graphs. Hence, the probability of false positive of the Bloom filter of the shortest path in king's graph is rather less than the probability of false positive of the Bloom filter of the shortest path in rectangular grid obtained approximately 0.15. Also, by comparison with the rectangular grid network, a node in a king's graph connects with the nearest nodes through more links than the links in rectangular grid. Therefore, the shortest paths in king's graph contain less edges than the shortest paths in the same size rectangular grid.

CHAPTER 4

HEXAGONAL GRIDS

4.1 Introduction

We have studied another encoding method in another shape for a network in this chapter, then we published the results obtained in this chapter as a paper [30].

In this chapter we assume that the shape for a computer network is a hexagonal grid which looks like a graph given in the Figure 4.1. Like other graphs given in previous chapters, the communication in this network is provided via computers that are placed in each vertex of the grid. We repeat the same routing scenario, which has already been discussed in previous chapters, in this network.

For convenience, we consider regular-shaped hexagonal networks given in the Figure 4.1, that is, such that the centres of the hexagons on its periphery form an outline of a regular hexagon (our construction can be easily generalised to irregular-shaped hexagonal networks).

Definition 4.1.1. *An infinite hexagonal grid is a graph* $G_H = (V_H, E_H)$ *with a set*

of vertices V_H and a set of edges E_H where $V_H = \{(i,j)|i \in \mathbb{Z}, j \in \mathbb{Z}\}$. The verices (i,j) and (p,q) are connected in a hexagonal grid by an edge if and only if $i = p \pm 1$ and $j = q$ or $i + j$ is even and $i = p$ and $j = q + 1$ or $i + j$ is odd and $i = p$ and $j = q - 1$.

Definition 4.1.2. *A regular hexagonal grid of size t is a fragment of the infinite hexagonal grid consisting of one fixed face of the grid together with all faces which can be reached from it by crossing at most $t - 1$ edges.*

Routing in hexagonal networks has already been studied in the literature. In particular, a regular hexagonal grid has been widely used as a model of network to study routing [15] [52] [60] and to position nodes in applications such as wireless networks [56] and cellular networks [2] [52] [61]. An optimum routing algorithm in both two and three dimensional hexagonal meshes has been discussed in [11] and [20].

A common method is constructing a coordinate system for a routing algorithm to use. Two possible coordinate systems, with two axes [25] and three axes [52], have been used to organise routing in a two dimensional hexagonal grid. In the paper [15] routing happens via shortest paths in hexagonal grid network. One approach focuses on finding an optimal number of steps required for communication and uses three components (x, y, z) for specification of the location of the nodes in [15]. Similarly, the shortest distance between nodes via representative points of the nodes in a three axes coordinate system has been used in [52]. Another way is to use two components (x, y) to specify the location of the nodes; here, the aim is to reduce the total number of bits representing nodes, because each edge connecting the nodes requires one bit

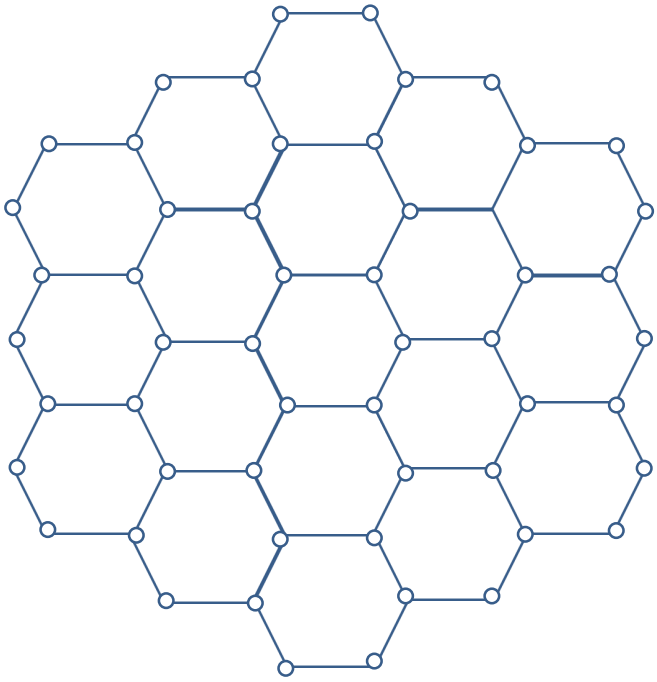


Figure 4.1: A regular hexagonal grid which has a computer in each node.

[3].

A smaller vertex degree of a hexagonal grid has been discussed in [60] and it is shown in [60] that the graph-theoretical properties of a hexagonal grid network allow the model to reduce the cost of routing by comparing other grid models. The Bloom filter has been used for routing in hexagonal grid networks in the literature, like the study [14]. The study [14] describes communication in a hexagonal network via Bloom filters, with the aim of preventing possible attacks.

4.1.1 A Routing Scenario

In this chapter, we consider a way of routing in a hexagonal grid with Bloom filters which eliminates false positives. We assume there is a computer on every node (see

Figure 4.1). In a hexagonal grid a node is connected to at most three edges. As discussed in Chapter 2 and [29] in the case of a rectangular grid, if a particular delivery path has been chosen in advance, to avoid false positives one just needs to ensure that the message is forwarded towards only one edge at each node of the path. As in the other model of networks discussed in other chapters, all messages in this network are sent via shortest paths between the sender and the receiver. We shall assume that once a node receives a message, it does not send it back. Hence, there are two possible edges at each particular node to carry the message further to its destination. We encode all edges of the hexagonal grid in such a way that it is possible to make a clear choice between these two routing possibilities as long as the path to follow is one of shortest paths between the source node and the target node.

We assume that the sender chooses a particular delivery path, encodes it as a Bloom filter, and this Bloom filter is sent along together with the message, like in [13]. For this model to function, a certain encoding of the edges of the grid needs to be introduced. One could represent edges by random Bloom filters, but this approach would generate false positives. Our approach is to allocate Bloom filters to edges according to their positions in the grid.

4.2 Encoding Edges in a Hexagonal Grid

We use the strategy that is behind the coding edges of a rectangular grid studied in Chapter 2 and [29]. In that case, firstly, we will build a convenient coordinate system for encoding the edges in a hexagonal grid. Unlike a rectangular grid, where

two axes are needed, three axes x, y, z will be associated with a hexagonal grid, since the edges in a hexagonal grid can be orientated in one of three different directions and all edges in the network must be encoded by a reasonably small length of Bloom filter. The three possible orientations of an edge x, y, z can be conveniently described as pointing towards 3 o'clock (or 9 o'clock), towards 1 o'clock (or 7 o'clock), and towards 5 o'clock (or 11 o'clock).

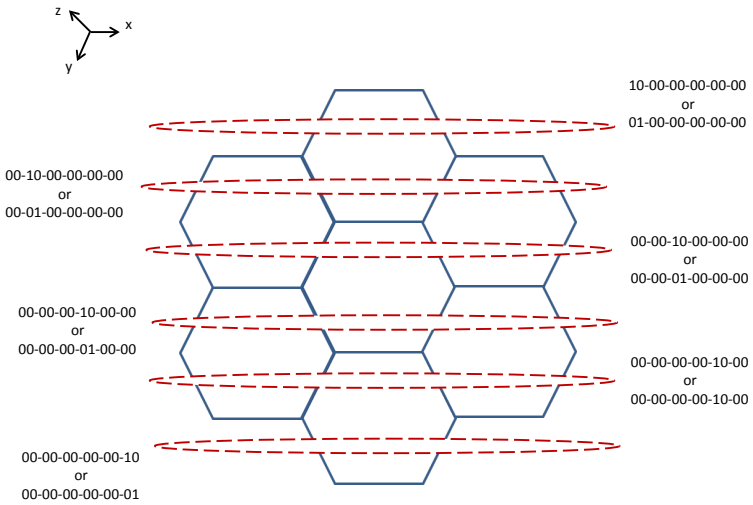


Figure 4.2: Given codes represent the first projections of the Bloom filters of edges and the codes belong to the edges which are intersected by horizontal rows being parallel to the axis-x in a hexagonal grid of size 2.

We introduce the size of a network as the number of edges crossed if an imaginary observer travels from the geometric centre of the network to its edge. One hexagon constitutes a hexagonal grid of size 1. A hexagonal grid of size 2 (shown on the diagrams in this section) is produced by adding hexagons around the boundary of the hexagonal grid of size 1, and so on. Hence, a hexagonal grid of size $t \geq 2$ is obtained by adding $6(t-1)$ unit hexagons around the boundary of the hexagonal grid of size

$t - 1$.

We split the set of all edges of a hexagonal grid of size t into $(4t - 2)$ rows in each of the three directions of axes, see Figures 4.2, 4.3 and 4.4. It is useful to think of rows as numbered from 1 to $(4t - 2)$ in each of the three directions.

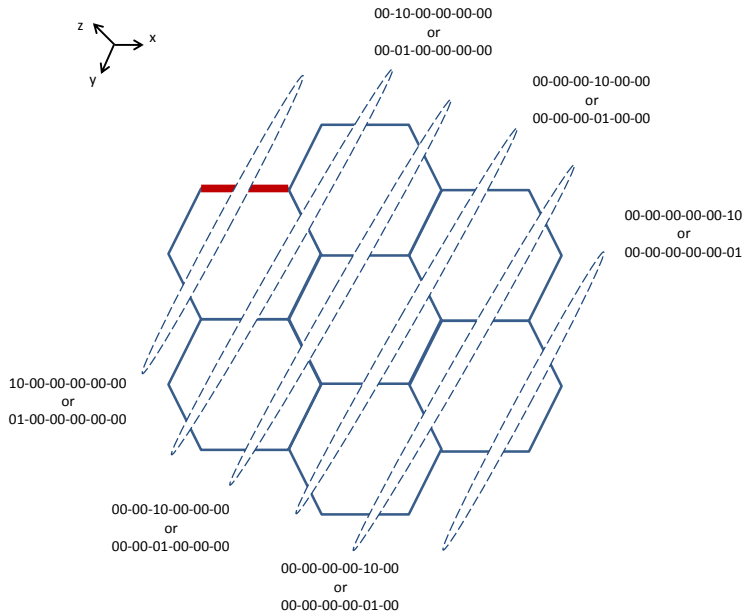


Figure 4.3: The codes represent the second projections of the Bloom filters of edges and these codes belong to the edges which are intersected by the rows being parallel to the axis-y in a hexagonal grid of size 2.

The Bloom filter of an edge consists of three parts, which we call projections. Each projection consists of $(4t - 2)$ two-bit length fragments, which we call blocks. The first, second and third projections reflect the edge's position in the rows parallel to the axes x, y and z, respectively. One projection (corresponding to the rows parallel to the edge) contains only 0s, whereas in the other two projections there is exactly one block which is either 10 or 01, all the other bits being 0. The position of the non-zero block within a projection reflects the number of the row by which the edge

is intersected. The choice between 10 and 01 is made according to the orientation of the edge (see Figures 4.2, 4.3 and 4.4). The Bloom filters of edges, that are the edges parallel to the axis y, include the block 10, similarly the Bloom filters of edges, that are the edges parallel to the axis z, contain the block 01. The blocks either 01 or 10 appears in the Bloom filters of edges, that are the edges parallel to the axis x, corresponding to the way of other edges intersected by the same row. Any row intersects the edges that are parallel to two different ways. Therefore, when a row intersects the edges being parallel to both the axes x and y, then the Bloom filter of x-coordinate parallel edges contain the block 01, otherwise it contains the block 10.

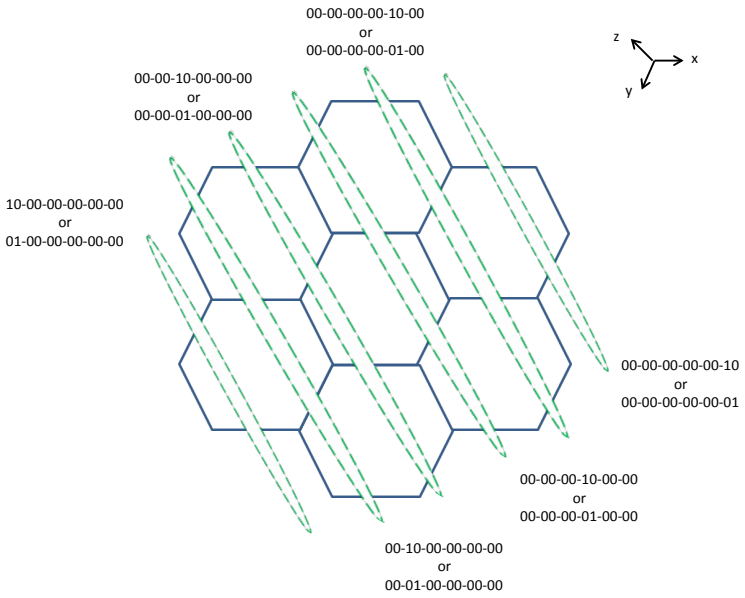


Figure 4.4: Given codes represent the third projections of the Bloom filters of edges and they belong to the edges which lie on the rows being parallel to the axis-z in a hexagonal grid of size 2.

Therefore, the length of Bloom filter of an edge is $3 \times 2 \times (4t - 2)$ where t is the size of the hexagonal grid.

As an example with small parameters, when an edge in a hexagonal grid of size 2 lies on the first row being parallel to axes y and the edge is parallel to the axis x (see the red edge in the Figure 4.3), then the first, second and third projections of Bloom filter of this edge look like 000000000000, 100000000000 and 000001000000, respectively. As seen, the first projection of the Bloom filter of this edge do not contain a block including the bit 1, since this edge is not intersected by a row being parallel to the axis-x. The Bloom filters of edges includes two bit 1s, since every edge is intersected by two different rows.

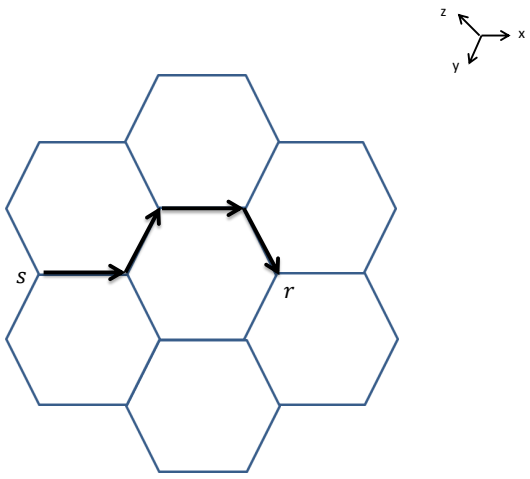


Figure 4.5: A message is directed in a shortest path of hexagonal grid of size 2.

Consider a hexagonal grid of size 2 given in the Figure 4.5, a message is sent from the node s to node r through a four edges length shortest path. The path has been chosen in advance by the server, so the message definitely follows this route. We apply the coding method described above to the edges, then we obtain the Bloom filter of the path by applying OR-operation to the Bloom filters of the edges of the

path. In brief, the Bloom filter of the path where the path is given in the Figure 4.5 is 000011000000 − 001010010000 − 000110010000. By comparing the Bloom filter of an edge and the Bloom filter of the path, it is easy to check whether an edge is on the way of the message or not. Recall that the purpose is to avoid the message follow any route except chosen one.

4.3 Shortest Paths in Hexagonal Grids

In our message delivery model, a message follows a shortest path between the sender and the receiver, and the Bloom filter is used to deliver the message represents this path as an undirected path. However, below in our study of shortest paths in a hexagonal grid, it is convenient to treat paths as directed paths. Accordingly, there are six possible orientations of a directed edge, which we can denote by C_1, C_3, C_5, C_7, C_9 and C_{11}, where the index corresponds to the hour on the clock face, see Figure 4.6.

Lemma 4.3.1. *If a directed path is a shortest path between two distinct vertices in a hexagonal grid then the path does not contain two edges directed in the opposite directions.*

Proof. Suppose a path P contains two edges directed in the opposite directions. Say, P contains an edge with orientation C_7 and then an edge with orientation C_1. Note that there may be several edges with such orientations in the path. Locate a minimal fragment F of the path with this property; that is, this fragment starts with C_7, finishes with C_1 and does not contain neither C_7 nor C_1 in the middle (see Figure 4.6). Such a fragment from vertex u to vertex v is shown in the Figure 4.6. All edges

of F (except the first and the last one) have only one of two orientations, alternatingly, say, C_5 and C_3, as shown in the Figure 4.6. It is easy to see that there is another path F' from u to v which is two edges shorter than F; indeed, this is the one which only uses edges with orientations C_3 and C_5, alternatingly. If one replaces the fragment F in the path P by F', one obtains another path P', which is two edges shorter than P.

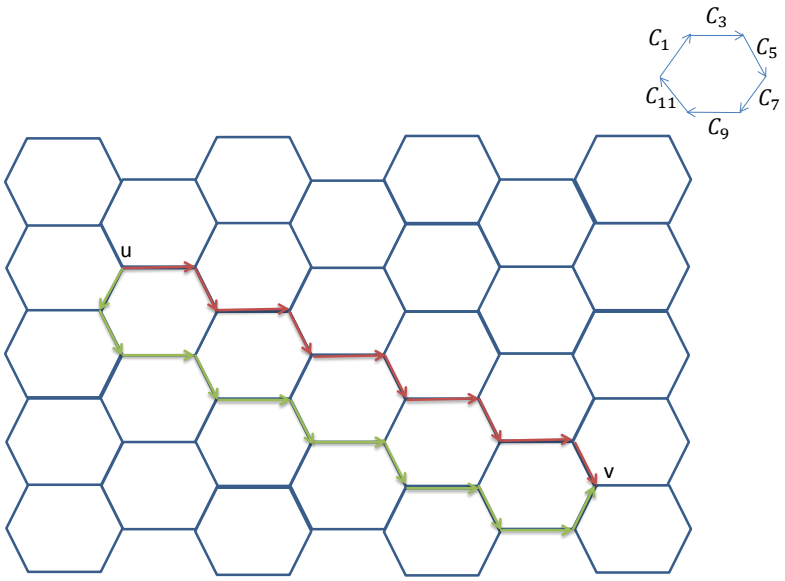


Figure 4.6: The paths $P^{'}$ (red path) and P (green path) are the paths between the vertices u and v.

□

4.4 Bloom Filters of Shortest Paths

In this section of this chapter, we show that the encoding method applied to the edges produces some special Bloom filters for the shortest paths. Recall that the Bloom filter of a shortest path is found by applying bitwise OR operation to the Bloom filters

of the edges on the path.

Lemma 4.4.1. *The Bloom filter of a shortest path may contain a block* 11 *in at most one projection.*

Proof. A block in a projection is 11 if and only if two non-parallel edges of the path lie on the same row of this projection. Let us proceed by contradiction. Suppose two projections contain a block 11. For the sake of an example, say, in the projection parallel to x we have two edges parallel to y and z in the same x-row, and in the projection parallel to y we have two edges parallel to x and z in the same y-row. Then the proof splits into several similar cases, of which we shall present one, for brevity. Suppose that the path has a fragment which starts in the south-east quadrant (that is, to the east of the y-row and to the south of the x-row), then crosses to the south-west quadrant, then to the north-west quadrant, and then back to the south-east quadrant (note that all quadrants must be visited by the path which contains four edges as described above). One of possible sub-cases of this case is that in the projection parallel to y we have two edges with orientations C_{11} and C_3 in the same row, and in the projection parallel to x we have two edges with orientations C_1 and C_5 in the same row (This case is illustrated in the Figure 4.8. The edge e oriented to C_{11} lies on a row with the edges oriented to C_3 in the path between the vertices u and w, and the edge f oriented to C_5 lies on a row with at least an edge oriented to C_1 in path between the vertices u and w). Then two edges have opposite orientations C_{11} and C_5, which is impossible, by Lemma 4.3.1. Considering each case and sub-case, we come to the same contradiction.

□

As a result of Lemma 4.4.1, the first projection includes the block 11 if all edges of the path belong to the cone of directions (C_1, C_3, C_5) or equivalently (C_7, C_9, C_{11}), since the edges on the directions of C_1 and C_5, or C_7 and C_{11} lie on the same row and they are represented by different types of blocks. Similarly, second and third projections contain the block 11, when the edges belong the cone of directions (C_1, C_3, C_{11}) or (C_9, C_7, C_5), and (C_1, C_9, C_{11}) or (C_7, C_3, C_5), respectively.

4.5 Without False Positives

As discussed, false positives of the Bloom filter of the delivery path will disrupt delivery if and only if they are edges adjacent to the path; this is an approach used in [29].

Theorem 4.5.1. *If the Bloom filter of a shortest path in a hexagonal grid is calculated on the basis of the edge encoding introduced in Section 4.2, it does not yield any false positives which are edges adjacent to the path.*

Proof. There are three different links connected to a node, and a message can be forwarded to any one of them. We assume that once a message arrives at a node, then it is not sent back. Then there will be 2 possible edges where to forward the message. When it is shown that only one of these edges link with the path and other edge is adjacent with the path for all possible scenarios, then it will be proved that any shortest path does not yield false positives.

We shall assume that a message arrives to a node via an edge e with orientation C_3, and next it should be forwarded either to direction C_1 or direction C_5, see Figure 4.7; all other cases are similar to this one and can be obtained by rotation.

The next two edges have the clock directions C_5 and C_1 and lie on the same kind of row that encodes the first projections of next two edges (see Figure 4.7 , the rows are drawn as horizontally). So the represented blocks of the following two edges coincide in the first projection of the Bloom filter, but on different block position. These two edges do not lie on the same row. The rows, which encode the first projections of the Bloom filters of the edges, passing through these two edges are consecutive, so they differ in 1 row position. Moreover, one of the next edges is parallel to the axis y whose represented block is 10 and other edge is parallel to the axis z whose represented block is 10. As a result, the first projection of the Bloom filter of the path is distinguishable with regard to any edge coming after the edge being parallel to the axis x.

However, there can be at least one edge called g on the path that might lie on the same row encoding the first projection and lie on the same direction C_1 with the adjacent edge f. So the adjacent edge f and the edge g on the path have the same encoding block 10 on the same block position of the same projection, but they differ with the block position of the third projections of the Bloom filters. Since they can not lie on the same row encoding the third projection of the Bloom filter of the edge (the rows are parallel to the axis-z in the Figure 4.7).

On the other hand, there might be another edge called h on the path lie on the same row, coding the third projections of the Bloom filters of the edges, with the adjacent edge f. Yet the Bloom filter of the edge f contain the block 10 in the third

projection and the Bloom filter of the edge h include the block 01 on the same block position with the block 10 of the edge f in the third projection. These two blocks appear in the same block position in the same projection, but the bit 1 is placed in different position. The Bloom filter of the path do not contain the block 11 in third projection by Lemma 4.4.1, since the edges of the path belong to the cone of directions (C_1, C_3, C_5). So the Bloom filter of the path contain the block 11 in only its first projection. Because of that the path cannot contain the edges f and h at the same time.

Additionally, there cannot be any edge apart from the edge h that lies on the same row, encoding the third projection, with the edge f. Otherwise, the path includes the edges directed opposite ways, so this contradicts the Lemma 4.3.1.

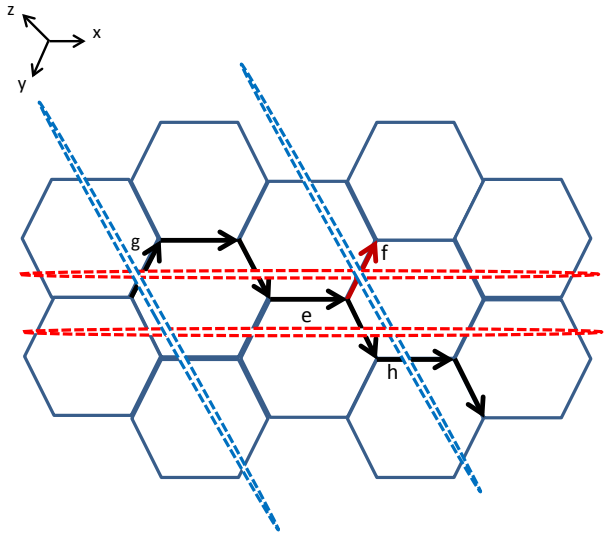


Figure 4.7: The rows encoding the first and third projections of the edge f and a possible path including the edges e, g, h.

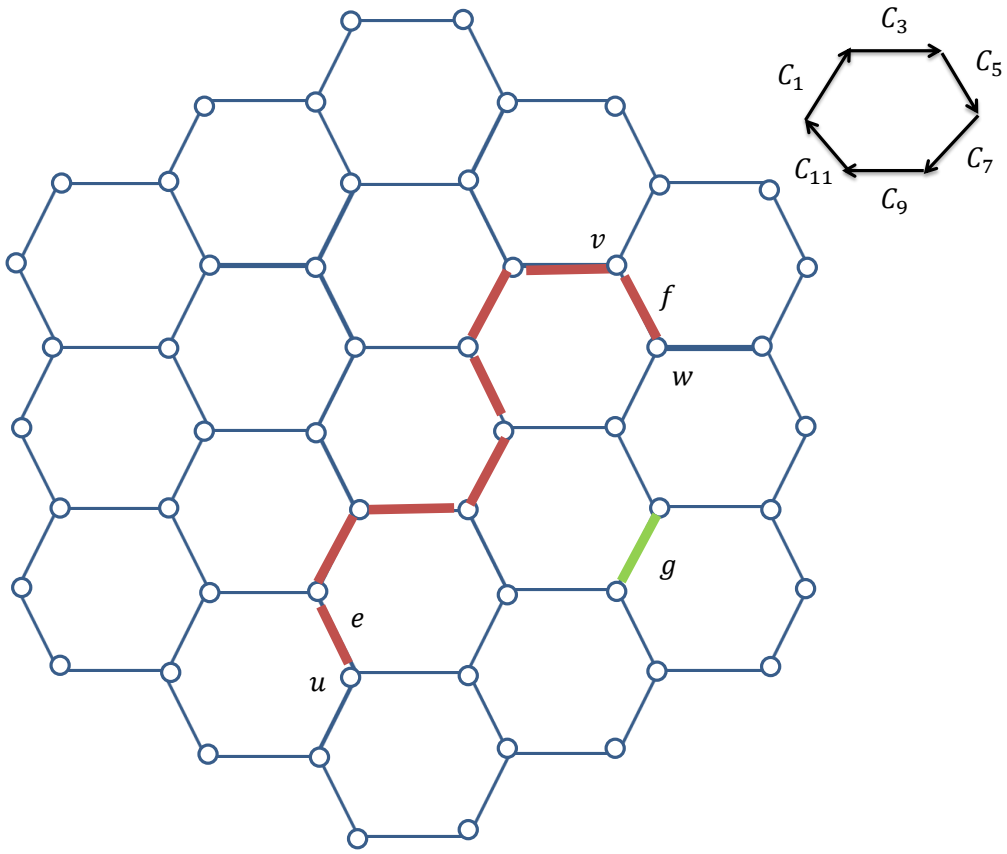


Figure 4.8: A path containing two edges having two opposite directions between the vertices u and w is not a shortest path. Another path between u and v is one of the shortest path in the hexagonal grid and the edge g is not an adjacent edge to this shortest path.

As a conclusion, there cannot be any edges recognised by the Bloom filter of the path that have the same Bloom filter with the edge f. Hence, the Bloom filter of the path including the edges e, g and h is greater than or equal to the Bloom filter of the edge f in all bits positions. Therefore, the edge f is definitely an adjacent edge of the path P and it is not a false positive for the Bloom filter of the path.

By following the same process, it should be shown that none of the adjacent edges being parallel to the axes x and y, followed by the edge being parallel to the axis z is false positive of the Bloom filter of the path. □

The encoding method that we introduced for the shortest paths in hexagonal grids does not produce false positives which are adjacent to corresponding shortest paths. Yet, there might be some edges which are not adjacent to a shortest path but their Bloom filters might be less than or equal to the Bloom filters of the shortest path in all bits positions. For example; the edge g given in the Figure 4.8 is not adjacent to the shortest path between the vertices u and v, but its Bloom filter is less than or equal to the Bloom filter of that shortest path. Hence, the edge g might be thought of as a false positive of the Bloom filter of the shortest path between u and v. This circumstance does not affect the message delivery in our routing scenario.

4.6 Comparing Arbitrary Encoding Models with our Approach in Hexagonal Grid

To see the advantage of our approach, we consider two possible models constructed by using random methods that we compare with our model. One model uses one bit per edge labelling, and in the other one the bits 1 are placed into random positions in Bloom filter.

If every edge was assigned to one bit (thus, never creating any false positives) then the total number of the bits in a Bloom filter of an edge would be $9t^2 - 3t$ which is the number of edges in a hexagonal grid of size t. This formula has been given by [60, Lemma 2] . But the length of the Bloom filters of the edges in our model is $24t - 12$. When t becomes large, $9t^2 - 3t$ increases faster than $24t - 12$. So for reasonably big

sizes of hexagonal grids, the length of Bloom filter of our model yields better results which is $24t - 12 < 9t^2 - 3t$, equivalently $t \geq 3$.

We construct Bloom filters for edges in a hexagonal grid by putting the bit 1 in the certain places. However, the standard Bloom filter is built using random allocation of 1s [7]. Because of the randomness, the standard Bloom filter is likely yield false positives. The theory of Bloom filters says that the probability of false positives depends on the length of the Bloom filter m, number of bits 1s k in the Bloom filter and the number of elements n in the set.

Now assume that random Bloom filters of the same length as in this chapter are used.

It is given in [60, Lemma 5], with a proof that the maximum number of edges between two nodes on a shortest path is $4t - 1$ where t is the size of the hexagonal grid. Hence, the number of edges in a shortest path is $\leq (4t - 1)$.

To minimise the probability of false positives, the optimum value of number of 1s in Bloom filters of elements is expressed by the formula $k = \lceil \ln 2 \times \frac{m}{n} \rceil$, where m is the length of the Bloom filter and n is the number of elements in the set [22]. So, the optimal number of 1s in our model would have been $k = \lceil \ln 2 \times \frac{24t-12}{4t-1} \rceil$, that is, $k \approx \lceil \ln 2 \times 6 \rceil = 4$ for reasonably large t. Yet, in our model, the number of 1s in a Bloom filter is fixed with integer 2.

Therefore, if we would have built the Bloom filters by putting the bit 1 in arbitrary places, then the maximum value of the probability of false positives of a shortest path is found by following equation with these values.

$$\left(1 - e^{\frac{-kn}{m}}\right)^k = \left(1 - e^{\frac{-4(4t-1)}{24t-12}}\right)^4 \approx 0.05 \tag{4.1}$$

Hence if the Bloom filters were obtained by placing the bit 1 randomly, then the probability of false positives would be approximately 0.05, when the length of the Bloom filter and the number of edges were chosen in their maximum values. The number of adjacent edges of a path in a hexagonal grid is $n-1$ where n is the number of edges on the path, since once a message is received by a node then there are two options for the message to follow. These options are the edge on the path and adjacent edge of the path. Therefore, the probability of having no false positives in a path would be approximately $0.95^{(n-1)}$.

In real applications of the Bloom filter, both the length of the Bloom filter m and the size of hexagonal grid t might be too large. Yet, our model can be modified to irregular hexagonal grid, so the length of the Bloom filter can be chosen under consideration of the width of the space. In applications of the Bloom filter, the locations of the bit 1 depend on the hash function, which builds the Bloom filter of the set as well as should be constructed in a various way. Hence, false positives will probably be produced because of the randomly placed bit 1. We build a system which tightly controls the locations of the bit 1 in the Bloom filter. Therefore, it is proved that we do not obtain false positives. A simulation of our method is out of scope neither this chapter nor this thesis, since the edge labels differ between the standard Bloom filter and our model.

CHAPTER 5

TRIANGULAR GRIDS

5.1 Introduction

The encoding method we studied in this chapter has been published as a paper [31].

We consider another graph given in the Figure 5.1, which is called triangular grid, as a model of a computer network. We shall assume that triangular grids have an overall shape as a regular hexagon, as shown in the Figure 5.1. However, the results established for regular triangular grids can be easily adapted to irregular triangular grids.

Definition 5.1.1. *An infinite triangular grid is a graph* $G_T = (V_T, E_T)$ *with a set of vertices* V_T *and a set of edges* E_T *where* $V_T = \{(i,j)|i \in \mathbb{Z}, j \in \mathbb{Z}\}$. *The verices* (i,j) *and* (p,q) *are connected by an edge in a triangular grid if and only if* $i = p$ *and* $j = q \pm 1$ *or* $i = p \pm 1$ *and* $j = q$ *or* $i = p-1$ *and* $j = q-1$ *or* $i = p+1$ *and* $j = q+1$.

Definition 5.1.2. *A regular triangular grid of size* s *is a fragment of the infinite triangular grid consisting of all vertices which are at the distance of at most* s *from a*

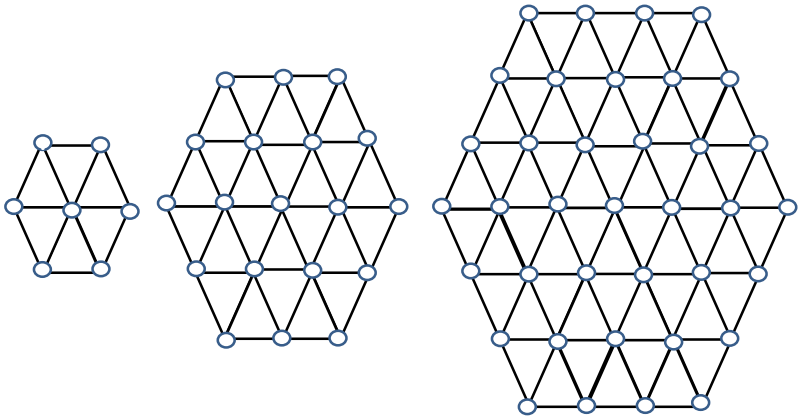


Figure 5.1: Regular triangular grids with 1 edge, 2 edges and 3 edges on a side.

certain fixed vertex.

In order to represent the cells in a hexagonal and a triangular grid, the three coordinate system has also been used by [49] [50]. We shall also use three coordinates in this chapter; previously we used a similar three-axes coordinate system in our study of hexagonal grids Chapter 4 and [30] and a two-axes coordinate system in our study of square grids Chapter 2 and [29].

A triangular grid has been used as a network model in some applications. For example, to reduce the power consumption during the distribution in triangular grids has been proposed in [33]. The method in [33] decreases the number of packets by combining the codes of packets. It has been noted that triangular grid is a convenient model for wireless sensor networks [33] [69].

5.1.1 A Routing Scenario

In this model of network, as usual each node represents a computer, and there are up to 6 links incidental to each computer. We assume that messages from one node to another are always sent along a shortest path between the two nodes. Also we assume that the message and an encoded description of the delivery path are sent together. This is the method described in [13]. Our way of labelling edges is to use Bloom filters [7].

Once two computers are marked as one sender and one receiver, the sender computer chooses one of the shortest paths in advance. We assume when a computer on the delivery path receives the message, it does not send it back. We encode all edges and the Bloom filter of a shortest path is obtained by applying bitwise OR operation to all edges in the shortest path. The Bloom filter of the shortest path and the message are sent together by the sender computer throughout the path. In this case, the edges are queried whether on the chosen shortest path or not. Hence, the adjacent edges to the shortest paths are possible false positives. As we examined in previous chapters, we aim to avoid the false positives to reduce the message traffic.

5.2 Encoding Edges in Triangular Grids

A triangular grid of size 1 with 1 edge at a side has an overall shape as a hexagonal grid (see the first graph on the left hand side in the Figure 5.1). The triangular grid of size s where $s \geq 2$ is obtained by adding unit triangles around the boundary of

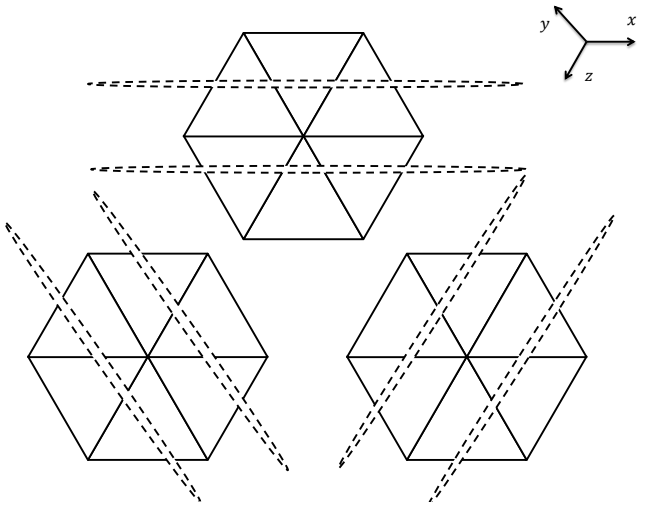


Figure 5.2: Imaginary encoding rows intersect the edges on the direction of corresponding axis.

triangular grid of size $s-1$ until the overall shape has a form of a regular hexagon with s edges at a side.

Similarly to hexagonal grids in Chapter 4 and [30], the edges in a triangular grid could be encoded by recording their intersections with imaginary rows parallel to three axes, see Figure 5.2.

All rows in each family of parallel rows are assumed to be numbered consecutively from 1 to $2s$, where s is the number of edges on any one side of triangular grid (refer to Figure 5.1).

The Bloom filter of each edge consists of three parts which we call projections and includes exactly two bits 1, the others being 0. Each of the three projections correspond to one family of parallel rows and consists of $2s$ two-bit fragments called blocks. A two-bit block includes a bit 1 if it corresponds to a row which intersects the edge which we are encoding. The number of rows being parallel to the axes x, y

and z specifies the places for 2-bit block including the bit 1 in first, second and third projections of the Bloom filter of an edge, respectively.

Accordingly, the length of the Bloom filter is $3 \times 2 \times 2s = 12s$, where $2s$ is the number of blocks in a projection, each block contains 2 bits, and the number of projections is 3.

As we have said, a block corresponding to a row intersecting the edge is either 01 or 10. We say that it is 01 (or 10) if the direction of the edge is produced by a rotation of the direction of the row clockwise (anticlockwise) by 60 degrees. In other words, the Bloom filter of an edge includes the block 10 in its first and third projections, when the edge is parallel to the axis y. If the edge is parallel to the axis z, then the Bloom filter of the corresponding edge contain the block 01 in its first and second projections. Lastly, if the edge is parallel to the axis x, then the choice of the block depends on the orientation of the other edges which are intersected by the same row with the corresponding edge being parallel to the axis-x. Namely, if other edges lying on the same row with the edges being parallel to the axis x are parallel to axis y, then the edge is encoded by the block 01. Similarly, if other edges lying on the same row with the edges being parallel to the axis x are parallel to axis z, then the edge contain the block 10. As a result of this property, the Bloom filter of all edges that are the edges parallel to the axis-x contains the block 01 in its third projection and 10 in its second projection.

5.3 Shortest Paths in a Triangular Grid

We assume that messages are always routed along a shortest path between the sender node and the receiver node. A triangular grid is similar to a hexagonal grid [49], therefore like in hexagonal grid, the shortest distances between nodes have been found by using three-axes coordinate system [51]. Note that although our graph is undirected, when we study paths, it is convenient to think of a path as directed. Like the orientations of edges in a hexagonal grid Chapter 4 and [30], it is convenient to denote orientations of directed edges in a triangular grid using the hour numbers on a clock face. Hence, any directed edge has one of six possible orientations $C_1, C_3, C_5, C_7, C_9, C_{11}$, C standing for direction on the 'clock' face.

Lemma 5.3.1. *If a path is shortest in a triangular grid then the edges of the path have at most two adjacent directions. The adjacent directions have consecutive indices, e.g.* C_1 *and* C_3 *are adjacent directions.*

Proof. Suppose a path P includes two non-consecutive directions, say C_3 and C_{11}. Note that there may be several edges with such orientations in the path P. We locate a fragment F of the path P with the property that is F starts with C_3 and finishes with C_{11} and does not include neither C_3 nor C_{11} in the middle. All edges between these two edges have a direction of C_1 in the fragment F of the path P (see Figure 5.3). It is obviously seen in the Figure 5.3 that there is another path F' from u to v which is one edge shorter than the path F. When we replace F in the path P by F', then other path P', which is one edge shorter than the path P, is obtained.

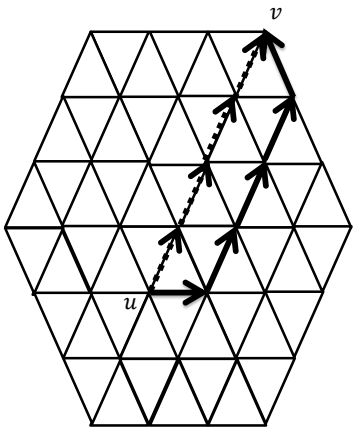


Figure 5.3: The paths between the vertices u and v.

□

Lemma 5.3.2. *A single row cannot intersect more than one edge on the shortest path.*

Proof. Suppose a row intersects more than one edge in a shortest path. At least two edges are intersected by the same row and suppose that they lie on the same shortest path. The edges in a shortest path have at most two adjacent directions by Lemma 5.3.1. Hence, we shall assume that all edges on the shortest path have two adjacent directions C_1 and C_3, see Figure 5.4.

Suppose the two edges e and f have the same direction, say, C_3. The left endpoints of the edges e and f are, say, u and v, respectively. The path containing either of the edges e or f between the vertices u and v is longer than the path which goes straight between the vertices u and v.

Now suppose one edge f has direction C_3 and one edge g has direction C_1 in the

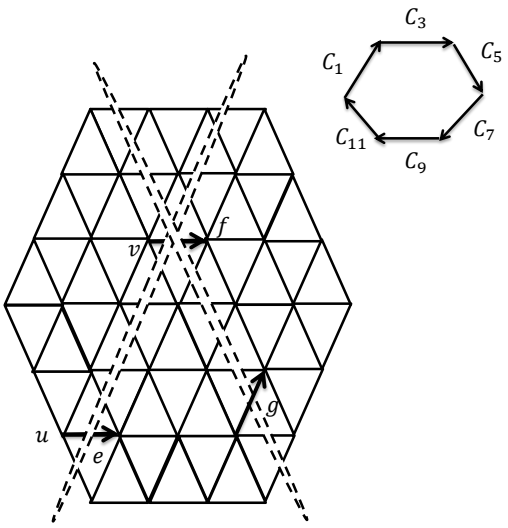


Figure 5.4: The rows, which are used for encoding the edges in a triangular grid, intersect the pair of edges, which are e, f with endpoints of the vertices u and v, and an edge g.

same shortest path, and both edges lie on the same row. Yet, some of the edges between these two edges f and g must have a direction which is neither C_1 nor C_3 (see Figure 5.4). This contradicts Lemma 5.3.1.

□

5.4 Bloom Filters of Shortest Paths

The encoding method that we introduce produces unique Bloom filters for the edges. As a result of this, the Bloom filters of paths have distinguishable forms. The following lemmas show these unique forms.

Lemma 5.4.1. *Blocks of both forms* 10 *and* 01 *can appear in at most one projection of the Bloom filter of a shortest path.*

Proof. If more than one projection of the Bloom filter of a path contains blocks of the form 10 and 01 then we can conclude that the path contains edges parallel to each of the three axes x, y and z. This contradicts the fact that edges in a shortest path are orientated in at most two directions, by Lemma 5.3.1. □

Lemma 5.4.2. *Non-zero blocks (corresponding to edges of the path) form a continuous sequence within each projection of the Bloom filter of the shortest path.*

Proof. Assume that there is a block 00 between non-zero blocks in a projection of Bloom filter of a shortest path. All edges are represented by a non-zero block depending on the orientation of the edges, and these representative blocks of the edges take place on the Bloom filter of the shortest path with respect to the number of rows that intersect the edge on any direction. The rows are numbered consecutively in all directions, separately. Hence, the block 00 between non-zero blocks in a Bloom filter of the shortest path shows that there is a gap between the edges on the shortest path. This contradicts the fact that in a path all edges lie consecutively without a gap. Therefore, the Bloom filter including the block 00 between non-zero blocks does not represent a shortest path.

□

Note that, the Lemma 5.4.2 is true for arbitrary paths. A path is a graph on a sequence of vertices which are connected by a sequence of edges. If there is a block 00 between non-zero blocks in the Bloom filter of a path, this shows that the edges are not connected. This contradicts the definition of the path.

We build the Lemma 5.4.2 for the shortest paths. We are only interested in the

Bloom filters of shortest paths in this thesis, since in the routing scenario that we consider the shortest paths carry the messages.

Lemma 5.4.3. *The Bloom filter of a shortest path does not contain the block* 11 *in any projection.*

Proof. The block 11 appears in the Bloom filter of a path when blocks 10 and 01 of at least two edges of the path occupy the same block position. The block positions correspond to rows. By Lemma 5.3.2, no two edges in a shortest path lie on the same row. Therefore, it is impossible that the Bloom filter of a shortest path contains the block 11 in any projection. □

5.5 Routing Without False Positives

Theorem 5.5.1. *When a shortest path in a triangular grid is encoded, the encoding does not produce any false positives adjacent to the path.*

Proof. A node in a triangular grid has up to six edges incidental with it. Usually there are five possible edges to forward a message from a node, since the message has arrived to the node via an edge, and we are not sending the message back.

Based on Figure 5.5, suppose a shortest path contains an edge e. A message comes from the edge e, then it must follow the edge which is on the shortest path containing edge e. Yet the message may follow all possible next edges, denoted by f, g, i, j and k. These adjacent edges are possible false positives of the shortest path. We examine if any of these next edges is a false positive or not.

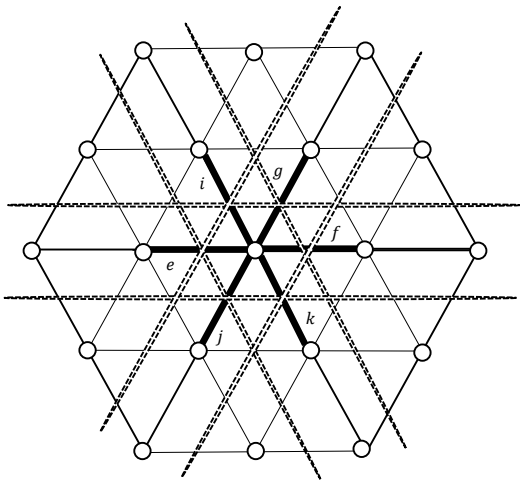


Figure 5.5: The rows intersecting the edges e, f, g, i, j, k.

The row encoding the second (third) projection of the edge e intersects also the edge j (i). Yet, by Lemma 5.3.2, the edges on a shortest path are not intersected by the same row. Hence, since the edge e on the path, the edges j and i cannot be on the same shortest path, and they may be false positives. However, by Lemma 5.4.3, the Bloom filter of the path does not contain the block 11 on the block position of corresponding rows which encodes the second and third projections of the edge e. Accordingly, the edges j and i are not false positives of the Bloom filter of the shortest path.

The next edges k, f and g do not lie on the same rows with the edge e. Hence, one of them may be an edge coming after the edge e on a shortest path. Suppose the next edge is g. One of the encoding rows of the edge g intersects the edge f, hence by Lemma 5.4.3, the edge f is not a false positive of the Bloom filter of the path when the edge g is assumed to be an edge on the shortest path.

Assuming that the edges e and g lie on the same shortest path, the path cannot

contain the edges having any directions except existing ones by Lemma 5.3.1. The edge k has direction of C_5. This is different from the directions C_1 and C_3 of the edges e and g on the shortest path. Hence, k may be a false positive. Since, the edge g is parallel to the axis-z and the edge k is parallel to the axis-y, the representative blocks of the edges g and k are 01 and 10, respectively. Besides, all the edges in the shortest path including the edge g are parallel to the axis-x (clock directions of C_3 and C_9) and axis-z (clock directions of C_1 and C_7) by Lemma 5.3.1. Therefore, if the shortest path including the edges e and g contains also the edge k, then as a representative block 10 appears together with the block 01 in more than one projections (first and third) of the Bloom filter of the shortest path. This contradicts Lemma 5.4.1. Accordingly, while a shortest path contain the edges e and g, then the edge k cannot be on the same path. That is why k is not a false positive of the Bloom filter of the shortest path.

Similarly, we suppose that the next edge of the edge e is k. Because of the orientations of the edges g and k in a shortest path, the block 10 and 01 appear in the first and the third projection of the Bloom filter of the path. This contradicts Lemma 5.4.1. Therefore, while e and k on the same shortest path, then the edge g is not on the shortest path and not a false positive. The edges k and f lie on the same single row. Therefore, by Lemma 5.4.3, the corresponding projection of the common row of the edges f and k contains the block 11. Thus, the edge f is neither an edge of the path nor a false positive, while both e and k lie on the same shortest path.

Finally, we assume that the edge f is on the shortest path. The adjacent edges g and k are intersected by the rows which intersect the edge f. Hence, the Bloom filter

of the path contains the block 11, yet this contradicts the Lemma 5.4.3. Accordingly, when the edges e and f are on the path, then the edges g and k are not false positives of the Bloom filter of the shortest path. They are only adjacent edges.

Another possibility is that the edge e is the first edge of the shortest path; in this case, we would want to see that none of the other five edges is a false positive. If the previous edges lie on the same row with the edge e, then they are obviously not on the same shortest path by Lemma 5.3.1 and Lemma 5.4.3. Suppose that the previous edges of e are not intersected by the same rows with the edge e. Yet, they lie on some rows that are parallel to some of the rows intersecting the edge e. The edge e is the first edge of the path, then the non-zero blocks of the edge e are either the first or last blocks in the corresponding projections of the Bloom filter of the shortest path by Lemma 5.4.2. In other words, there are only blocks of the form 00 before or after the block corresponding to the edge e in each projection of the Bloom filter of the shortest path by Lemma 5.4.2. □

We proved that the adjacent edges to a shortest path in a triangular grid are not false positives of the Bloom filter of that shortest path. Yet, there can be found some edges which are not adjacent to a certain shortest path, they might be thought of as an edge of this shortest path by comparing the Bloom filters of these edges and the Bloom filter of the shortest path. For instance: the edge e in the Figure 5.6 is not adjacent to the shortest path between the vertices u and v, yet the Bloom filter of the edge e is less than or equal to the Bloom filter of this shortest path in all bits positions. In this case the edge e might be called false positive, yet in our routing

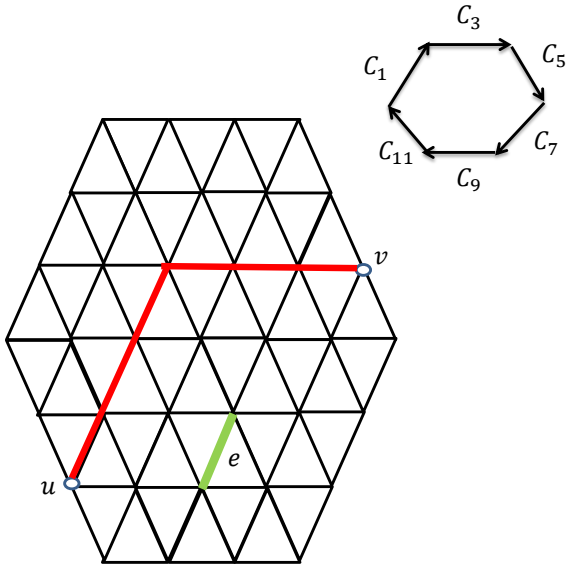


Figure 5.6: There may be some edges whose Bloom filters are less than or equal to the Bloom filter of a shortest path in a triangular grid.

scenario the message follows only specific routes therefore the edge e does not cause an additional message traffic.

5.6 Comparing Our Approach with Arbitrary Encoding Methods

We build the Bloom filters for edges in triangular grids by putting the bits 1 in certain places. Now, we compare our model with random methods to see the advantages of our encoding method. When the bits of Bloom filters encoding edges in a triangular grid are designated randomly, the probability of false positives is not zero. In this section, we examine the probability of false positives with our parameters on random methods.

One random method is the encoding per edge with one bit. In this model, the edges on the shortest path are represented by the bit 1, all other bits are 0 in the Boolean array of a shortest path. By Theorem 1.3.1 this edge labelling method does not produce any false positives. If the Bloom filter of edges in triangular grid would have been built by labelling each edge with one bit, then the length of Bloom filter in a triangular grid would be the number of edges. The number of edges in a triangular grid with s edges along each side is $3[2(s + (s + 1) + (s + 2) + \cdots + (2s - 1)) + 2s] = 9s^2 + 3s$. Since one bit would have been designated for each edge. It is obvious that $12s < 9s^2 + 3s$ when $s \geq 2$, where $12s$ is the length of Bloom filter of our model. For big size of triangular grid networks, our approach produces reasonably small length of Bloom filters.

Now, we compare our encoding method with the random Bloom filters build with the our parameters that are the length of Bloom filter $m = 12s$ where s is the number of edges on a side and the maximum number of edges n in a shortest path.

Lemma 5.6.1. *The maximum number of edges in a shortest path between two distinct vertices in a regular triangular grid is* $2s$ *where* s *is the number of edges on a side.*

Proof. It is easy to prove this by induction. It is true for $s = 1$. When the number of edges on 1-side triangular grid is 1, then the maximum number of edges on a shortest path is 2 (please consult the graph where $s = 1$ in the Figure 5.1).

For each pair of vertices u and v in the triangular grid of size t, there is n paths between two vertices u and v of length $\leq 2t$. Suppose it is true that the maximum number of edges on a shortest path is $2t$ between some vertices x and y, while the

number of edges on a side is t. By inductive step, we will show that the maximum number of edges in a shortest path is $2(t+1)$ where the number of edges on a side is $t+1$. A triangular grid with $t+1$ edges on a side is obtained by adding one edge to all sides of a triangular grid containing t edges on a side. Hence, it is obtained that 2 additional edges connected to the vertices x and y, which are the two end vertices of a shortest path with maximum number of edges in a triangular grid of size t, are added to the shortest path in a triangular grid of size $t+1$. It is accepted that there are $2t$ edges on a path in the triangular grid of t edges on a side. Accordingly, the maximum number of edges on a shortest path in a triangular grid with $t+1$ edges on one side is $2t+2$. □

The formula given for the optimal number of bits 1 to minimise the probability of false positives is $k = \lceil \ln 2 \times \frac{m}{n} \rceil$, where m is the length of the Bloom filter and n is the number of edges in a path; see, for example, [46]. Hence, the optimal number of 1s in a Bloom filter would have been $k = 4$ with the length of Bloom filter assigned by $m = 12s$, in order to compare the random Bloom filter approach with our approach described in previous sections, and the maximum number of edges in a shortest path $n = 2s$, as per Lemma 5.6.1.

When we assume that random Bloom filter has been built with these parameters, then the probability of false positives would be:

$$\left(1 - e^{\frac{-kn}{m}}\right)^k = \left(1 - e^{\frac{-4(2s)}{12s}}\right)^4 \approx 0.05 \tag{5.1}$$

The number of adjacent edges to the path, which consists of maximum number

of edges, is $\leq 4 \times (2s)$, since there are ≤ 4 adjacent edges to every edge on the path, and $2s$ is the maximum number of edges on a shortest path. Note that, the adjacent edges of the path are the possible false positives of the Bloom filter. Therefore, the probability of having no false positives a Bloom filter built randomly in a path would be approximately 0.95^{8s}.

The approximation (5.1) is similar with the approximation (4.1) showing the possible false positives adjacent to the shortest paths in a hexagonal grid when the random methods are used, see Chapter 4 and [30]. Yet, the probability of the nonexistence of false positives adjacent to the shortest paths differs between these two graphs when they have same size. Since, when a node on a shortest path links with 4 adjacent edges in a triangular grid, a node on the shortest path in a hexagonal grid links with 1 adjacent edge to the shortest path.

CHAPTER 6

TREE-DENSE GRAPHS

6.1 Introduction

The results obtained from this chapter are submitted to a journal as a paper [32].

In this chapter, we examine more than one network model that the combination of these networks produces another network.

The graph in the Figure 6.1 with a dense core (circled) and a tree-like periphery can be decomposed into the core and the periphery. We call this network *tree-dense graph*. This network might be thought of a realistic model for real world networks. We will show in the section 6.2 that a tree-dense graph might be decomposed into some parts which are an almost complete graph, and star graphs which are a particular tree graphs. The core graph is an almost complete graph in our model of network in the Figure 6.1. Note that, in some cases a core graph might be represented by another graphs instead of an almost complete graph (for details see the section 6.2).

Definition 6.1.1. *Bit-per-vertex labelling: Suppose* $G_D = (V_D, E_D)$ *is a graph with*

a set of vertices V_D and a set of edges E_D, and U is a universal set of Bloom filters, then $|E_D|$ is relatively large where $V_D \leq E_D$. Consider a labelling such that $U = V_D$, and for each $e \in E_D$, $\beta(e) = \{u, v\}$, where u and v are the endpoints of the edge e and $\beta(e)$ is the Bloom filter of the edge e. This labelling method is called the bit-per-vertex labelling.

By "almost complete graphs" we mean graphs with many edges relatively to the number of vertices. We formally only require that $E \geq V$ where E denotes the number of edges and V denotes the number of vertices in the graph, but think of the number of edges as much higher than the number of nodes.

By "dense graph" we mean graphs for which bit per vertex labelling is the best way to represent the edges by comparison of the other labelling methods in this thesis.

Definition 6.1.2. *Tree-dense graph: A tree-dense graph has the set of vertices partitioned into two subsets, called the core and the periphery, which satisfy the following properties.*

(1). *A set of vertices in the core represents a subgraph which is an almost complete graph.*

(2). *All vertices in an almost complete graph connect with the vertices on periphery to obtain a tree graph.*

In this chapter, we examine the Bloom filters of edges and shortest paths in an almost complete graph and star graphs which are a particular tree graph, and tree-dense graph, separately.

Some regular (e.g. star and tree) and irregular graphs as a network models has

been generated by [67].

A tree is an undirected graph where there is exactly one path between any pair of nodes. The tree graph, which is a possible shape for a network, has been chosen in the literature as a model in some applications of Bloom filters. For instance, the Bloom filter has been used in [54] as a storage of information when routing from one computer to another in a tree model network. The tree graph is a base for network model for some applications of Bloom filters [38] [66] where the links between computers or both links and nodes are labeled by Bloom filters. The routing from one sender to multiple receivers [1] [38] [39] [58] or the routing from computer to computer [54] [66] in a tree-shaped network by using Bloom filters is widely studied in literature.

Additionally, in this chapter, we examine a particular tree graph which is a star graph. A star is a graph with a single centered node connecting to all other nodes. For an implementation of the Bloom filters in star graphs the study [68] can be seen, as well as some studies of routing in star graphs are [5] [18] [19]. A routing algorithm through paths, which all are shortest in a star graph, has been proposed in [18] by using its basic graph-theoretical properties. The comparison of the graph-theoretical properties between the star graphs and hypercubes proposed in [5] and [19] has been another approach to the paths between nodes.

6.2 Decomposing a Tree-Dense Graph

A realistic network might have a complicated form which has a large number of nodes and a large number of edges. Hence, to manage the message traffic in these

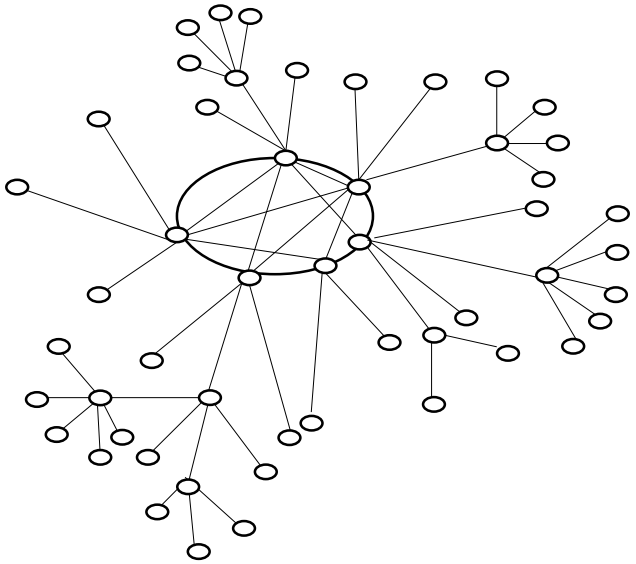


Figure 6.1: A tree-dense graph.

networks might be difficult and expensive [67]. We repeat the routing scenario from one computer to another computer as in the other chapters. We encode all edges with Bloom filters which has reasonably small length in tree-dense graphs. We aim to eliminate the adjacent false positives to enable the error-free communication between nodes.

We consider a special graph being similar to the Figure 6.1. As usual with the models in other chapters, each node has a computer in the tree-dense graph and the routing is provided by the shortest paths between nodes.

The core graph in a tree-dense graph might have a variety of forms, yet we assume it has a form of an almost complete graph. Since it is a realistic model in many applications of networks. An almost complete graph may have so many edges, in other words the number of edges might be much more than the number of vertices in an almost complete graph. This enables the usage of bit per vertex labelling which is

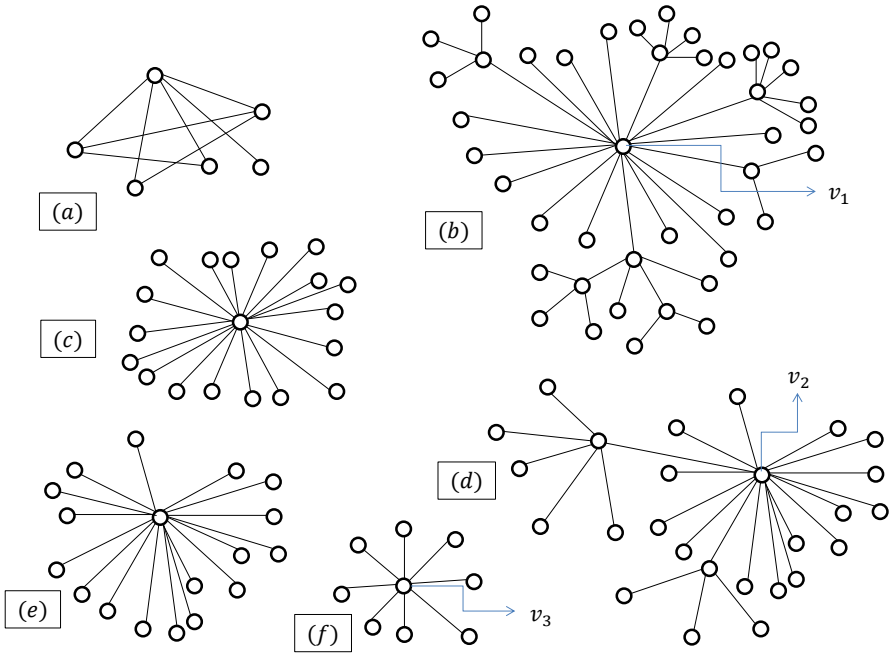


Figure 6.2: The parts of a tree-dense graph are obtained after the decomposition of the tree-dense graph given in the Figure 6.1.

an advantage for space efficiency (the details of bit per vertex labelling for edges in an almost complete graph is given in the Section 6.3.1). This is another advantage to use an almost complete graph as a core in a tree-dense graph.

We consider a decomposition of a tree-dense graph into parts. The core graph (the graph (a) in the Figure 6.2) is collapsed into one vertex (v_1 in the graph (b) in the Figure 6.2) and all vertices from the core are merged on this vertex (v_1), then a tree graph given in the Figure 6.2 (b) can be obtained. Accordingly, the tree-dense graph in the Figure 6.1 is decomposed into the core and the periphery. After the first decomposition, we have one almost complete graph (a) and one tree graph (b) given in the Figure 6.2.

Now, the graph (b) in the Figure 6.2 represents another tree-dense graph and a

star graph stands for a core in this graph. It is obvious that this tree-dense graph in the Figure 6.2 (b) can be further decomposed into parts. By repeating the same process as above, the tree-dense graph in the Figure 6.2 (b) can be decomposed into star graphs. The graphs in the Figure 6.2 (c) and (d) are these parts of the tree-dense graph (b) in the Figure 6.2. As a core graph of the graph (b), the graph (c) is collapsed into the vertex v_2 in the graph (d) in the Figure 6.2.

Note that, if any part of a tree-dense graph represents another tree-dense graph, then the process of decomposition lasts until none of the parts of the tree-dense graph have a form of another tree-dense graph. As seen the graph in the Figure 6.2 (d) can be still decomposed into parts, the graph (d) has another core graph which has a form of a star graph. By following the same process as above on the graph (d) in the Figure 6.2, the graphs (e) and (f) given in the Figure 6.2 are obtained. All vertices in the graph (e) are merged to the vertex v_3 in the graph (f) of the Figure 6.2, since the graph (e) is the core of the graph (d).

Hence, in total there are four parts in the tree-dense graph given with the Figure 6.1. These are the graphs in the Figure 6.2 (a) which is an almost complete graph denoted by C and $(c), (e)$ and (f) which are all star graphs denoted by S_1, S_2 and S_3,respectively, in the Figure 6.2.

Now, we are able to encode each edge in each part of the tree-dense graph individually (see section 6.3 for the encoding methods). There is a computer on each node, and remember that, we assume that the sender chooses a shortest path to carry the messages and encodes the path as a Bloom filter. Note that, there is only one path that is shortest between each pair of nodes in tree graphs. Hence, while a message is

not sent back from a computer which is on the shortest path in a tree-dense graph, the message follows exactly one possible path during distribution.

6.3 Encoding Edges in Parts of Tree-Dense Graphs

To encode edges in a tree-dense network, firstly, we split it into small parts which have the form of an almost complete graph and stars. After the decomposition is completed, the edges in any part of the tree-dense graph can be encoded individually. We aim to build Bloom filters with reasonably small number of bits for all edges in tree-dense graph.

6.3.1 Edge Coding in Almost Complete Graphs

We assume there is an almost complete graph in the middle of the tree-dense graph. The number of nodes in an almost complete graph is much smaller than the number of edges. Therefore, bit per-edge labelling is not optimal for this graph. In order to build the Bloom filters for all edges with a reasonably small number of bits, each vertex corresponds to one position in the Bloom filter. We enumerate every vertex denoted by $v_1, v_2, \ldots, v_n$ in an almost complete graph with an integer, then the bits 1 are placed in the bit position of the Bloom filter of an edge which corresponds to this number. Accordingly, the length of the Bloom filter of an edge is equal to the number of vertices in the almost complete graph. The end vertices of edges are represented by the bit 1 in the Bloom filter of the edge. For instance; when the end vertices of an edge are given with the pair $\{v_i, v_j\}$, then two bits 1 are placed in the bit positions

of ith and jth among n bits in the Bloom filter of an edge.

Note that, bit per vertex labelling might be used to encode edges in any graph, but it may not be an advantage for space efficiency in all graphs. This method works in an almost complete graph better than in many other graphs, since the number of vertices is much smaller than the number of edges in an almost complete graph. Also, another reason that we use an almost complete graph as a core of tree-dense graph is that it is used in many applications of computer systems as a form of a real network.

6.3.2 Edge Coding in Star Graphs

The edges in a star graph have a central node. As we will see in this section, the methods of labelling of all edges in a star graph depends on the number of edges in order to keep the length of the Bloom filters reasonably small.

Consider a star graph with n edges where $n < 3\sqrt{n} \iff n < 9$. It is useful to think of encoding each edge by one bit when n is reasonably small. When we enumerate the edges with an integer k, where $1 \leq k \leq n$, then the bit 1 is placed into the kth bit position in the n bits length Bloom filter. Obviously, the place of the bit 1 in the Bloom filter of an edge in this kind of star depends on the corresponding integer of the edge.

Now, when the number of edges n of a star graph satisfies that $3\sqrt{n} \leq n \iff 9 \leq n$, then the reasonable length of the Bloom filters can be thought as $3\lceil\sqrt{n}\rceil$ where $\lceil\sqrt{n}\rceil$ is the smallest integer which is not less than $\sqrt{n}$. A Bloom filter of an edge consists of three parts where each part is called projection and each projection

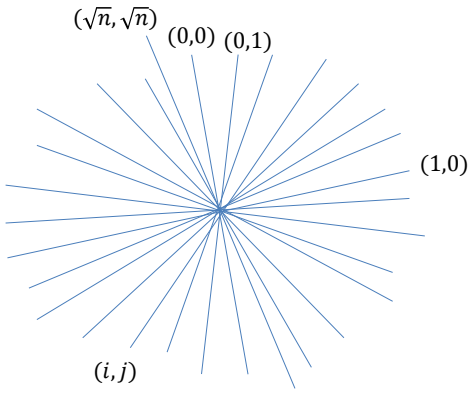


Figure 6.3: Representative points of edges in a star graph where the number of edges n satisfies that $3\sqrt{n} \leq n \iff 9 \leq n$.

contains one bit 1. All edges are assigned with ordered pairs from a coordinate plane of two axes x, y (see Figure 6.3). The bit 1 in the Bloom filter takes place based on representative points of edges.

For instance, an edge e in a star is represented by the points (i, j) (see Figure 6.3), then the bits 1 are placed on the i, j and $(i + j) \pmod{\lceil\sqrt{n}\rceil}$ bit positions in the first, second and third projections, respectively. In order to save the space we keep the length of the Bloom filters of edges as small as possible. There can be found a positive integer m which satisfies the inequality that $m^2 < n \leq (m + 1)^2$. When $m^2 < n \leq (m + 1)^2$, then the representative points (i, j) of edges are collected in the set $\{0, 1, \ldots, \sqrt{(m + 1)^2}-1\}$. In other words, if $m^2 < n \leq (m+1)^2$, then the places for the bit 1 in the Bloom filters of edges are specified with respect to $\pmod{\sqrt{(m + 1)^2}}$ (equivalently $\pmod{(m + 1)}$). Otherwise, there cannot be found the representative points for some edges whose number is $n - m^2$. For example, the number of edges n might be 14, then $m = 3$, since $3^2 < 14 \leq 4^2$. For these parameters, the places for the

bits 1 in the Bloom filters of edges are specified with modulo 4 where $0 \leq i, j \leq 3$. Hence, there are enough number of pairs of points per edge where there are 16 pairs of points and 14 edges. Otherwise, when the places of the bits 1 were performed by modulo 3, then there would be 9 pairs of points which is less than the number of edges. Hence, some edges whose number is $14 - 9 = 5$ would not be represented by pairs of points, hence some edges could not be encoded by a Bloom filter.

On the other hand, when $m^2 = n$, then the bit 1 is placed on the bit position of i, j and $(i + j) \pmod m$ in the first, second and third projections, respectively. Accordingly, the length of the Bloom filter of the edge is either $3\sqrt{(m+1)^2} = 3(m+1)$ or $3\sqrt{m^2} = 3m$ based on the number of the edges. Note that, $3\sqrt{n} = n$ occurs when $n = 0$ or $n = 9$.

Overall, there are two different labelling methods for edges in star graphs in order to keep the length of the Bloom filter at its optimum value. These methods are based on the number of edges in the star.

6.3.3 Bloom Filters of the Edges in a Tree-dense Graph

We assume that a tree-dense graph is decomposed into stars denoted by $S_1, S_2, \ldots, S_t$, where t is the number of star graphs, and an almost complete graph denoted by C. Hence, an edge of any graph decomposed is an edge of a tree-dense graph. The Bloom filter of an edge in any part of the tree-dense graph is labeled individually by using an appropriate method introduced in the sections 6.3.1 and 6.3.2. Let an edge e_{S_i} from a star S_i of a tree-dense graph be represented by a Bloom filter $\beta(e_{S_i})$,

and $\beta(0_j)$ where $i \in \{1, \ldots, t\}$ and $j \in \{C, S_1, \ldots, S_t\} \setminus S_i$ is the Bloom filter of all bits 0. The Bloom filter of an edge e in a tree-dense graph should be formed as concatenation of Bloom filters with all bits 0 and the edge e itself in an order such as $\beta(e) = \beta(0_C)\beta(0_{S_1}) \ldots \beta(e_{S_i}) \ldots \beta(0_{S_t})$.

The length of the Bloom filters $\beta(0_j)$ depends on the length of the Bloom filters of the edges in corresponding parts of tree-dense graph. Accordingly, the total length of Bloom filter of an edge in the tree-dense graph is $3(\lceil\sqrt{n_1}\rceil + \lceil\sqrt{n_2}\rceil + \cdots + \lceil\sqrt{n_p}\rceil) + (v_c + n_{p+1} + n_{p+2} + \cdots + n_t)$ where $n_1, n_2, \ldots, n_p, n_{p+1}, \ldots, n_t, v_c \in \mathbb{Z}^+$ are the number of edges in the subgraphs $S_1, \ldots, S_t$ and the number of vertices in the graph C.

6.4 Routing without False Positives

Theorem 6.4.1. *The Bloom filter of a shortest path in an almost complete graph does not yield a false positive adjacent to the path.*

Proof. Suppose that the number of vertices in an almost complete graph is n which denotes the length of the Bloom filter of edges and the path in the graph. Also, it is assumed that the vertices in a shortest path are $v_0, v_1, v_2, \ldots, v_j$ where the shortest path lies between the vertices v_0 and v_j and $j < n$. Note that, subscripts $0, 1, \ldots, j$ show the sequence of the vertices on the shortest path, and the number of the edges on the shortest path is j. The vertices might be assigned by other integers rather than $0, 1, \ldots, j$, so the bit 1 in the Bloom filter of the shortest path does not necessarily appear in the same order with the subscripts $0, 1, \ldots, j$ of representatives of vertices (see Figure 6.4).

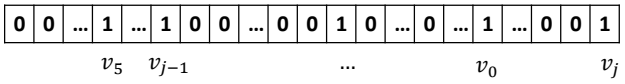


Figure 6.4: The Bloom filter of a path containing the vertices $v_0, v_1, v_2, \ldots, v_j$ in an almost complete graph.

Suppose that an edge f is an adjacent edge to a shortest path in an almost complete graph. Since the edge f is adjacent to the shortest path with endpoints v_i and v_l, one of the endpoints of the edge f links with the shortest path where $v_i \notin \{v_0, v_1, v_2, \ldots, v_j\}$ and $v_l \in \{v_0, v_1, v_2, \ldots, v_j\}$. Suppose the Bloom filter of the path contains the bit 1 on the bit position of representative integer of the vertex v_i. All vertices $v_0, v_1, v_2, \ldots, v_j$ in the set are represented by an integer and the Bloom filter of the shortest path reflects them in the corresponding bit positions, yet v_i is not a member of this set. Accordingly, there is no other bit 1 in some places in the Bloom filter of the shortest path instead of the bit positions of representative positions of the vertices $v_0, v_1, v_2, \ldots, v_j$. Hence, the edge f is an adjacent edge to the shortest path, it is not a false positive of the Bloom filter of the shortest path.

□

Note that, we do not get a false positive for an edge which is not adjacent to the given shortest path in an almost complete graph. Each edge has two endpoints which are numbered uniquely, therefore each edge has a unique Bloom filter in an almost

complete graph. As usual we use shortest paths in our routing model, that is why we examine only shortest paths in an almost complete graph, and the adjacent edges to the shortest paths are only the threats for our routing model.

Additionally, the bit per vertex labelling does not produce a false positive in any graph. Yet, in the core of the tree-dense graph, we examine an almost complete graph, since it is commonly used in many applications as a model of network. Also, this method works in an almost complete graph better than in many other graphs, since the number of vertices is much smaller than the number of edges in an almost complete graph. This is an advantage for space efficiency.

Theorem 6.4.2. *The Bloom filter of a shortest path in a star graph does not yield a false positive adjacent to the shortest path.*

Proof. There are two types of encoding methods introduced in section 6.3.2 for the edges in star graphs, hence we examine whether a star graph contain a false positive in these two types of stars on individual basis. We query the existence of false positives in the star graphs with the number of edges $n < 9$ and the $n \geq 9$, separately, since the labels of edges differ in these both kind of stars.

Firstly, when $n < 9$, all edges are assigned by an integer from the interval $[1, n]$, and the length of the Bloom filter is the number of edges n. A shortest path in a star graph consists of at most two edges. If the shortest path contains one edge in a star graph, then obviously there is no false positive.

Suppose the shortest path consists of two edges e and g, and an edge f adjacent to a shortest path in this star graph where $n < 9$. The bit 1 is placed on two distinct

bit positions, say i and j where $1 \leq i, j \leq n$, in the Bloom filter of the shortest path, since the edges are distinct and numbered by different integers. Suppose the edge f is a false positive. The Bloom filter of the shortest path contains the bit 1 on two bits positions i and j. Hence, the bit 1 in the Bloom filter of the edge f must be placed either ith or jth bit position. Yet, these bit positions are reserved for the edges e and g, respectively. Otherwise, the Bloom filter of the shortest path contains three bits 1, yet the Bloom filter of the shortest path contains at most two bits 1. Therefore, the edge f is either an edge named e or g on the shortest path or an adjacent edge to the shortest path in the star graph where the number of edges less than $3\sqrt{n}$.

Now, we consider the stars with the number of edges $n \geq 9$. To provide that the length of the Bloom filter of edges is an integer, the number of edges n satisfies the inequality $m^2 < n \leq (m+1)^2$.

Assume the edges denoted e and g belongs to a path in a star graph. Suppose that the edges e and g are represented by the ordered pairs (i, j) and (k, l) of two coordinate axes, respectively, where $(i, j) \neq (k, l)$. Therefore, the Bloom filter of the shortest path of a star contains the bits 1 in at most six different bit positions such as $i, j, k, l, (i+j), (k+l) \pmod{(m+1)}$ where the number of edges n satisfies the inequality $m^2 < n \leq (m+1)^2$. If $m^2 = n$, then the bit 1 takes bit positions with respect to $\pmod{\sqrt{m^2}}$. Yet, the coding method for these two values does not make a difference. It only affects the length of the Bloom filter where it is $3\sqrt{(m+1)^2}$ when $m^2 < n \leq (m+1)^2$ and $3\sqrt{m^2}$ when $n = m^2$, this is for sure all edges are represented by a pair of points. Therefore, in the rest of the proof we assume the modulus of the congruence is the integer $(m+1)$ which produces the same result with $\pmod{m}$.

Consider another edge f in the same star which is an adjacent edge to the shortest path and let the representative points of the edge f be (p, q) where $0 \leq p, q \leq m$. If all three bits of 1 in the Bloom filter of f take exactly the same places with any three bit positions of $i, j, k, l, (i+j), (k+l) \pmod{(m+1)}$ of the Bloom filter of the shortest path, then we are able to say that the edge f might be a false positive of the Bloom filter of the shortest path.

A Bloom filter of an edge in a star graph with the number of edges $n \geq 9$ consists of three parts where each part is called projection. There are $\binom{2}{1}\binom{2}{1}\binom{2}{1} = 8$ possible triple combinations of points $i, j, k, l, (i+j), (k+l) \pmod{(m+1)}$ which represent the places of the bit 1 in the Bloom filter of the edge f, since there is one option among two possible positions per projection and there are three projections in Bloom filters of edges. If the bit 1 takes places in the Bloom filter of the edge f in any possible combinations of the points given with the set $C = \{(i, j, u), (k, l, v), (i, l, u), (i, l, v), (i, j, v), (k, l, u), (k, j, u), (k, j, v)\}$ where $0 \leq i, j, k, l \leq m$ and $i+j \equiv u \pmod{(m+1)}$ and $k+l \equiv v \pmod{(m+1)}$, then we may conclude that the edge f is a false positive of the Bloom filter of the shortest path containing edges e and g.

Now we consider the results when the Bloom filter of the edge f matches up with the Bloom filter of the path, containing the edges e and g, in all positions of the bit 1.

First of all, when the bit 1 in the Bloom filter of the edge f takes places on the bit positions of triples either (i, j, u) or (k, l, v) in first, second and third projections, respectively. This occurs when $(i, j) = (p, q)$ or $(k, l) = (p, q)$. Yet, all edges in a star

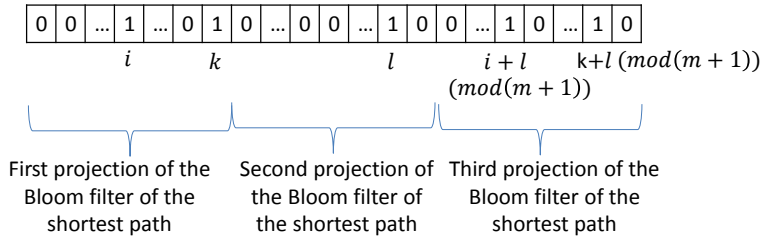


Figure 6.5: The Bloom filter of a path in the star graph when $j = l$.

graph are assigned by unique points where $9 \leq n$. Hence, the edge f is either the edge e or g with these points.

Suppose that the bit 1 is placed on the bit positions corresponding triple points (i, j, v). This happens when $(p, q) = (i, j)$, and in the third projection the bit 1 takes place on the bit position of $p + q \equiv i + j \pmod{(m+1)}$. Besides, with given this triple combinations it is obtained that $i + j \equiv v \pmod{(m+1)}$ and it is known that $k + l \equiv v \pmod{(m+1)}$, hence $k + l \equiv i + j \pmod{(m+1)}$. Accordingly, the bit 1 takes place in the bit positions of the Bloom filter of the edge f on the triple points $(i, j, k + l \equiv i + j \pmod{(m+1)})$ which is exactly the same bit positions of the bit 1 in the Bloom filter of the edge e. Therefore, f is not a false positive of the path in the star with given triple encoding points. It is obvious that the Bloom filters of the edges g and f are identical when the bit 1 is placed in the bit positions of (k, l, u) in the Bloom filter of the edge f. Since, with given encoding points of the edge f it is obtained that $u \equiv i + j \equiv k + l \pmod{(m+1)}$.

Additionally, suppose the Bloom filter of the edge f contains the bit 1 on the bit

positions of triples either (i, l, u) or (i, l, v). Accordingly, the representative points of the edge f is that $(p, q) = (i, l)$ in both cases. Therefore, $i + l$ is equivalent to either $i + j \pmod{(m+1)}$ or $k + l \pmod{(m+1)}$. Since the bit 1 in the third projection of the Bloom filter of the edge f should be placed in either one of these sums. When the Bloom filter of the edge f contains the bit 1 on the bit positions of triple (i, l, u), then $i + l \equiv i + j \pmod{(m+1)}$, hence $l = j$. Given with these values of the bit positions, the Bloom filter of the shortest path have a form like in the Figure 6.5. This occurs, when the coding points for the edge e is given when $(i, j) = (i, l)$ which is the assumption of the representative points of the edge f. When the edge f is assigned by the coordinate point of (i, l), then the Bloom filter of the f is equal to the Bloom filter of the edge e in all bit positions. Therefore, the edge f is not a false positive with the bit positions of triple (i, l, u). Similarly, when the Bloom filter of the edge f contains the bit 1 on the bit positions of triple (i, l, v), then $i + l \equiv k + l \pmod{(m+1)}$, hence $i = k$. In this case, the Bloom filter of the shortest path has a form like in the Figure 6.6. This happens, when the coding points for the edge g is given when $(k, l) = (i, l)$. Accordingly, the Bloom filters of the edges g and f are obtained as identical with the triple (i, l, v).

Finally, suppose the bit 1 in the Bloom filter of the edge f takes places on the bit positions of the triples (k, j, u) or (k, j, v). In both cases, the representative points of the edge f is (k, j). By using same arguments as above, the bit 1 in the third projection of the Bloom filter of the edge f takes bit positions of either $(i + j) \pmod{(m+1)}$ or $(k + l) \pmod{(m+1)}$ which are the same bit positions of either the edge e or g. Also, it is observed that $k + j \equiv i + j \pmod{(m+1)}$, hence $k = i$

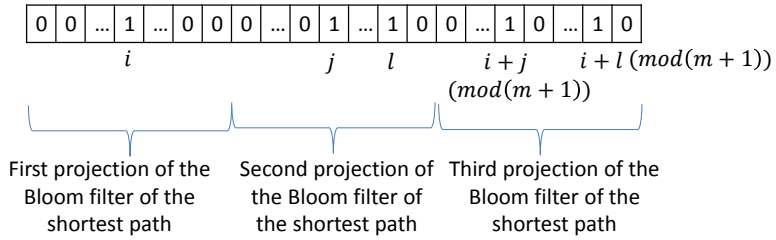


Figure 6.6: The Bloom filter of a path in the star graph when $k = i$.

or $k + j \equiv k + l \pmod{(m+1)}$, hence $j = l$ with these triples, respectively. Hence, this occurs, when the representative points of the edges e and g are $(i, j) = (k, j)$ and $(k, l) = (k, j)$. Accordingly, the edge f is obtained as an edge, which is either e or g, in the shortest path in a star graph with the number of edges being greater than or equal to $3\sqrt{n}$.

As a result of the observations above, we can definitely say whether any edge is in the star graph or not. Therefore, the Bloom filter of a path in a star graph does not produce a false positive. □

Theorem 6.4.3. *The Bloom filter of a shortest path in a tree-dense graph does not yield a false positive adjacent to the shortest path.*

Proof. Consider that a tree-dense graph is split into one almost complete graph denoted by C and t number of stars denoted by $S_1, S_2, \ldots, S_t$. The edges in all parts of tree-dense graph are encoded by an appropriate method, and an edge in any part which is either an almost complete graph or a star graph is an edge in

the tree-dense graph. Decomposition of the Bloom filters of shortest paths of all parts generates the Bloom filter of the shortest path in the tree-dense graph. Hence, the Bloom filter of the shortest path in the tree-dense graph can be illustrated as a concatenation of the Bloom filters of shortest paths in the parts such that $\beta(P) = \beta(P_C)\beta(P_{S_1})\beta(P_{S_2})\dots\beta(P_{S_t})$ where P is the shortest path in the tree-dense graph and P_k where $k \in \{C, S_1, \dots, S_t\}$ is the shortest path in a part of the tree-dense graph.

Suppose that an edge f is an adjacent to a shortest path in a tree-dense graph. The edge f might belong to any star graph S_j where $j \in \{1, \dots, t\}$ or to the almost complete graph C. The Bloom filter of f in the star S_j or the almost complete graph is denoted by $\beta(f_{S_j})$ or $\beta(f_C)$. Therefore, as an edge in tree-dense graph, the Bloom filter of the edge f can be denoted by $\beta(f) = \beta(0_C)\beta(0_{S_1})\dots\beta(f_{S_j})\dots\beta(0_{S_t})$ or $\beta(f) = \beta(f_C)\beta(0_{S_1})\dots\beta(0_{S_j})\dots\beta(0_{S_t})$ where all bits of $\beta(0_i)$ is 0 and $i \in \{C, S_1, \dots, S_t\} \setminus S_j$ when f is in the set S_j or $i \in \{S_1, \dots, S_t\}$ when f is in the almost complete graph.

Suppose the edge f is a false positive in a star graph. Hence, it must be satisfied that $\beta(f) \leq \beta(P)$ in all bits positions. Obviously, $\beta(0_C) \leq \beta(P_C)$ and $\beta(0_{S_h}) \leq \beta(P_{S_h})$, where $h \in \{1, 2 \dots, t\} \setminus j$, in all bits positions. Yet, by the Theorem 6.4.2, a star graph does not produce a false positive, therefore the Bloom filter of the edge f cannot be less than or equal to the Bloom filter of the path in the star graph S_j in all bits positions. Therefore, the edge f is not a false positive of the Bloom filter of the shortest path of a tree-dense graph.

Similarly, suppose the edge f is a false positive in an almost complete graph. By following the same argument as above, a contradiction of Theorem 6.4.1 is obtained.

Hence, when any part of a tree-dense graph does not produce a false positive, then the tree-dense graph does not yield any false positive with the coding method of edges that we introduced.

□

Unlike the other encoding methods that we introduced for other graphs in previous chapters, there is no false positive for an edge which is not adjacent to the given shortest path in a tree-dense graph.

Suppose an edge e is not adjacent to a shortest path in a tree-dense graph. If this edge is an edge in the almost complete graph, then $\beta(e) = \{u, v\}$ where u and v are the endpoints of the edge e. There is no other edge in the almost complete graph that coincides with these endpoints, hence $\beta(e) = \beta(e_C)\beta(0_{S_1}) \dots \beta(0_{S_j}) \dots \beta(0_{S_t})$ is not less than or equal to $\beta(P) = \beta(P_C)\beta(P_{S_1})\beta(P_{S_2}) \dots \beta(P_{S_t})$ in all bits positions where one almost complete graph denoted by C and t number of star graphs denoted by $S_1, S_2, \dots, S_t$ are the parts of a tree-dense graph obtained after decomposition ,and P_j where $j \in \{C, S_1, S_2, \dots, S_t\}$ is the shortest path in the parts of the tree-dense graph and P is the shortest path in the tree-dense graph.

Similarly if the edge e is in a star graph S_i where $1 \leq i \leq t$ and t is the number of star graphs obtained after decomposition of a tree-dense graph. The edges in a star graph are either enumerated by an integer when $n < 9$ or represented by ordered pair when $n \geq 9$ where n is the number of edges in the star graph. In both cases each edge has its own representative integer or ordered pair, therefore the edges are encoded by unique Bloom filters in star graphs. Hence, $\beta(e) = \beta(0_C)\beta(0_{S_1}) \dots \beta(e_{S_i}) \dots \beta(0_{S_t})$ is

not less than or equal to $\beta(P) = \beta(P_C)\beta(P_{S_1})\beta(P_{S_2})\dots\beta(P_{S_t})$ in all bits positions.

6.5 Comparing Arbitrary Encoding Methods with Our Approach in Tree-dense Graphs

One encoding method is to label per edge with one bit. If each edge would have been represented by a bit, then the length of Bloom filter of an edge would be the number of edges in the tree-dense graph. For big $n \in \mathbb{Z}^+$, it is satisfied that $3\sqrt{n} < n \iff 9 < n$. The length of Bloom filter of an edge in the tree-dense graph is $l = 3(\lceil\sqrt{n_1}\rceil + \lceil\sqrt{n_2}\rceil + \dots + \lceil\sqrt{n_p}\rceil) + (v_c + n_{p+1} + n_{p+2} + \dots + n_t)$ where $n_1, n_2, \dots, n_p, n_{p+1}, \dots, n_t \in \mathbb{Z}^+$ are the number of edges in the parts of the tree-dense graph and v_c is the number of vertices in an almost complete graph, when our encoding methods are applied to the edges in a tree-dense graph.

On the other hand, the total number of edges in a tree-dense graph is $s = n_1 + n_2 + \dots + n_p + (n_c + n_{p+1} + n_{p+2} + \dots + n_t)$ where $n_1, n_2, \dots, n_p, n_{p+1}, \dots, n_t, n_c \in \mathbb{Z}^+$ are the number of edges in the parts of the tree-dense graph. We know that the number of vertices in an almost complete graph is much less than the number of edges, namely $v_c < n_c$, therefore $l < s$ for some star graphs whose edge numbers are > 9.

Hence, this is an advantage that our encoding method produces shorter Bloom filters than the Bloom filters obtained by using bit-per-edge labelling. This shows that our encoding method for the edges in tree-dense graphs saves space.

Another encoding method is using the standard Bloom filter. The standard Bloom

filter produces false positives. Therefore, if we had encoded the edges in a tree-dense graph by using the methods building the standard Bloom filter, then we would obtain false positives. We examine this by using our parameters in standard Bloom filters.

The maximum number of edges in a shortest path in a star and an almost complete graph are 2 and r, respectively. Therefore, the maximum number of edges in a shortest path of the tree-dense graph is $2t + r$, where t is the number of star graphs, when all parts of tree-dense contain edges from the shortest path.

The optimum number of bit 1 is $k = \lceil \ln 2 \times \frac{m}{n} \rceil$ where m is the length of the Bloom filter of elements (edge in our model) and n is the number of elements in the set (edges on the shortest path in our model), when the probability of false positives is assumed to be minimum, [46]. If the edges were encoded by using the methods building a standard Bloom filter, then the number of bit 1 in the Bloom filter of an edge in some star graphs would be $k = \lceil \ln(2) \times \frac{3\lceil\sqrt{n_i}\rceil}{2} \rceil$ where $i \in \{1, 2, \ldots, p\}$ or for some stars, whose edges are assigned by an integer, $k = \lceil \ln(2) \times \frac{n_j}{2} \rceil$ where $j \in \{p+1, p+2, \ldots, t\}$. In both cases, k depends on the number of edges in the star graph. In our model, the number of the bit 1 in the Bloom filter of an edge in some star graphs is 3, when $9 \leq n$. Otherwise, the Bloom filters of edges in some star graphs contain one bit 1. Similarly, for an almost complete graph the optimum number of bit 1 in the Bloom filter of an edge would be $k = \lceil \ln(2) \times \frac{v_c}{r} \rceil$ where r is the maximum number of edges on the shortest path in an almost complete graph and v_c is the number of vertices in the almost complete graph, if the edges would have been encoded randomly. We encode the edges in an almost complete graph with the Bloom filters containing one bit 1.

If the edges in a tree-dense graph were encoded by using random methods, then the optimum number of the bit 1 in the Bloom filter of an edge in the tree-dense graph would look like the equation (6.1).

$$k = \lceil \ln(2) \times \frac{3(\lceil\sqrt{n_1}\rceil + \lceil\sqrt{n_2}\rceil + \cdots + \lceil\sqrt{n_p}\rceil + (v_c + n_{p+1} + n_{p+2} + \cdots + n_t)}{2t + r} \rceil \tag{6.1}$$

which is equal to $\lceil \ln(2) \times \frac{l}{2t+r} \rceil$ where l is the length of the Bloom filters of edges in a tree-dense graph. For reasonable number of edges in each part, equation (6.1) would result in the bit 1 with a large number. Despite the fact that, in our model, the maximum number of the bit 1 in Bloom filter of an edge in tree-dense graph is 3.

By using random methods, the Bloom filter may produces false positives. The probability of false positive of standard Bloom filter is $\left(1 - e^{\frac{-kn}{m}}\right)^k$, [7]. Accordingly, if an edge would have been encoded by using any method which places the bit 1 in the Bloom filter randomly, then the probability of false positives would be the following equation.

$$\left(1 - e^{\frac{-kn}{m}}\right)^k = \left(1 - e^{\frac{-\lceil \ln(2) \times \frac{l}{2t+r} \rceil (2t+r)}{3(\lceil\sqrt{n_1}\rceil + \lceil\sqrt{n_2}\rceil + \cdots + \lceil\sqrt{n_p}\rceil) + (v_c + n_{p+1} + n_{p+2} + \cdots + n_t)}}\right)^{\lceil \ln(2) \times \frac{l}{2t+r} \rceil} \tag{6.2}$$

It is not difficult to compute this equality, when all parameters of whole tree-dense graph are known. When the number of edges increases in a tree-dense graph, the probability of false positives decreases. We might obtain more concrete and simple

results, when the main graph has a form of a full binary tree which is very simple and easy to observe.

6.5.1 An Example of a Tree-dense Graph: A Full Binary Tree

In this section of this chapter, we assume the edges in a full binary tree are encoded by using random methods. A full binary tree is a tree that every node other than the leaves connects with two nodes. Hence, we examine the ratio of false positives in this graph.

All parts of a tree-dense graph might be star graphs. A full binary tree (see Figure 6.7) satisfies this property. If we decompose a full binary tree graph into parts by following the process described in the Section 6.2, then each part has a form of a star graph. In this graph core is the vertex denoted by v_1 in the Figure 6.7 (a) on the root of the full binary tree. It is decomposed into a 2-edge star graph rooted v_1. This 2-edge star graph is collapsed into one vertex denoted by v_2, then a tree graph in the Figure 6.7 (c) is obtained. After the first decomposition we have one 2-edge star graph (b) in the Figure 6.7 and a tree graph (c) in the Figure 6.7. The graph (c) in the Figure 6.7 can be decomposed into further parts. By following the same process, a 4-edge star graph (d) in the Figure 6.7 is obtained, and all vertices in the Figure 6.7 (d) are merged into a vertex v_3, then a 8-edge star graph in the Figure 6.7 (e) is obtained. Finally, after the decomposition of the full binary tree given in the Figure 6.7 (a) three star graphs $(b), (d), (e)$ given in the Figure 6.7 are obtained.

The numbers of the edges in each part (where all parts are star graphs) of a full

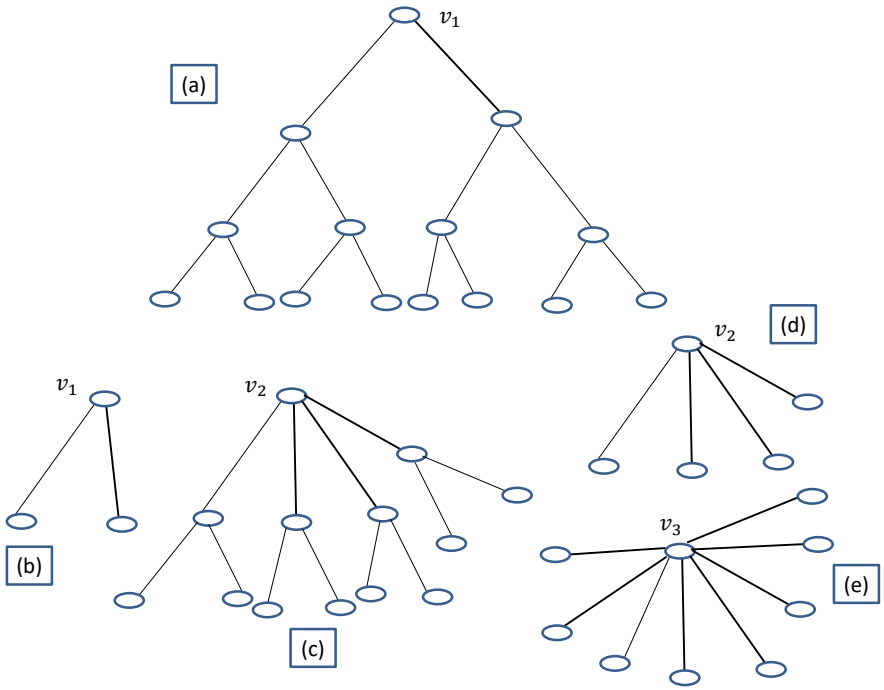


Figure 6.7: A full binary tree (a) where the number of edges is $2^1 + 2^2 \cdots + 2^r$ and $r = 3$, and the star graphs are obtained after the decomposition of this tree.

binary tree graph is the sequence of integers that $2, 4, 8, 16, 32 \ldots$ which is briefly 2^r where r is the number of leaves of the tree as well as the number of parts of a full binary tree graph. Hence, the total number of edges n in a full binary tree is $2^1 + 2^2 + \cdots + 2^r$ which is the length of the Bloom filter of edges in a full binary tree when each edge is represented by one bit in the Boolean array of a shortest path in a full binary tree. When our encoding method is applied to the edges in the full binary tree, then the length of the Bloom filter of an edge is obtained as $2 + 4 + 8 + 3\lceil\sqrt{16}\rceil + \cdots + 3\lceil\sqrt{2^r}\rceil$. Recall that in our encoding method the length of the Bloom filter in tree-dense graph is the sum of the lengths of the Bloom filters in all parts of a tree-dense graph, and the length of the Bloom filter of an edge in a star graph is $3\sqrt{n}$ when $9 \leq n$ where n is the number of edges in a star graph,

otherwise it is n . Obviously, $2+4+8+3\lceil\sqrt{16}\rceil+\cdots+3\lceil\sqrt{2^r}\rceil < 2^1+2^2+\cdots+2^r$ when $r \geq 4$. Our approach in the full binary tree saves more space than bit-per-edge labelling method, since the length of the Bloom filters in our model is shorter than the Bloom filters obtained by applying bit-per-edge labelling method.

If the edges were encoded with standard Bloom filter by using our parameters in full binary tree, then we would have false positives in a full binary tree graph. The table in the Figure 6.8 shows the probability of false positives obtained by applying standard Bloom filter to the edges with our parameters. Remember that in a tree graph every path is shortest between nodes. Hence, each leaf contains at most one edge from the shortest path, since there is only one way between two distinct nodes in a full binary tree. Therefore, each star as a part of a full binary tree contains one edge from the shortest path.

It is known from the Section 6.5 that the optimum number of the bit 1 in the Bloom filter of an edge is obtained from the formula $\lceil \ln(2) \times \frac{m}{t} \rceil$ where m is the length of the Bloom filter and t is the maximum number of edges in the shortest path. As seen in the Figure 6.8, the length of the Bloom filter of an edge increases faster than the maximum number of edges in a path of a star graph when the size of full binary tree increases. Therefore, the number of the bit 1 in a Bloom filter of an edge rises when m and t increase.

As an experiment in the Figure 6.8, if the Bloom filter of the edges in a full binary tree had been build by an algorithm that distributes the bits arbitrary bit positions in the Bloom filter, then a false positive would have probably been obtained. When a full binary tree was constructed with extremely small size, the probability of false

r	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
s	2	4	8	16	32	64	128	256	512
n	2	6	14	30	62	126	254	510	1022
t	1	2	3	4	5	6	7	8	9
m	2	6	14	26	44	68	104	152	221
k	1	2	3	4	6	7	10	13	17
p	~0.39	~0.23	~0.10	~0.04	~0.014	~0.004	~0.0007	~0.0001	~0.000007

r: number of leaves of a binary tree,
s: number of edges in each leaf,
n: total number of edges in a binary tree,
t: maximum number of edges in a shortest path,
m: the length of Bloom filter of an edge,
k: number of the bit 1 in the Bloom filter of an edge,
p: probability of false positives of the Bloom filter of a path.

Figure 6.8: The probability of false positives in a full binary tree changes when the parameters change.

positive would be obtained rather high.

CHAPTER 7

ARBITRARY GRAPHS

7.1 Introduction

We have chosen consciously variety of graphs in this thesis. In this chapter, we pick a model which looks like the graph in the Figure 7.2. We have been inspired to use this graph by the graph given in the Figure 7.1 (taken from [35]), since we would like to show how our theoretical approach works in real world network models. One particular collection of networks is called *Topology zoo*, [17] [35] [36] (see Figure 7.1). This is an example of a real world network. As a real world network model, the Figure 7.1, [35], shows the links between the computers in the network in Kentucky, USA, 2010.

We have encoded edges in networks by using some useful methods in previous chapters. In this chapter, we attempt to generalise a method to encode edges in an arbitrary graph. As we will see in the following sections of this chapter, this approach may require more space than the previous methods we introduced in previous chapters

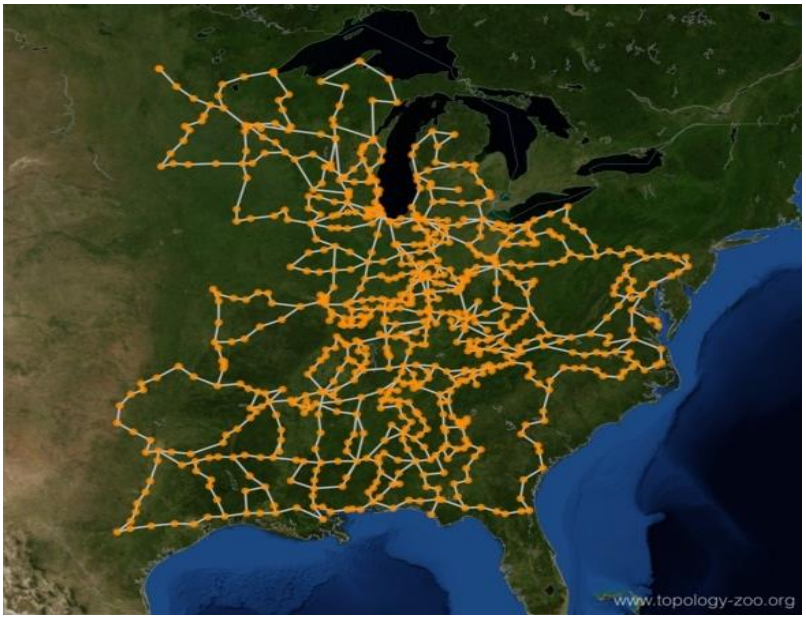


Figure 7.1: An example from the Topology zoo project [35].

(see the Section 7.4 for a concrete example). In practice, this may not be useful, but in theory this method might be helpful for some users who need particular methods. This method might be appropriate for arbitrary graphs (see Figure 7.2).

The Figure 7.2 presents an abstract example of arbitrary graph and some networks in real world might have similar shape.

7.1.1 Routing Scenario in Arbitrary Graphs

As usual, we assume there is a computer on each vertex, yet note that the links between nodes do not intersect each other.

Likewise throughout the thesis, we assume that the messages are delivered through the shortest paths between two distinct nodes in an arbitrary graph. As a possible scenario, same shortest paths might be preferred by multiple senders. This makes

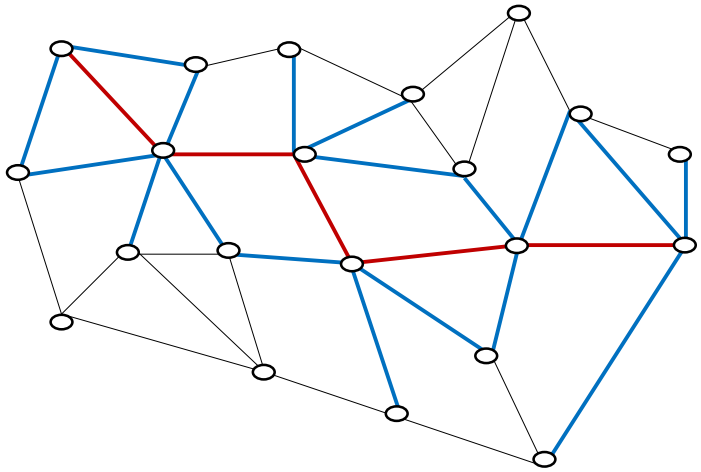


Figure 7.2: An abstract example of an arbitrary graph with a computer on each vertex.

these routes probably busy. In that case, some senders might choose longer routes for message delivery. In practical applications, it is sometimes possible that a message might be delivered via a path which is not a shortest path. Also, sometimes a message must be delivered to multiple destinations (this is what is called multicast, as opposed to unicast). In this chapter, we do not consider these generalisations.

We consider shortest paths for routing, and longest shortest paths for encoding in arbitrary graphs. A longest shortest path is a path which is not a subpath of any other shortest path. Note that, each longest shortest path might contain different numbers of edges in the graph.

Another direction of research is to consider directed graphs, with two edges pointing in the opposite directions (that is, from vertex u to vertex v and from vertex v to vertex u) having distinct labels; as usual we consider undirected graphs.

We encode the edges with Bloom filters, and the Bloom filter of the shortest paths

is obtained by applying bitwise OR operation to all edges on the shortest path. In this model of network, we define *ladders* (consult Section 7.2) that are used for encoding the edges in any network. Recall that we have given a ladder definition in Chapter 3, yet the definition of ladder in this chapter is slightly different than the previous one.

As usual, once a computer on the shortest path receives the message and the Bloom filter of the shortest path together, it does not send them back and forwards it to the next computer through on the shortest path. We query the edges between the computers by comparing the Bloom filter of the shortest path and the Bloom filter of the edge. In this case, the adjacent edges to the shortest path are the possible false positives.

7.2 Edge Coding in Arbitrary Graphs

Definition 7.2.1. *Ladder: The set of adjacent edges to a longest shortest path constitutes a ladder.*

We count in all adjacent edges to a particular longest shortest path in one ladder (in the Figure 7.2 the red path is one of the longest shortest path and the blue edges which are adjacent to this longest shortest path constitutes a ladder). Obviously, after specifying all possible longest shortest paths, all edges are specified either as an edge on the path or an adjacent edge to the path. Note that, while an edge has a role as an edge on a shortest path, the same edge might be an adjacent edge to another longest shortest path. Hence, each edge might be a member of some ladders except the ladders containing adjacent edges of the paths containing that edge.

Therefore, when the total number of longest shortest paths is p, then the number of ladders is $\leq p$. Since, some distinct longest shortest paths might have some adjacent edges which belong to the same ladder. In this network model, each ladder is represented by one bit, and each ladder is numbered by an integer. Therefore, the length of the Bloom filter of an edge is the number of ladders whose number is up to p. The Bloom filter of an edge contains the bit 1 in the places the corresponding number of ladders that this edge belongs to. An edge might belong to a maximum of k different ladders, then the number of bit $1 \in [1, k]$. The places of these bits depend on the number of the ladders containing the corresponding edge. The Bloom filter of a path is found by applying bitwise OR operation to the Bloom filters of the edges on the path.

7.2.1 A Small Example

In this section, we show the encoding method for arbitrary graphs with an example. We consider a star graph and a line as a graph like in the Figure 7.3 (a) and (b), respectively.

The longest shortest paths in the graph (a) in the Figure 7.3 contain two edges. There are 3 longest shortest paths, which are denoted by P, Q and R, in this graph. Hence, there are 3 ladders. We enumerate the ladders denoted by L_P, L_Q and L_R which contain the adjacent edges to the path P, path Q and path R, respectively, as $1, 2$ and 3. The length of the Bloom filters is the number of ladders which is 3. The edge e is an adjacent edge to the path R and belongs to the ladder L_R, the edge f

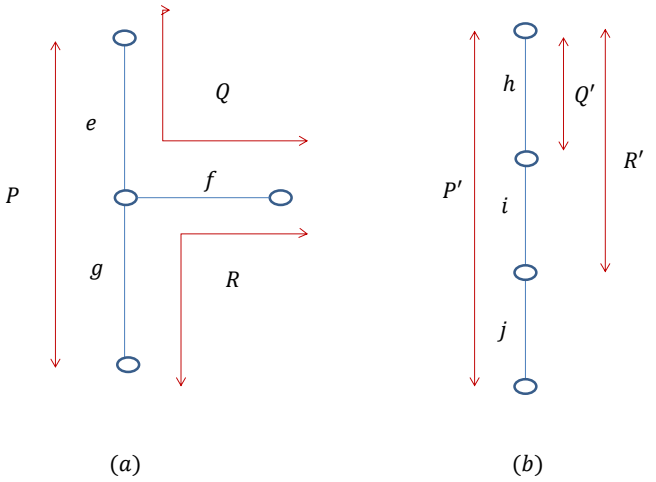


Figure 7.3: The shortest paths colored red in the graph (a) contain at most two edges in a simple star graph. The graph (b) is a graph containing one longest shortest path.

is adjacent to the path P and belongs to the ladder L_P, and the edge g is adjacent to the path Q and belongs to the ladder L_Q. The Bloom filter of the edges contain the bits 1 in the bit positions corresponding to the number of ladders. Therefore, the Bloom filters of the edge e, f and g are 001,100 and 010, respectively.

Suppose the edge f is a false positive of the Bloom filter of the path P. The path P consists of the edges e and g, hence the Bloom filter of the path P is 011 which is obtained by applying bitwise OR operation to the Bloom filters of the edges e and g on the path P. Yet, when the Bloom filter of the path P contains the bit 0 in the first bit position, there is the bit 1 in the same bit position of the Bloom filter of the edge f. Hence, the edge f is obviously neither an edge of the path P or a false positive.

Consider another graph contains only one longest shortest path which represents a line like in the Figure 7.3 (b). The number of edges in this simple graph is 3. The longest shortest path P' contains the edges h, i and j. There is no adjacent edges to

this longest shortest path P'. Therefore, the path P' generates only one ladder which is empty set of adjacent edges. In this case, each edge in the path P' is represented by one bit length Bloom filter which is 0.

As a shortest path in the graph (b) of the Figure 7.3, the shortest path R' consists of two edges i and h, and the edge j is the adjacent edge of the first edge of the shortest path R'. The path R' is a shortest path in a longest shortest path P'. We know that the longest shortest path P' generates one ladder which is empty set of adjacent edges. Hence, the Bloom filter of all edges h, i and j contains one bit 0. The Bloom filter of the path R' and the Bloom filter of the edge j are represented by the bit 0. Hence, we are not able to distinguish whether an edge is in the path or not. Therefore, the ladder construction does not work, if the edges are adjacent to the first or last edges of a longest shortest path.

Similarly, the edge i is the only adjacent edge to the path Q'. Yet, the ladder generated by the longest shortest path P' is empty set of adjacent edges. The Bloom filters of the edges are 0 and it is impossible to specify the edges whether in the path or not by comparing the Bloom filter of the path and edges.

7.3 Routing without False Positives

Theorem 7.3.1. *The Bloom filters of edges encoded by using ladders do not produce any false positives adjacent to the path in a graph, except edges which are adjacent to the first or last edge of a longest shortest path.*

Proof. Consider a shortest path P might be a subset of some longest shortest paths

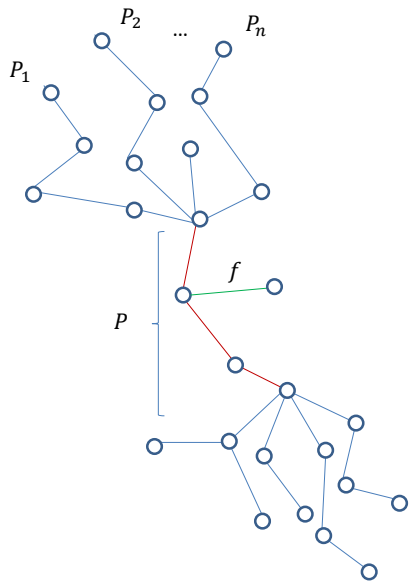


Figure 7.4: This shows a part of an arbitrary graph containing some longest shortest paths which produces ladders including an adjacent edge f to a shortest path P.

$\{P_1, \ldots, P_n\}$ where each longest shortest path links between any pair of vertices (see Figure 7.4). By definition of ladder, these n distinct longest shortest paths generate up to n different ladders such as $L = \{L_1, \ldots, L_n\}$. Recall that, all ladders are numbered by integers and the Bloom filter of an edge contains the bit 1 in the bits positions corresponding to the number of the ladder.

Suppose an edge f is adjacent to a shortest path P. Hence, the edge f belongs to $\cap_{j=1}^{n} L_j$ and the Bloom filter of the edge f contains the bit 1 in the bit positions of corresponding number of all these ladders.

Suppose the Bloom filter of the shortest path P contains the bit 1 in the bit position corresponding to some ladder L_i where $i \in [1, n]$ from the set L. The ladder L_i is produced by the longest shortest path P_i where $i \in [1, n]$ and $P \subseteq \cap_{j=1}^{n} P_j$,

namely the path P is a shortest path in the longest shortest path P_i. Hence, none of the edges in the path P is an edge of the ladder L_i, since $P \cap \cup_{j=1}^{n} L_j = \emptyset$. Therefore, the bit position corresponding to the number of the ladder L_i in the Bloom filter of the path P is 0. However, the same bit position in the Bloom filter of the edge f is 1. Therefore, the bits 1 in the Bloom filter of the edge f do not coincide to the bits 1 in the Bloom filter of the path P in any bit position corresponding to the ladders in L. The edge f is not a false positive of the Bloom filter of the shortest path P.

Note that, the edge f might belong some other ladders, say $L_{n+1}, \ldots, L_m$, rather than the ladders in L where these ladders might be generated by some other longest shortest paths, say $P_{n+1}, \ldots, P_m$, which are different from the paths $P_1, \ldots, P_n$. In this case, some edges in the path P might be the adjacent edges to the longest shortest paths $P_{n+1}, \ldots, P_m$ with the edge f. In this case the edge f and these edges in the path P belong to the same ladders $L_{n+1}, \ldots, L_m$ which are different from the ladders in the set L. Hence, the bits 1 in the Bloom filter of the path P and the Bloom filter of the edge f might coincide in some bit positions corresponding to the number of ladders $L_{n+1}, \ldots, L_m$.

Nevertheless, the Bloom filter of the path and the Bloom filter of the edge f have the bits 1 in some common bit positions, the bits 1 in the Bloom filter of the edge f do not coincide the bits 1 in the Bloom filter of the path P in all bits positions. Hence, we can conclude that the edge f is not a false positive to the Bloom filter of the path P.

□

7.4 Example: Encoding Edges with Ladders in a Rectangular Grid

In order to understand the results of the encoding method in arbitrary graphs and the parameters used for encoding edges in these graphs, we have examined this encoding method in rectangular grids. When the edges in a $M \times N$ size rectangular grid, where M and N are the number of edges on the horizontal and vertical lines, are encoded by using ladders, the length of the Bloom filters is obtained as $2\binom{M+N}{N}$ where $\binom{M+N}{N}$ is the number of longest shortest paths between two nodes placed on two opposite corners of a rectangular grid. There are $M + N$ moves from one corner to other and there are M (or equivalently N) choices to get from one corner to the other, and there are 4 corners and 2 choices for the pair of corners to draw a longest shortest path in a rectangular grid (see Figure 7.5 for all longest shortest paths between the top left corner and bottom right corner, and Figure 7.6 for all possible longest shortest paths between top left corner and bottom right corner, in a rectangular grid sized 2×2).

The set of edges, which are adjacent to a longest shortest path, generates a ladder. Therefore, the adjacent edges of the paths in each graph in both Figures 7.5 and 7.6 constitute one ladder. We assume the ladders are numbered from starting the graph at the top left corner to the graph on bottom right corner in both Figures 7.5 and 7.6 in zigzag form. The Bloom filters of the edges are built corresponding to this order of the ladders (see Figure 7.7).

The length of the Bloom filter of an edge in size of 2×2 rectangular grid is obtained

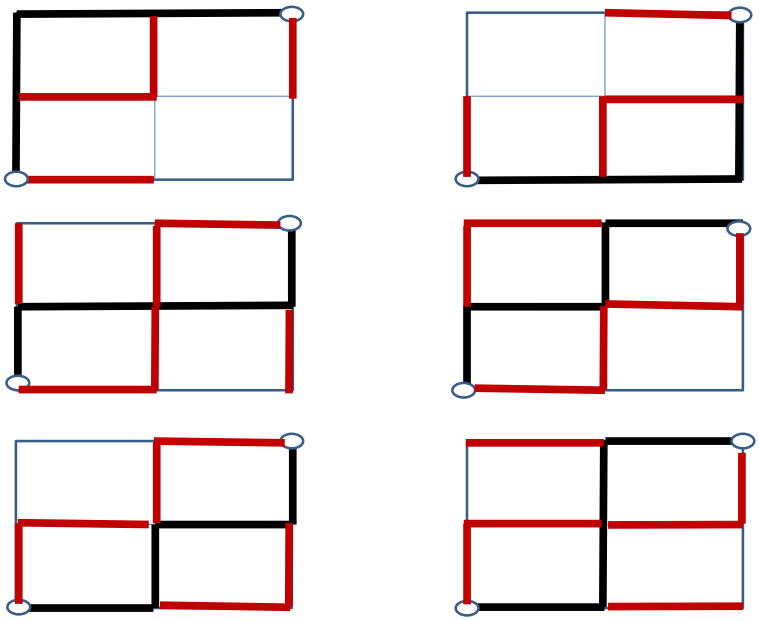


Figure 7.5: The paths all colored with black are the longest shortest paths between two vertices placed at opposite corners, and red edges are the adjacent edges to these paths.

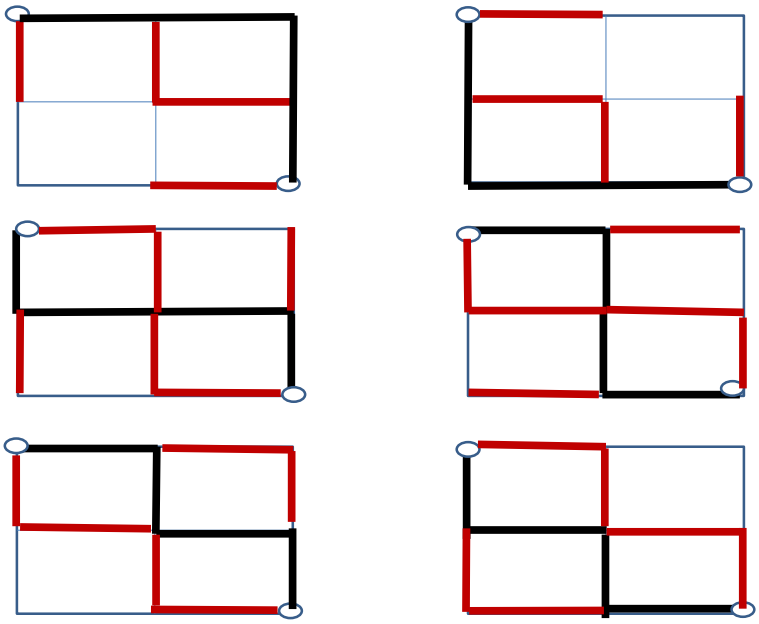
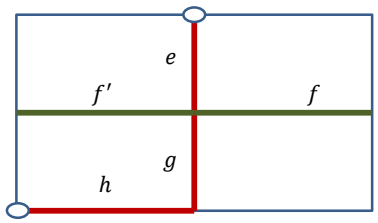


Figure 7.6: The paths all colored with black are the longest shortest paths between two vertices placed at opposite corners, and red edges are the adjacent edges to these paths.



β(*e*)	1	0	1	0	1	0	1	0	1	0	0	1
β(*g*)	0	1	1	1	0	0	0	1	1	0	1	0
β(*h*)	1	0	1	1	0	0	0	0	0	1	0	1
β(*P*)	1	1	1	1	1	0	1	1	1	1	1	1
β(*f*)	0	1	0	1	0	1	1	0	0	1	0	1
β(*f′*)	1	0	0	0	1	1	0	1	0	1	1	0

Figure 7.7: The red path is a possible shortest path between two distinct nodes and the green edges denoted by f and f' are the adjacent edges to this shortest path.

as $2\binom{M+N}{N} = 2 \times 6 = 12$. This is the total number of ladders that are given in both Figures 7.5 and 7.6. For comparison, the Bloom filter of an edge in a rectangular grid has been encoded by $4(M+N) = 4(2+2) = 16$ bits in Chapter 2, when another encoding method that we invented has been used. However, the encoding method using ladders yields better result for the small size of rectangular grid, it is obvious that $2\binom{M+N}{N} \geq 4(M+N)$ for bigger rectangular grids where their size is at least 3×2 or 2×3. Even though the encoding method we introduced in this chapter requires more space than the other method we introduced in Chapter 2, this encoding method might be useful for some other graphs.

Now, we are able to examine how the encoding method works on a small rectangular grid sized 2×2. In the rectangular grid in the Figure 7.7, there is a shortest

path colored in red and the possible false positives f and f' which are adjacent to red path. The Bloom filters of the edges are found by using the ladders given with the Figures 7.5 and 7.6. As seen in the Figure 7.7, the Bloom filters of edges have different numbers of bit 1. Besides, when the 6th bit position of the Bloom filter of the red path is 0, the same bit position of the Bloom filters of both adjacent edges f and f' to the red path are the bit 1. Therefore, these two edges are not on the path and they are not false positives of the Bloom filter of the path.

BIBLIOGRAPHY

[1] Imad Aad, Claude Castelluccia, and Jean-Pierre Huubaux. Packet coding for strong anonymity in ad hoc networks. In *Securecomm and Workshops, 2006*, pages 1–10. IEEE, 2006.

[2] Ian F Akyildiz, Joseph SM Ho, and Yi-Bing Lin. Movement-based location update and selective paging for pcs networks. *IEEE/ACM Transactions on Networking (TON)*, 4(4):629–638, 1996.

[3] Bader Albader, Bella Bose, and Mary Flahive. Efficient communication algorithms in hexagonal mesh interconnection networks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(1):69–77, 2012.

[4] Romas Aleliunas and Arnold L Rosenberg. On embedding rectangular grids in square grids. *IEEE Transactions on Computers*, 100(9):907–913, 1982.

[5] AM Awwad, A Al-Ayyoub, and Mohamed Ould-Khaoua. On the topological properties of the arrangement–star network. *Journal of Systems Architecture*, 48(11):325–336, 2003.

[6] Steven M Bellovin and William R Cheswick. Privacy-enhanced searches using encrypted bloom filters. *IACR Cryptology ePrint Archive*, 2004:22, 2004.

[7] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[8] Flavio Bonomi, Michael Mitzenmacher, Rina Panigrahy, Sushil Singh, and George Varghese. An improved construction for counting bloom filters. In *Algorithms–ESA 2006*, pages 684–695. Springer, 2006.

[9] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang. On the false positive rate of bloom filters. *Information Processing Letters*, 108(4):210–213, 2008.

[10] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.

[11] Jean Carle and Jean Frédéric Myoupo. Topological properties and optimal routing algorithms for three dimensional hexagonal networks. In *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, volume 1, pages 116–121. IEEE, 2000.

[12] Laura Carrea, Alexei Vernitski, and Martin Reed. Yes-no bloom filter: A way of representing sets with fewer false positives. *arXiv preprint arXiv:1603.01060*, 2016.

[13] Vernitski A. Reed M. Carrea, L. Optimized hash for network path encoding with minimized false positives. *Computer networks*, 58:180–191, 2014.

[14] Eric YK Chan, HW Chan, KM Chan, Vivien PS Chan, Samuel T Chanson, Matthew MH Cheung, CF Chong, KP Chow, Albert KT Hui, Lucas CK Hui, et al. Idr: an intrusion detection router for defending against distributed denial-of-service (ddos) attacks. In *Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on*, pages 581–586. IEEE, 2004.

[15] Ming-Syan Chen, Kang G Shin, and Dilip D Kandlur. Addressing, routing, and broadcasting in hexagonal mesh multiprocessors. *Computers, IEEE Transactions on*, 39(1):10–18, 1990.

[16] Ken Christensen, Allen Roginsky, and Miguel Jimeno. A new analysis of the false positive rate of a bloom filter. *Information Processing Letters*, 110:944–949, 2010.

[17] Luca Davoli, Luca Veltri, Pier Luigi Ventre, Giuseppe Siracusano, and Stefano Salsano. Traffic engineering with segment routing: Sdn-based architectural design and open source implementation. In *Software Defined Networks (EWSDN), 2015 Fourth European Workshop on*, pages 111–112. IEEE, 2015.

[18] Khaled Day and Anand Tripathi. Arrangement graphs: a class of generalized star graphs. *Information Processing Letters*, 42(5):235–241, 1992.

[19] Khaled Day and Anand Tripathi. A comparative study of topological properties of hypercubes and star graphs. *IEEE Transactions on Parallel and Distributed Systems*, 5(1):31–38, 1994.

[20] Catherine Decayeux and David Seme. 3d hexagonal network: Modeling, topological properties, addressing scheme, and optimal routing algorithm. *Parallel and Distributed Systems, IEEE Transactions on*, 16(9):875–884, 2005.

[21] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, and John Lockwood. Deep packet inspection using parallel bloom filters. In *High Performance Interconnects, 2003. Proceedings. 11th Symposium on*, pages 44–51. IEEE, 2003.

[22] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking (TON)*, 8(3):281–293, 2000.

[23] Odile Favaron, Gerd H Fricke, Dan Pritikin, and Joël Puech. Irredundance and domination in kings graphs. *Discrete Mathematics*, 262(1):131–147, 2003.

[24] Frank Harary, Robert A Melter, and Ioan Tomescu. Digital metrics: A graph-theoretical approach. *Pattern recognition letters*, 2(3):159–163, 1984.

[25] Xiangjian He and Wenjing Jia. Hexagonal structure for intelligent vision. In *Information and Communication Technologies, 2005. ICICT 2005. First International Conference on*, pages 52–64. IEEE, 2005.

[26] Eugen J Ionascu, Dan Pritikin, and Stephen E Wright. k-dependence and domination in kings graphs. *American Mathematical Monthly*, 115(9):820–836, 2008.

[27] Petri Jokela, András Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander. Lipsin: line speed publish/subscribe inter-networking. *ACM SIGCOMM Computer Communication Review*, 39(4):195–206, 2009.

[28] David Joyner, Minh Van Nguyen, and Nathann Cohen. Algorithmic graph theory. *Google Code*, 2010.

[29] Gokce C. Kayaturan and Alexei Vernitski. A way of eliminating errors when using bloom filters for routing in computer networks. In *Networks, ICN 2016. The Fifteenth International Conference on*, pages 52–57. IARIA, 2016.

[30] Gökçe Çaylak Kayaturan and Alexei Vernitski. Routing in hexagonal computer networks: How to present paths by bloom filters without false positives. In *Computer Science and Electronic Engineering (CEEC), 2016 8th*, pages 95–100. IEEE, 2016.

[31] Gokce Caylak Kayaturan and Alexei Vernitski. Encoding shortest paths in triangular grids for delivery without errors. In *Proceedings of the International Conference on Future Networks and Distributed Systems*, page 7. ACM, 2017.

[32] Gokce Caylak Kayaturan and Alexei Vernitski. Encoding shortest paths in graphs assuming the code is queried using bit-wise comparison. Arxiv eprint arXiv:1806.09442, 2018.

[33] Sukwon Kim, Michelle Effros, and Tracey Ho. On low-power multiple unicast network coding over a wireless triangular grid. In *Proceedings of the 45th Annual Allerton Conference on Communication, Control and Computing*. Citeseer, 2007.

[34] Adam Kirsch and Michael Mitzenmacher. Less hashing, same performance: Building a better bloom filter. In *Algorithms–ESA 2006*, pages 456–467. Springer, 2006.

[35] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29(9):1765 –1775, october 2011.

[36] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[37] David M. Lane. Confidence intervals introduction, 2014. *URL: http://onlinestatbook.com/2/estimation/confidence.html*, last accessed 2017-10-29.

[38] Dan Li, Yuanjie Li, Jianping Wu, Sen Su, and Jiangwei Yu. Esm: efficient and scalable data center multicast routing. *IEEE/ACM Transactions on Networking (TON)*, 20(3):944–955, 2012.

[39] Dan Li, Jiangwei Yu, Junbiao Yu, and Jianping Wu. Exploring efficient and scalable multicast routing in future data center networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1368–1376. IEEE, 2011.

[40] Xueming Li, Lijuan Peng, and Chunlin Zhang. Application of bloom filter in grid information service. In *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, pages 866–870. IEEE, 2010.

[41] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20, 2003.

[42] Yi Lu, Balaji Prabhakar, and Flavio Bonomi. Perfect hashing for network applications. In *Information Theory, 2006 IEEE International Symposium on*, pages 2774–2778. IEEE, 2006.

[43] Nicolas Maillet, Guillaume Collet, Thomas Vannier, Dominique Lavenier, and Pierre Peterlongo. Commet: comparing and combining multiple metagenomic datasets. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 94–98. IEEE, 2014.

[44] Martin Mauve, Jorg Widmer, and Hannes Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, 2001.

[45] Loizos Michael, Wolfgang Nejdl, Odysseas Papapetrou, and Wolf Siberski. Improving distributed join efficiency with extended bloom filter operations. In *Advanced Information Networking and Applications, 2007. AINA'07. 21st International Conference on*, pages 187–194. IEEE, 2007.

[46] Michael Mitzenmacher. Compressed bloom filters. *IEEE/ACM Transactions on Networking (TON)*, 10(5):604–612, 2002.

[47] Michael Mitzenmacher and George Varghese. Biff (bloom filter) codes: Fast error correction for large data sets. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 483–487. IEEE, 2012.

[48] James K. Mullin. A second look at bloom filters. *Communications of the ACM*, 26(8):570–571, 1983.

[49] Benedek Nagy. Finding shortest path with neighbourhood sequences in triangular grids. In *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, pages 55–60. IEEE, 2001.

[50] Benedek Nagy. Cellular topology and topological coordinate systems on the hexagonal and on the triangular grids. *Annals of Mathematics and Artificial Intelligence*, 75(1-2):117–134, 2015.

[51] Benedek N Nagy. Shortest paths in triangular grids with neighbourhood sequences. *CIT. Journal of computing and information technology*, 11(2):111–122, 2003.

[52] Fabian Garcia Nocetti, Ivan Stojmenovic, and Jingyuan Zhang. Addressing and routing in hexagonal networks with applications for tracking mobile users and connection rerouting in cellular networks. *Parallel and Distributed Systems, IEEE Transactions on*, 13(9):963–971, 2002.

[53] M Xavier Punithan and Seung-Woo Seo. King's graph-based neighbor-vehicle mapping framework. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1313–1330, 2013.

[54] Andreas Reinhardt, Olivia Morar, Silvia Santini, Sebastian Zöller, and Ralf Steinmetz. Cbfr: Bloom filter routing with gradual forgetting for tree-structured wireless sensor networks with mobile nodes. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–9. IEEE, 2012.

[55] Sean C Rhea and John Kubiatowicz. Probabilistic location and routing. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1248–1257. IEEE, 2002.

[56] Joshua Robinson and Edward W Knightly. A performance study of deployment factors in wireless mesh networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2054–2062. IEEE, 2007.

[57] Christian Esteve Rothenberg, Carlos Alberto Braz Macapuna, Maurício Ferreira Magalhães, Fábio Luciano Verdi, and Alexander Wiesmaier. In-packet bloom filters: Design and networking applications. *Computer Networks*, 55(6):1364–1378, 2011.

[58] Mikko Särelä, Christian Esteve Rothenberg, Tuomas Aura, András Zahemszky, Pekka Nikander, and Jörg Ott. Forwarding anomalies in bloom filter-based multicast. In *INFOCOM, 2011 Proceedings IEEE*, pages 2399–2407. IEEE, 2011.

[59] Haoyu Song, Fang Hao, Murali Kodialam, and TV Lakshman. Ipv6 lookups using distributed and load balanced bloom filters for 100gbps core router line

cards. In *INFOCOM 2009, IEEE*, pages 2518–2526. IEEE, 2009.

[60] Ivan Stojmenovic. Honeycomb network. *Proc. Math. Foundations of Computer Science MFCS '95, Lecture Notes in Computer Science*, 969:267–276, 1995.

[61] Kamala Subramaniam, Alan L Tharp, and Arne A Nilsson. Location updates in cellular networks using bloom filters. In *Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International*, volume 2, pages 3–9. IEEE, 2006.

[62] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2012.

[63] Yi-Min Wang, Lili Qiu, Dimitris Achlioptas, Gautam Das, Paul Larson, and Helen J Wang. Subscription partitioning and routing in content-based publish/subscribe systems. In *16th International Symposium on DIStributed Computing (DISC'02)*, 2002.

[64] Tilman Wolf. A credential-based data path architecture for assurable global networking. In *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pages 1–7. IEEE, 2007.

[65] Peter Yap. Grid-based path-finding. In *Advances in Artificial Intelligence*, pages 44–55. Springer, 2002.

[66] Minlan Yu, Alex Fabrikant, and Jennifer Rexford. Buffalo: Bloom filter forwarding architecture for large organizations. In *Proceedings of the 5th international*

conference on Emerging networking experiments and technologies, pages 313–324. ACM, 2009.

[67] Ellen W Zegura, Kenneth L Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, volume 2, pages 594–602. IEEE, 1996.

[68] Bin Zhao, Weining Qian, and Aoying Zhou. Towards bipartite graph data management. In *Proceedings of the second international workshop on Cloud data management*, pages 55–62. ACM, 2010.

[69] Yun Zhou, Yanchao Zhang, and Yuguang Fang. Key establishment in sensor networks based on triangle grid deployment model. In *MILCOM*, volume 3, page 1450, 2005.