AI

CSPC53-Computer Networks

Theory - Project Report

Group 17

Roll Numbers

106119112
106119100

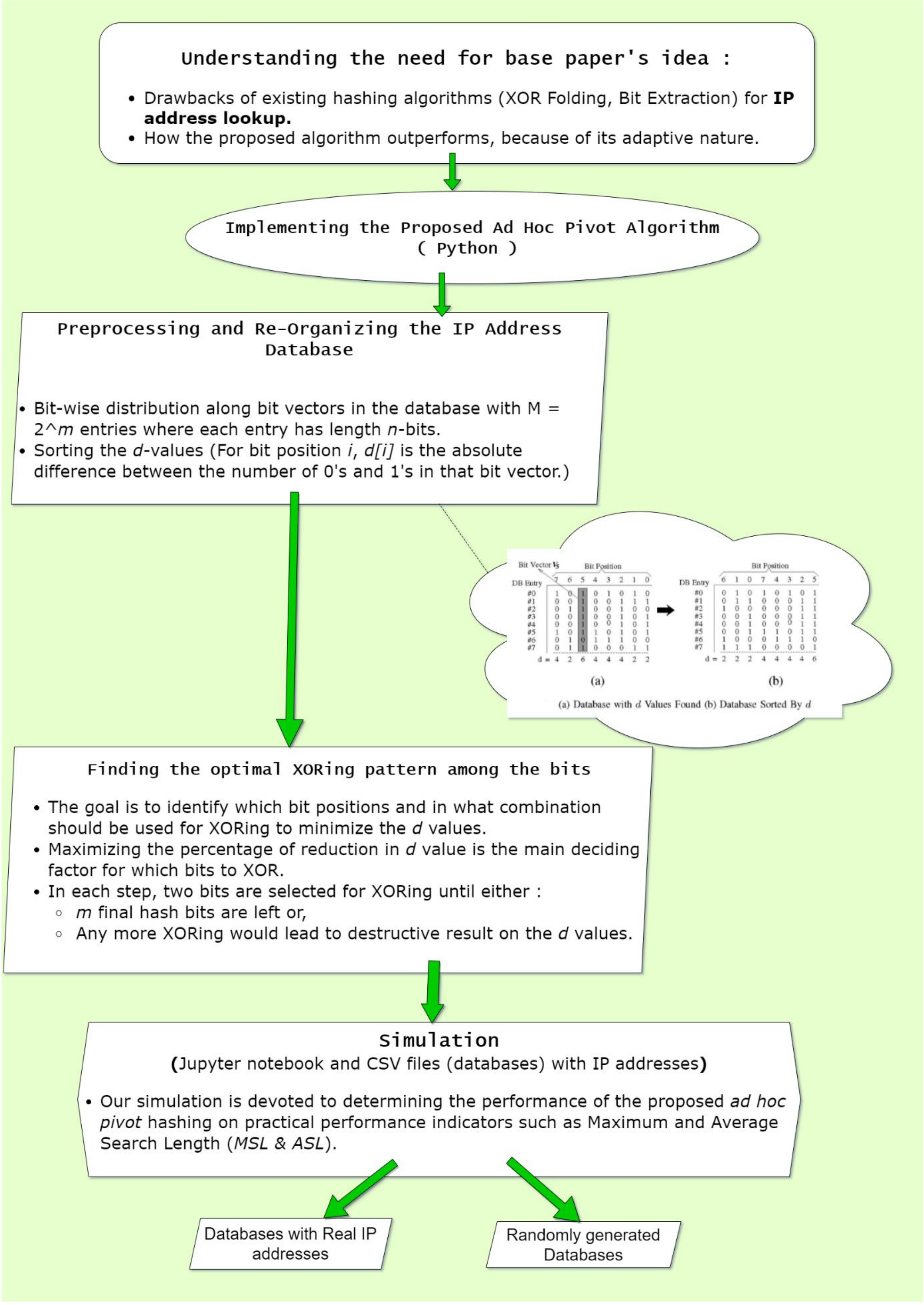
Group Members

SATYARTH PANDEY
RAJNEESH PANDEY

Section

CSE-B

Workflow Diagram



Understanding the need for base paper's idea :

- Drawbacks of existing hashing algorithms (XOR Folding, Bit Extraction) for **IP address lookup.**
- How the proposed algorithm outperforms, because of its adaptive nature.

Implementing the Proposed Ad Hoc Pivot Algorithm (Python)

Preprocessing and Re-Organizing the IP Address Database

- Bit-wise distribution along bit vectors in the database with M = 2^*m* entries where each entry has length *n*-bits.
- Sorting the *d*-values (For bit position *i*, *d[i]* is the absolute difference between the number of 0's and 1's in that bit vector.)

DB Entry	7	6	5	4	3	2	1	0
#0	1	0	1	0	1	0	1	0
#1	0	0	1	0	0	1	1	1
#2	0	1	1	0	0	1	0	0
#3	0	0	1	0	0	1	0	1
#4	0	0	1	0	0	1	0	1
#5	1	0	1	1	0	1	0	1
#6	0	1	0	1	1	1	0	0
#7	0	1	1	0	0	0	1	1
d =	4	2	6	4	4	4	2	2

(a)

DB Entry	6	1	0	7	4	3	2	5
#0	0	1	0	1	0	1	0	1
#1	0	1	1	0	0	0	1	1
#2	1	0	0	0	0	0	1	1
#3	0	0	1	0	0	0	1	1
#4	0	0	1	0	0	0	1	1
#5	0	0	1	1	1	0	1	1
#6	1	0	0	0	1	1	1	0
#7	1	1	1	0	0	0	0	1
d =	2	2	2	4	4	4	4	6

(b)

(a) Database with *d* Values Found (b) Database Sorted By *d*

Finding the optimal XORing pattern among the bits

- The goal is to identify which bit positions and in what combination should be used for XORing to minimize the *d* values.
- Maximizing the percentage of reduction in *d* value is the main deciding factor for which bits to XOR.
- In each step, two bits are selected for XORing until either :
  - *m* final hash bits are left or,
  - Any more XORing would lead to destructive result on the *d* values.

Simulation

(Jupyter notebook and CSV files (databases) with IP addresses**)**

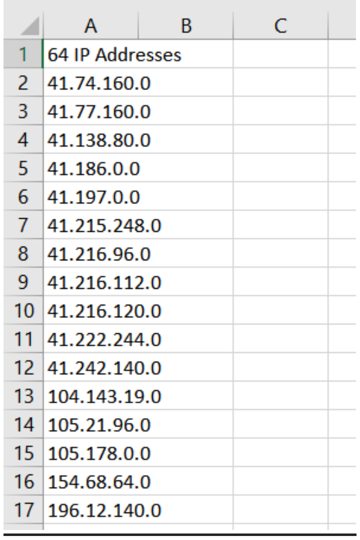
- Our simulation is devoted to determining the performance of the proposed *ad hoc pivot* hashing on practical performance indicators such as Maximum and Average Search Length (*MSL & ASL*).

Databases with Real IP addresses

Randomly generated Databases

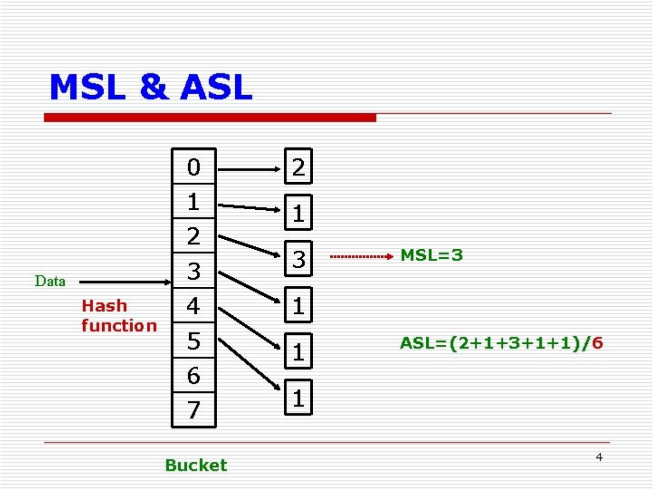
Simulation Environment

Jupyter notebook (python) and CSV files (databases) with IP addresses



	A	B	C
1	64 IP Addresses		
2	41.74.160.0		
3	41.77.160.0		
4	41.138.80.0		
5	41.186.0.0		
6	41.197.0.0		
7	41.215.248.0		
8	41.216.96.0		
9	41.216.112.0		
10	41.216.120.0		
11	41.222.244.0		
12	41.242.140.0		
13	104.143.19.0		
14	105.21.96.0		
15	105.178.0.0		
16	154.68.64.0		
17	196.12.140.0		

Our simulation is devoted to determining the performance of the proposed ad hoc pivot hashing on practical performance indicators. This paper looks at two important performance indicators: maximal search length (MSL) and average search length (ASL).



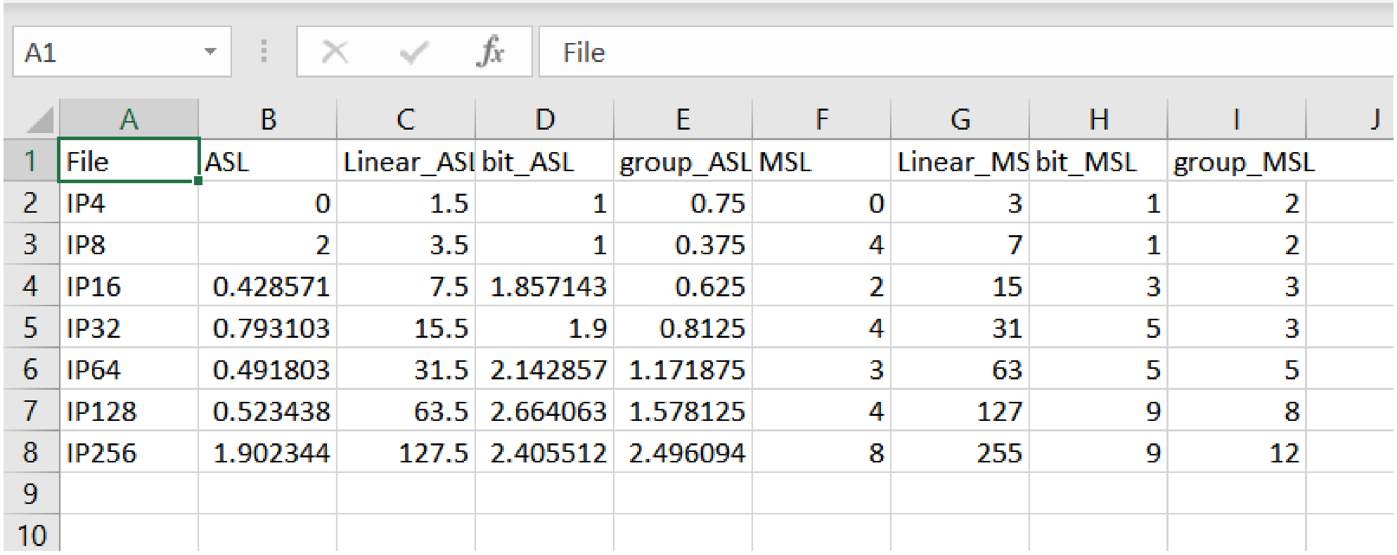
MSL & ASL

The above fig shows a graphical example of the performance indicators. The indicator MSL denotes the maximum number of hash collisions which in turn indicates the maximal number of search steps required to search through the collision. ASL reflects the average number of lookups needed to find an item in the database.

Results (Table, Graph)

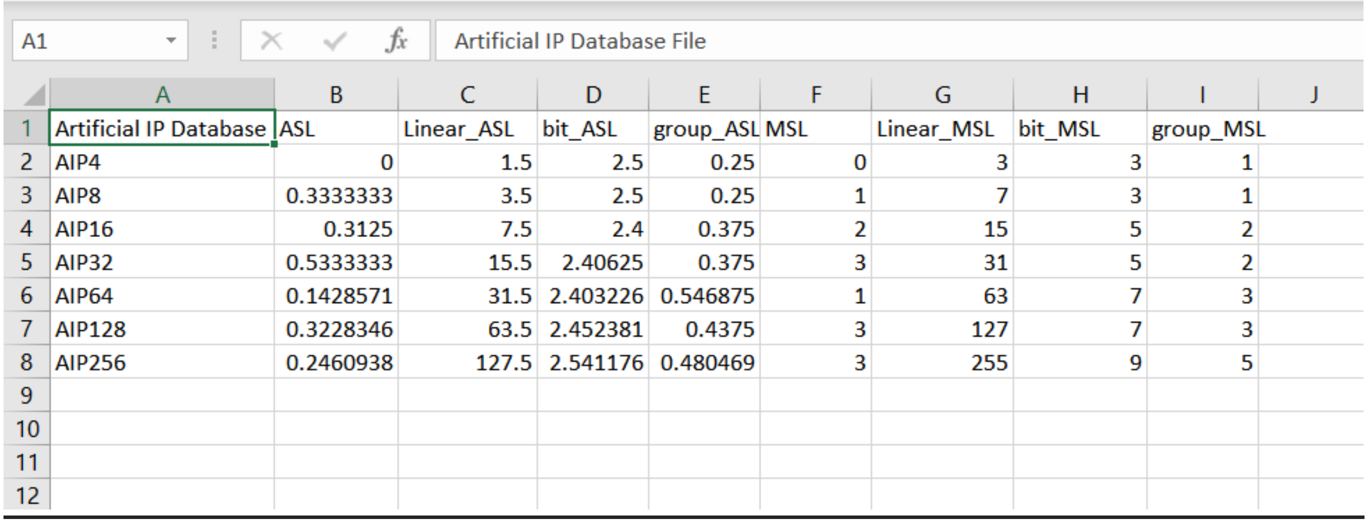
Comparison of MSL and ASL values between proposed and existing Algorithms on across multiple databases on :

1) *Real IP Addresses :*



	A	B	C	D	E	F	G	H	I	J
1	File	ASL	Linear_ASL	bit_ASL	group_ASL	MSL	Linear_MS	bit_MSL	group_MSL	
2	IP4	0	1.5	1	0.75	0	3	1	2	
3	IP8	2	3.5	1	0.375	4	7	1	2	
4	IP16	0.428571	7.5	1.857143	0.625	2	15	3	3	
5	IP32	0.793103	15.5	1.9	0.8125	4	31	5	3	
6	IP64	0.491803	31.5	2.142857	1.171875	3	63	5	5	
7	IP128	0.523438	63.5	2.664063	1.578125	4	127	9	8	
8	IP256	1.902344	127.5	2.405512	2.496094	8	255	9	12	
9										
10										

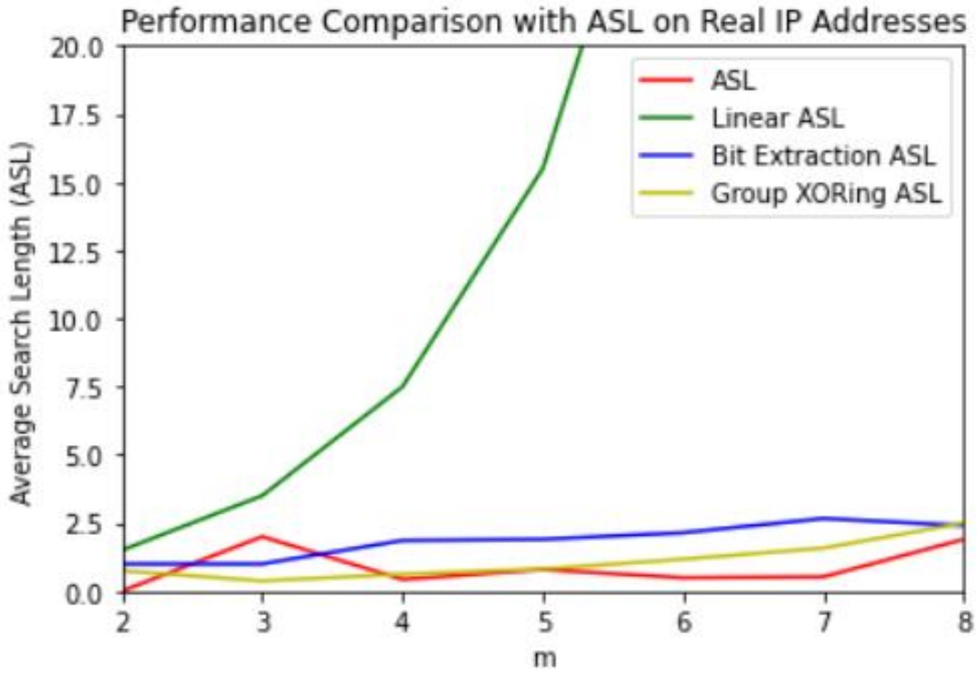
2) *Artificial IP Addresses:*

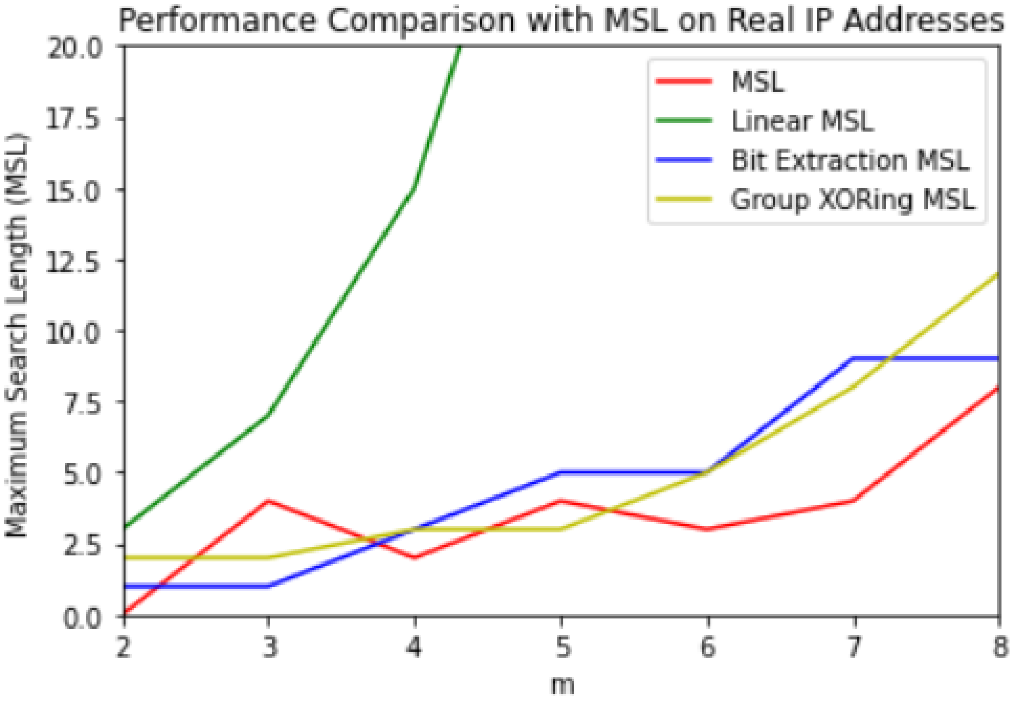


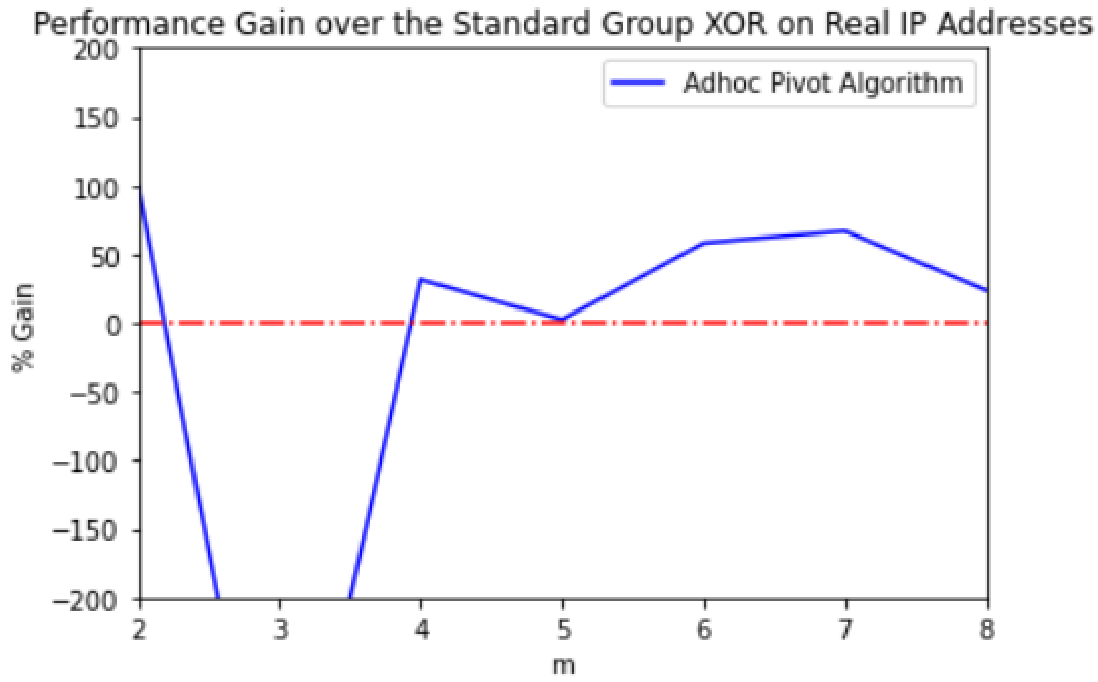
	A	B	C	D	E	F	G	H	I	J
1	Artificial IP Database	ASL	Linear_ASL	bit_ASL	group_ASL	MSL	Linear_MSL	bit_MSL	group_MSL	
2	AIP4	0	1.5	2.5	0.25	0	3	3	1	
3	AIP8	0.3333333	3.5	2.5	0.25	1	7	3	1	
4	AIP16	0.3125	7.5	2.4	0.375	2	15	5	2	
5	AIP32	0.5333333	15.5	2.40625	0.375	3	31	5	2	
6	AIP64	0.1428571	31.5	2.403226	0.546875	1	63	7	3	
7	AIP128	0.3228346	63.5	2.452381	0.4375	3	127	7	3	
8	AIP256	0.2460938	127.5	2.541176	0.480469	3	255	9	5	
9										
10										
11										
12										

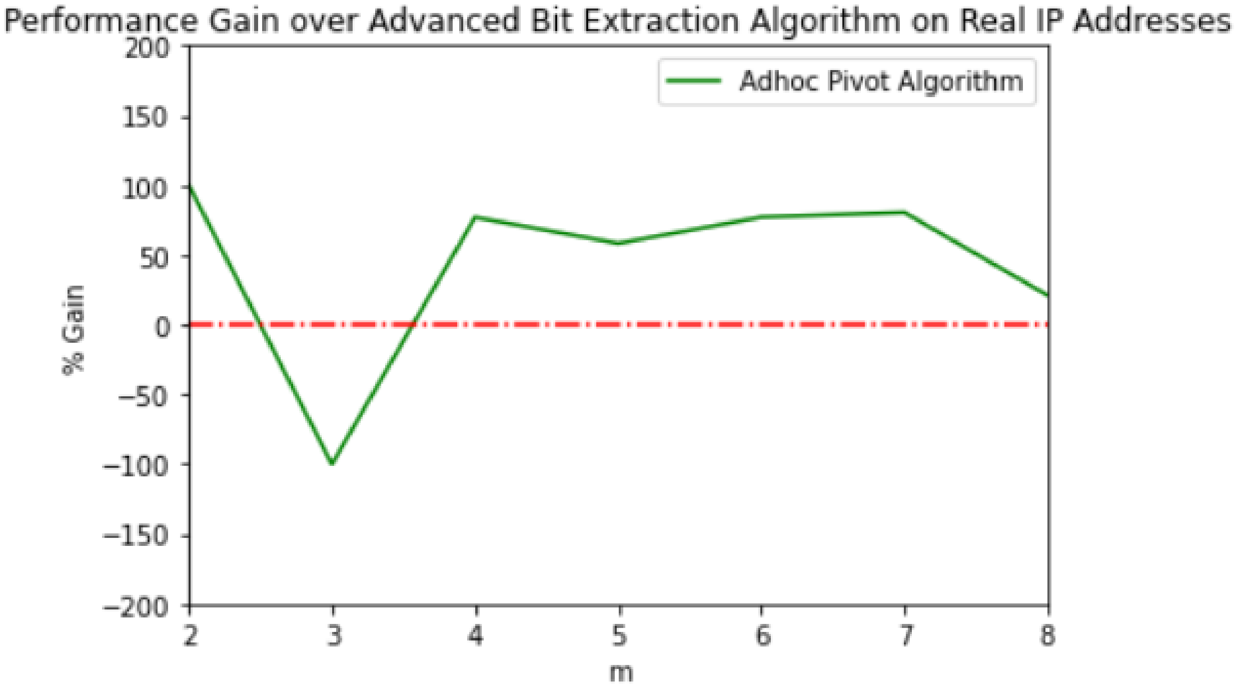
Performance Comparison Graphs with previously Well-Known Algorithms :

1) On Real IP Addresses

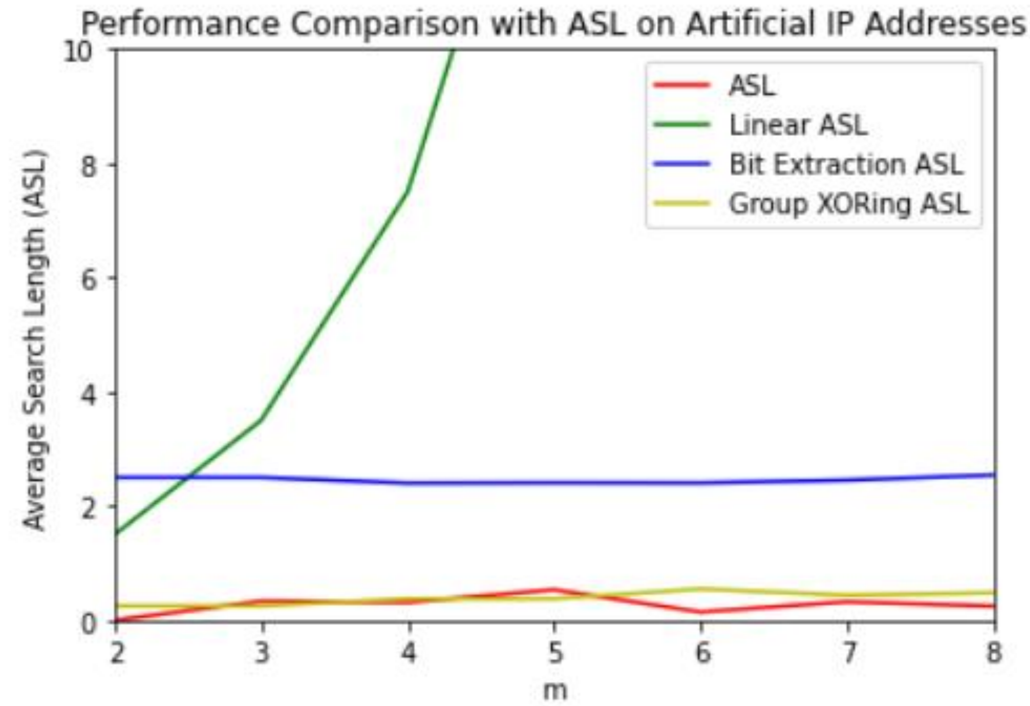


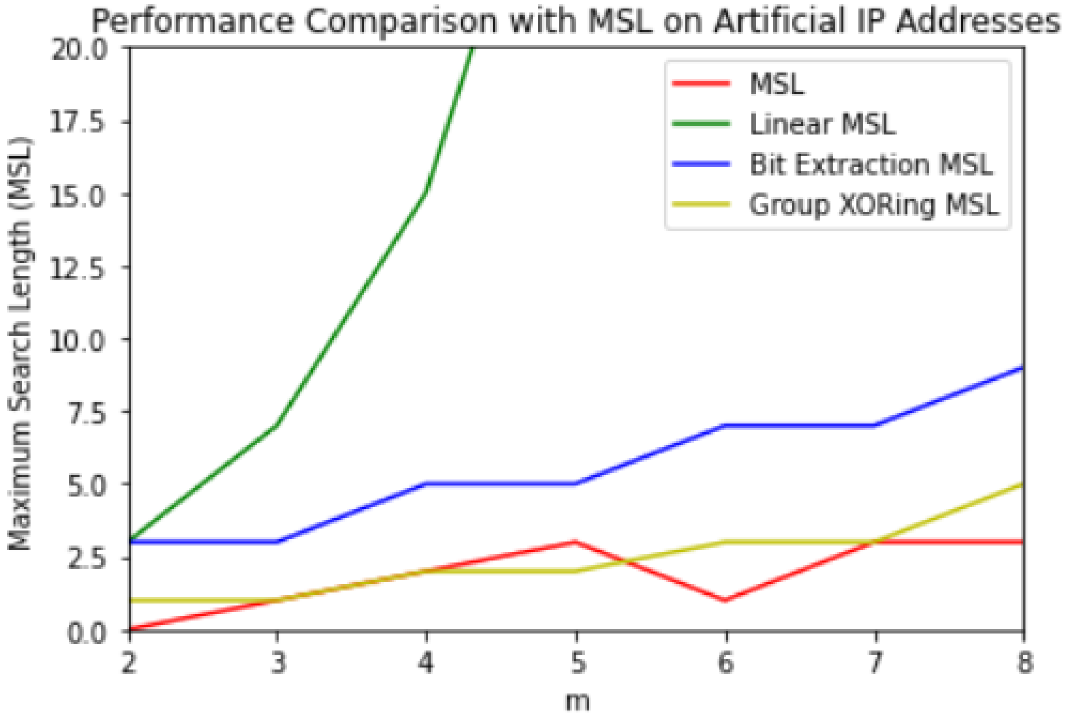


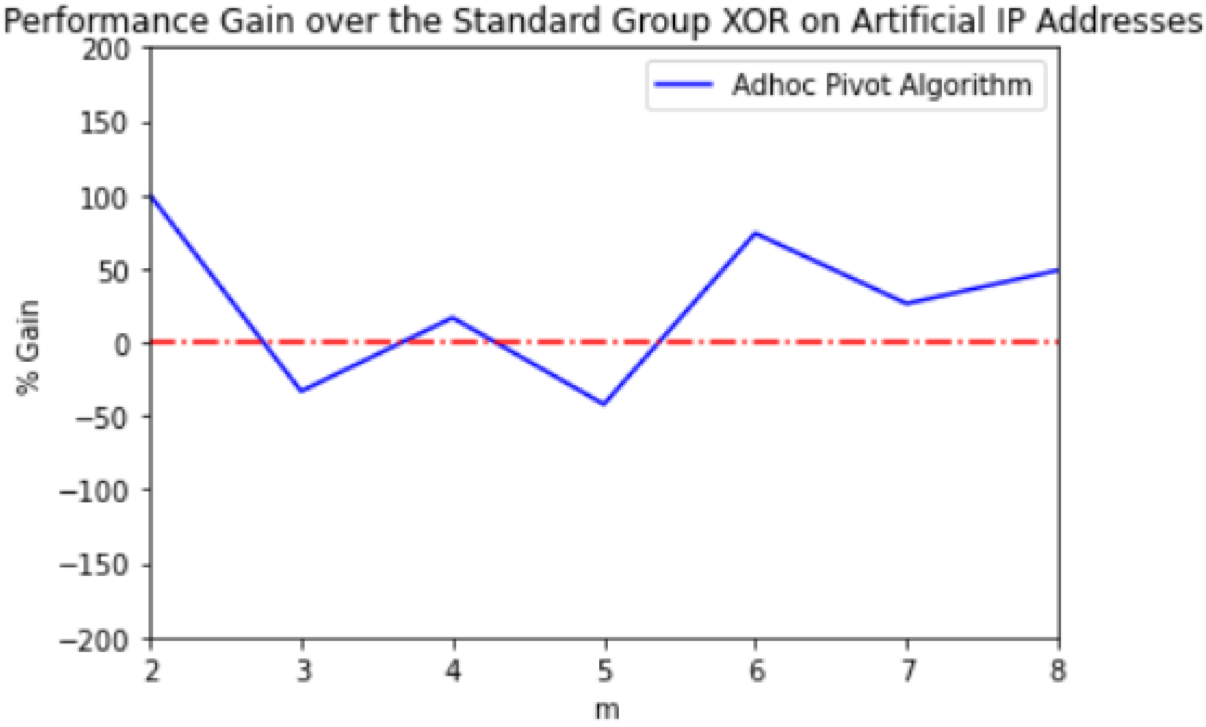

Performance Gain over the Standard Group XOR on Real IP Addresses
Adhoc Pivot Algorithm
200
150
100
50
0
−50
−100
−150
−200
% Gain
2
3
4
5
6
7
8
m

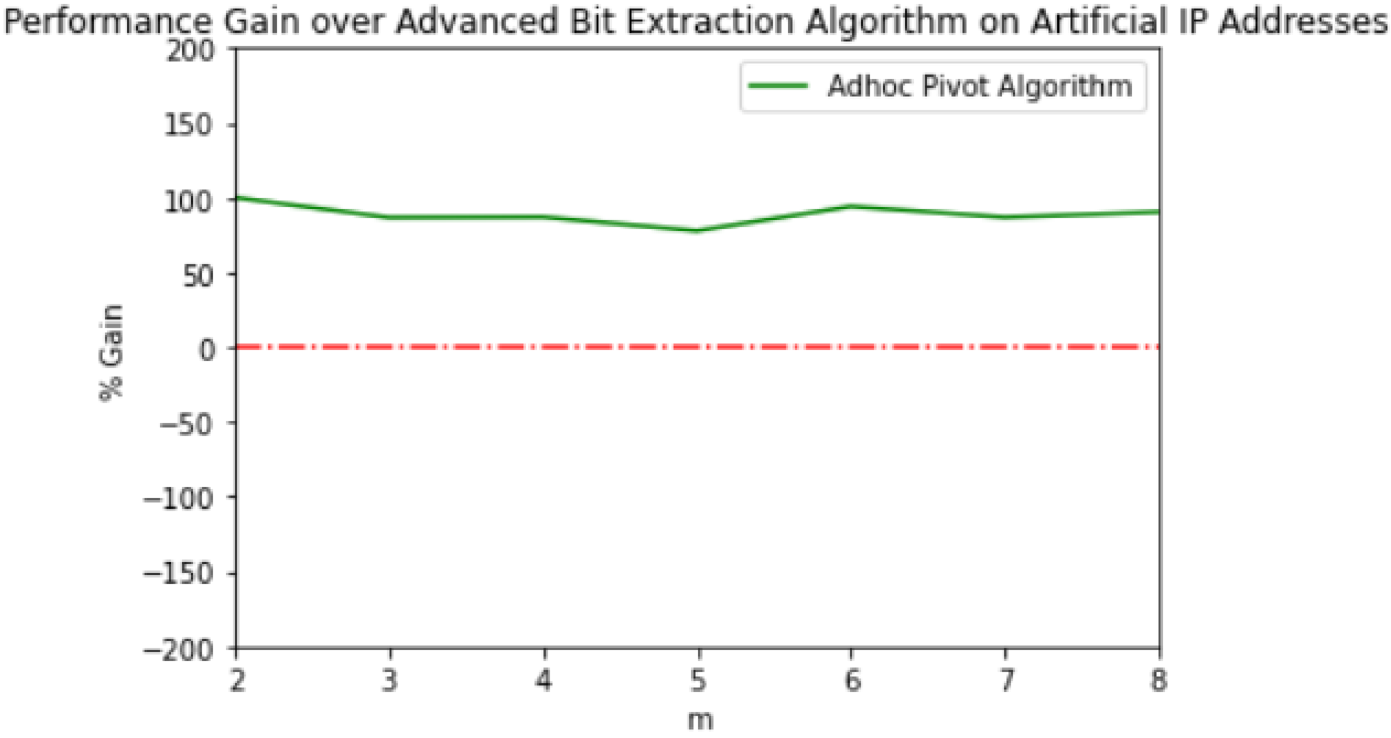

Performance Gain over Advanced Bit Extraction Algorithm on Real IP Addresses
Adhoc Pivot Algorithm
200
150
100
50
0
−50
−100
−150
−200
% Gain
2
3
4
5
6
7
8
m

2) On Artificial IP Addresses






Performance Gain over the Standard Group XOR on Artificial IP Addresses
Adhoc Pivot Algorithm
% Gain
200
150
100
50
0
−50
−100
−150
−200
2
3
4
5
6
7
8
m


Performance Gain over Advanced Bit Extraction Algorithm on Artificial IP Addresses
Adhoc Pivot Algorithm
% Gain
200
150
100
50
0
−50
−100
−150
−200
2
3
4
5
6
7
8
m

Resultant *d* value on XORing and Percentage reduction Graphs

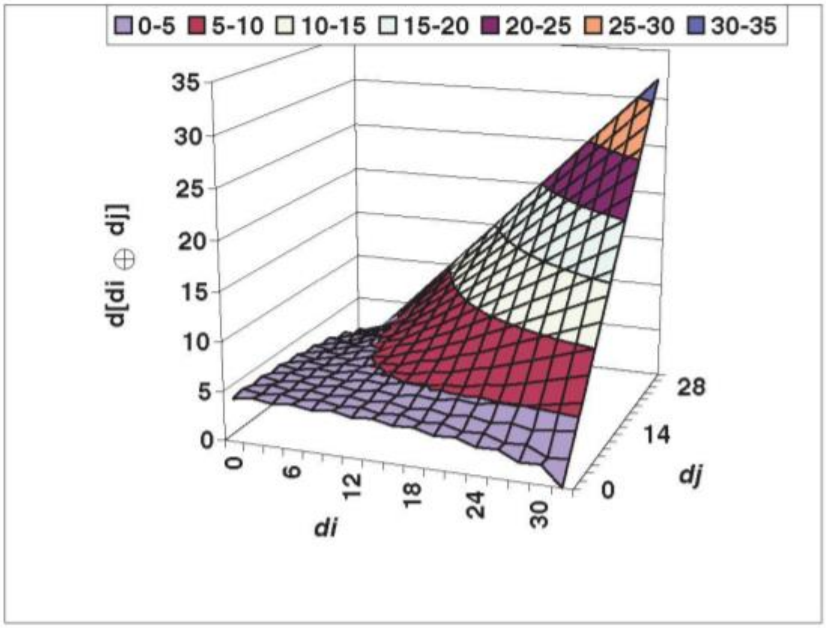


Fig. 3. Spectrum of $d_{[d_i \oplus d_j]}$ Values

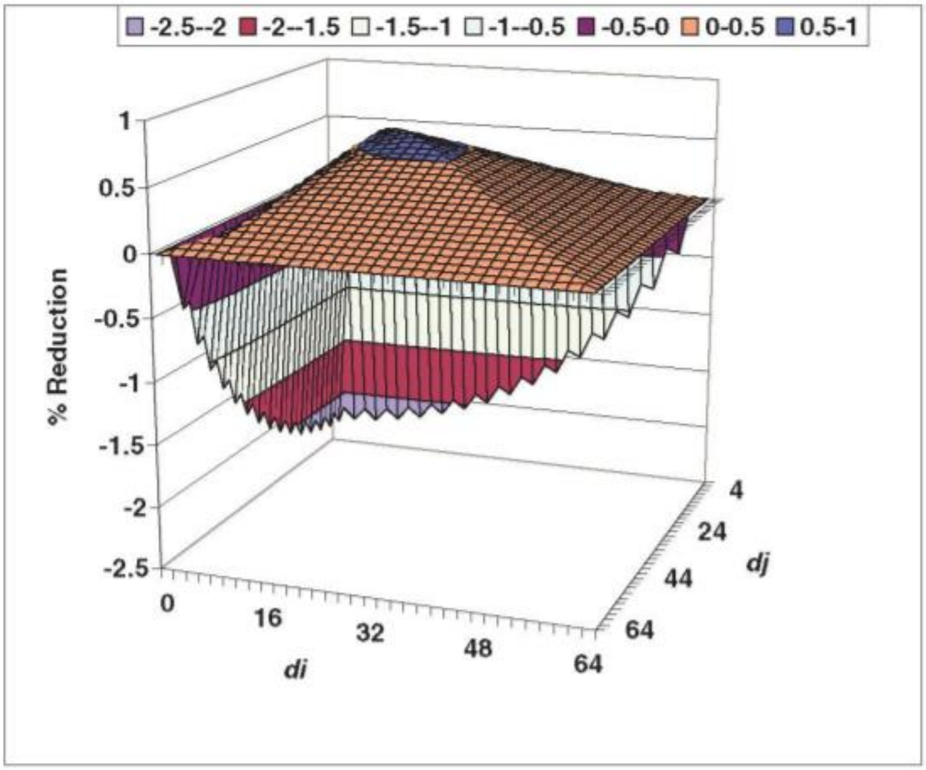


Fig. 4. Distribution of Percentage of Reduction p_{ij}

Possible Extensions

Concept Extension :

The distribution of data in a wide range of applications requires new research in hashing. Many applications such as IP address lookup, intrusion detection systems, general database query and string matching can benefit from hashing algorithms designed for an arbitrary distributed database. The proposed methodology of ad hoc pivot hashing demonstrates that improvement in overall performance can be achieved by carefully adapting to the distribution of the application. The ad hoc pivot hashing delivers several critical insights into new areas of hashing research. A potential expansion to hashing includes further exploring the database by investigating correlation among bit vectors for even better decision on how and which bits to be XORed. Other extensions include finding a non-exclusive XORing hashing in which bits are reused to further improve the search performance.

Extension on Data structure :

Instead of using Linear data structure, We can use Balanced Binary Search Tree such as Red Black Tree, AVL Tree etc. to store the Key (Hashed Value of IP Address) & Value(IP Address/Other Networking Info.) pairs .

*Base Paper :

https://ieeexplore.ieee.org/document/4087686

*Real IP Address Database source :

https://datahub.io/core/geoip2-ipv4

*Artificial IP Address Database source :

https://www.ipvoid.com/random-ip/