Design and Analysis of LSH Based Techniques for Inner Product

Venkatesh Guntakindapalli
(Under the Supervision of Dr Debajyoti Bera)

IIIT-Delhi

29-07-2017

Agenda

1. Locality Sensitive Hashing(LSH)
2. Inner product similarity(IP)
3. Why it is hard to design LSH for IP
4. Asymmetric transformations
5. LSH for IP
6. Analysis metrics
7. LSH Analysis
8. Two Level Banding
9. An Use Case- Frequent Pattern Mining
10. Conclusion

What is LSH

- Informally, an LSH $\mathcal{H}$ is a set of hash functions satisfying the following condition
  - Let h is chosen uniformly at random from $\mathcal{H}$
    - p and q are similar then $Pr[h(x) = h(q)]$ is high.

- Formally, $\mathcal{H}$ is (s_1, s_2, p_1, p_2) - sensitive LSH
  - $s_1 \geq s_2$ and $p_1 \geq p_2$
  - Let h is chosen uniformly at random from $\mathcal{H}$
    - $Sim(x, y) \geq s_1$ then $Pr_h[h(x) = h(y)] \geq p_1$
    - $Sim(x, y) \leq s_2$ then $Pr_h[h(x) = h(y)] \leq p_2$

Probability Amplification

- Increasing p_1 and decreasing p_2.
- **(K,L)-Banding:** Any two points x, q are hashed into the same location iff

$$\exists j \text{ s.t. } b_j(x) \equiv b_j(q) \text{ for } 1 \leq j \leq L,$$
$$\text{here } b_j(x) \equiv b_j(q) \text{ iff } \forall i \; h_{ij}(x) = h_{ij}(q) \text{ for } 1 \leq i \leq K$$

- The resulting probability is $1 - (1 - p^K)^L$

p	K	L	Prob
0.8	4	7	0.974
0.6	4	7	0.6215
0.3	4	7	0.0553

Table 1: Probability Amplification

LSH

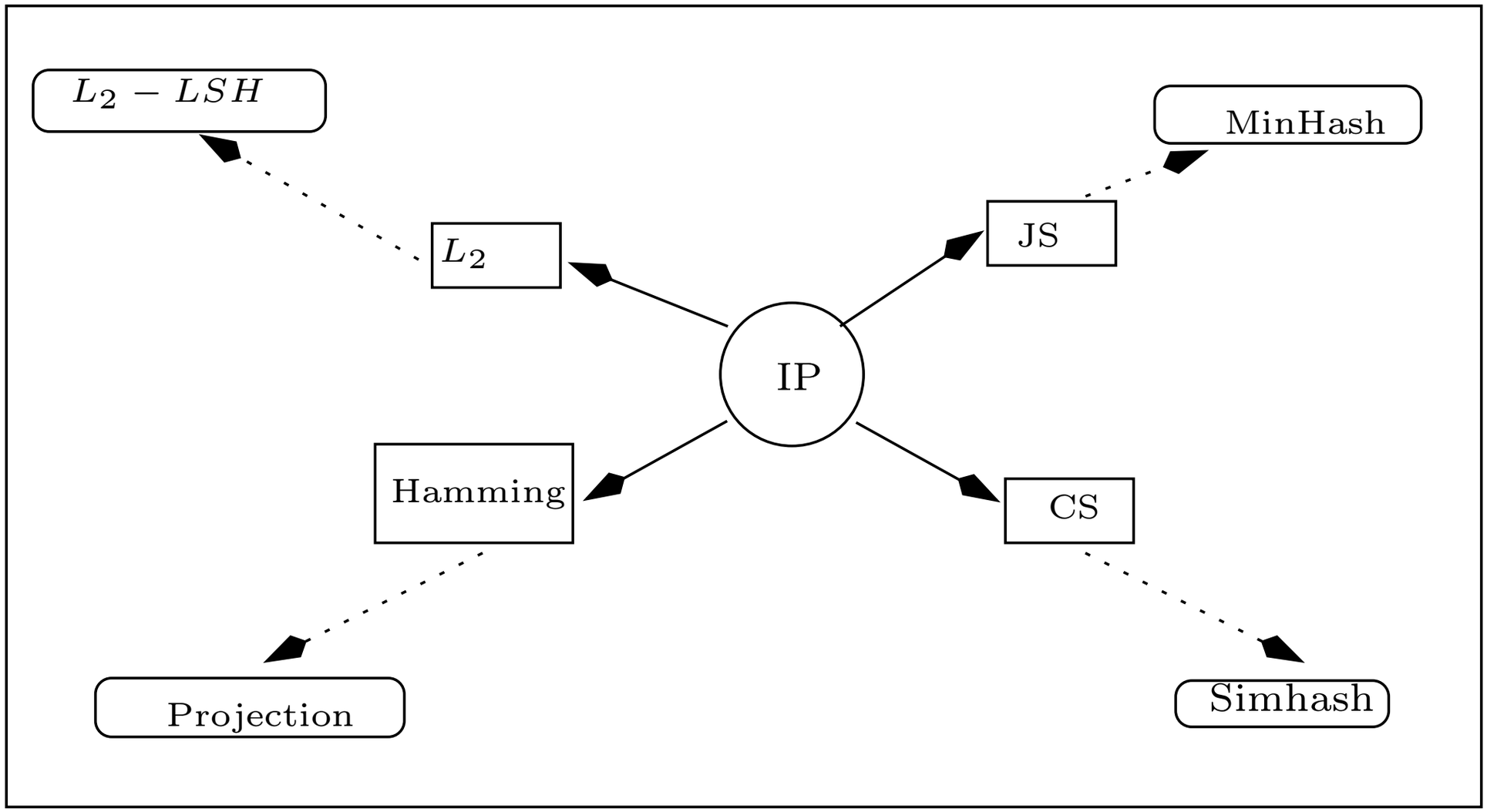

Figure 1: LSH for different similarity measures

- We mainly focus on two LSH hashing schemes: Minhash(An LSH for Jaccard similarity) and simhash(An LSH for cosine similarity).

LSH(Minhash)

- Similarity measure is Jaccard similarity

$$JS(x,y) = \frac{|x \cap y|}{|x|+|y|-|x \cap y|}$$

- Takes random permutation Π_k of input set,x.

$$minhash(x) = first\_val\_of \Pi_k(x)$$

- Computing all permutations is very expensive.
- Use some hash function that gives the same effect of random permutation.
- Minhash value : $h(x) = min\{h_k(x)\}$
- $Pr_{h \in \mathcal{H}}[h(x) = h(y)] = JS(x,y)$

LSH(Simhash)

- Similarity measure is cosine similarity $CS(x, y) = 1 - \frac{\theta}{\pi}$

$$\theta = \arccos\left(x^T * y / ||x||_2 * ||y||_2\right)$$

- The hash value is

$$h_w(x) \equiv \begin{cases} 1, & \text{if } w^T * x \geq 0. \\ 0, & \text{otherwise.} \end{cases}$$

w is a vector of length d and $w_i \sim N(0, 1)$.

- $Pr[h_w(x) = h_w(y)] = 1 - \frac{\theta}{\pi} = CS(x, y)$

Inner Product similarity(IP)

- Inner product similarity(IP) of two points(or vectors) $x, y \in \mathbb{R}^d$ is defined as

$$IP(x, y) = \Sigma_{i=1}^{d} x_i * y_i = x^T * y$$

- **Example 1:** Let x = [1,0,1,0] and y = [1,1,0,0] then IP(x, y) = 1*1+0*1+1*0+0*0 = 1

- Ubiquitous measure
  - Classification
  - Clustering
  - Information retrieval
  - Recommendation systems

A Negative Result

Lemma:

There can not exist any LSH family for inner product similarity.

Proof:

- Lets assume that h is an LSH for IP.
- $IP(x, x) = x^T * x = ||x||_2^2$.
- It is also possible to find $y \in R^d$ such that $IP(x, y) > IP(x, x)$
- As per LSH definition $Pr[h(x) = h(y)] > Pr[h(x) = h(x)]$.
- Since $Pr[h(x) = h(x)]$ =1, $Pr[h(x) = h(y)]$ can not be greater than 1.
- It contradicts the LSH definition and the assumption that h is an LSH for IP.

Asymmetric transformations

- The negative result indicates that sub-linear similarity search algorithm for inner product can not be obtained using LSH.
- We use asymmetric transformations
- **Binary-Transformation:** For $x \in D$, P(x) is defined as

$$P(x) = [x; 1; 1; 1; .....1; 1; 0; 0; .....0]$$

For query point q, Q(q) is defined as

$$Q(q) = [x; 0; 0.....0; 0]$$

- **Simple-Transformation:** For $x \in D$, P(x) is defined as

$$P(x) = [x; \sqrt{1 - ||x_i||_2^2}; 0]$$

For query point q, Q(q) is defined as

$$Q(x) = [x; 0; \sqrt{1 - ||x_i||_2^2}]$$

LSH for IP

- **Minhash:**

$$\left(\frac{\theta}{2M-\theta}, \frac{(1-\epsilon)\theta}{2M-(1-\epsilon)\theta}, \frac{\theta}{2M-\theta}, \frac{(1-\epsilon)\theta}{2M-(1-\epsilon)\theta}\right)$$

- **Simhash:**

$$\left(1-\frac{\arccos\left(\frac{\theta}{M}\right)}{\pi}, 1-\frac{\arccos\left(\frac{(1-\epsilon)\theta}{M}\right)}{\pi}, 1-\frac{\arccos\left(\frac{\theta}{M}\right)}{\pi}, 1-\frac{\arccos\left(\frac{(1-\epsilon)\theta}{M}\right)}{\pi}\right)$$

- θ - inner product threshold.
  ϵ - error margin.
  M is maximum number of non zeros in any data point.

Analysis Metrics

1. Number of hash function evaluations

2. **true positive rate(tp):** It is the fraction of points which have inner product(with query q) greater than the required threshold and LSH scheme also hash them into the same location as query point q.

3. **false positive rate(fp):**It is the fraction of points which have inner product(with query q) less than the required threshold but the LSH scheme hash them into the same location as query point q.

LSH Analysis

- How many hash functions are required to construct $(\theta, (1-\epsilon)\theta, tp, fp)$- sensitive LSH family from (s_1, s_2, p_1, p_2)-sensitive LSH family.

- The number of hash function evaluations will be K*L.

$$K = \log_{\rho}^{\delta} \text{ and } L = -\frac{\ln(1-tp)}{p_1^k}$$

$$\delta = \log_{(1-fp)}^{(1-tp)} \text{ and } \rho = \frac{p_1}{p_2}$$

p_1	p_2	tp	fp	K	L	Total
0.8	0.7	0.95	0.1	25	805	20125
0.2	0.1	0.95	0.1	4	7115	28460

Table 2: Number of Hash Functions

LSH Analysis

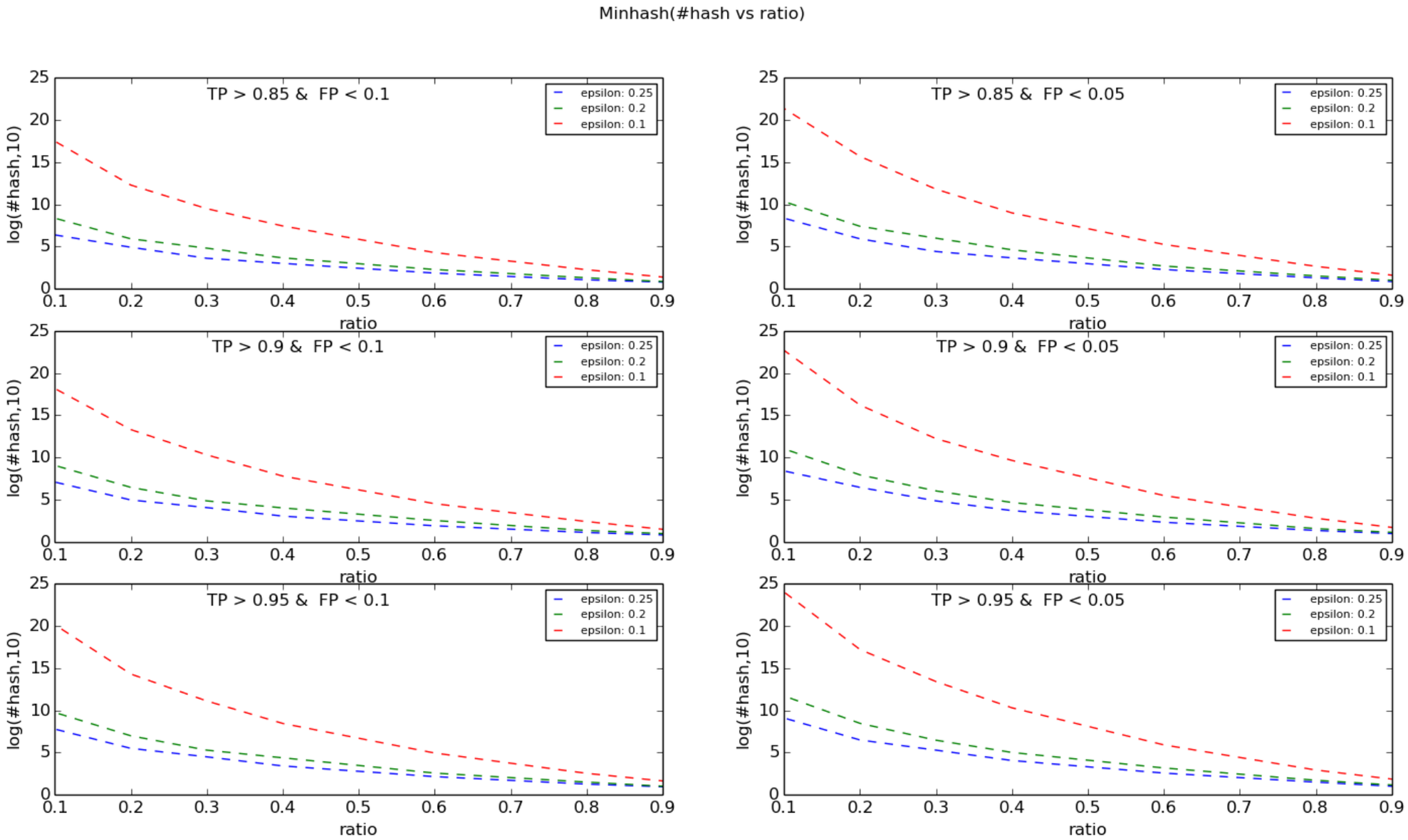

Figure 2: Minhash(no of hash vs ratio(θ/M))

LSH Analysis

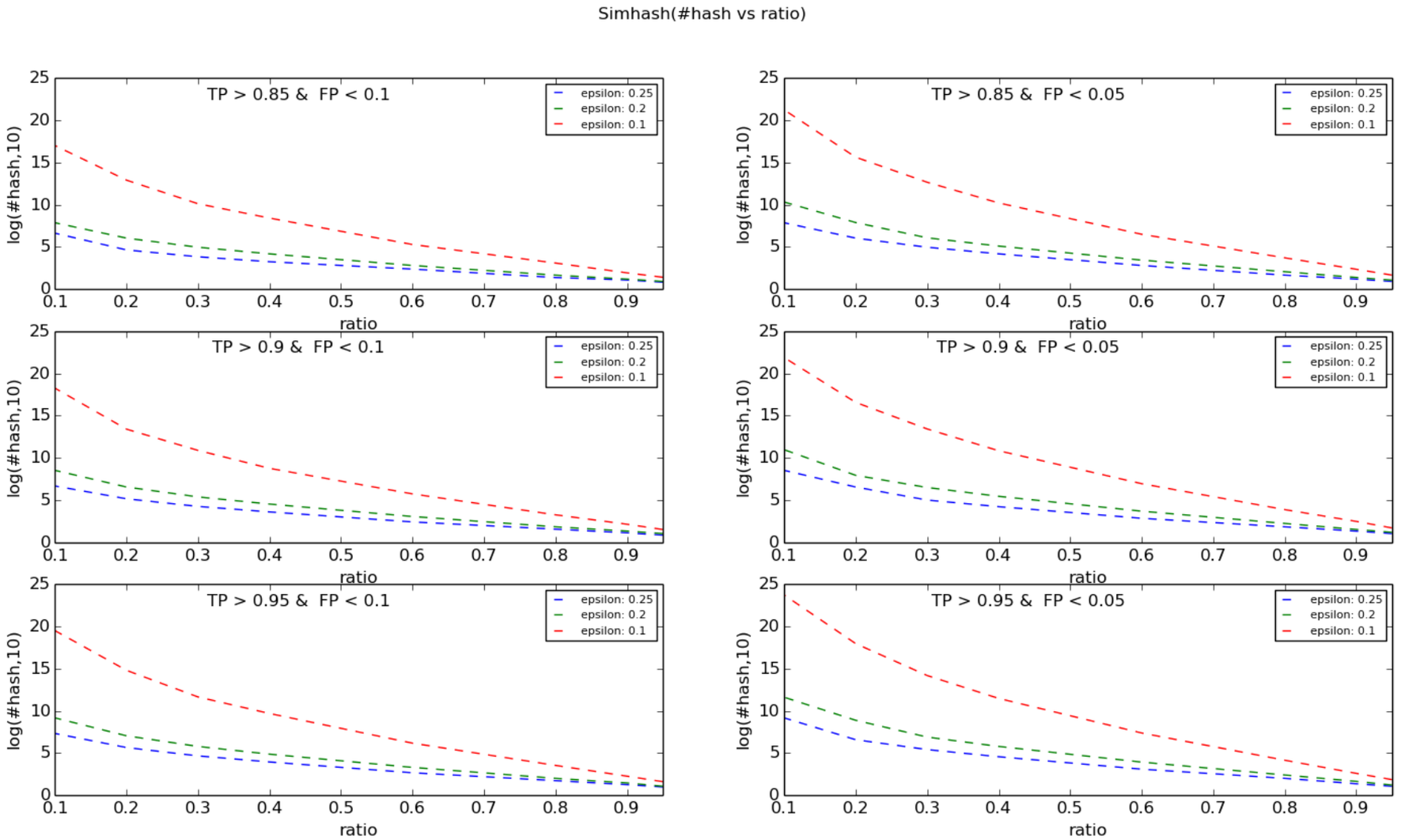

Figure 3: Simhash(no of hash vs ratio(θ/M))

- $tp \approx 1$ and $fp \approx 0$ implies $(K * L)$ increase.
- Minhash need relatively less number of hash values.

LSH Analysis

Dataset Results:

Dataset	#Train	#Query	#Dim	M
EP2006	16088	3309	4,272,227	27,299
MNIST	50000	10000	784	306

Table 3: Datasets

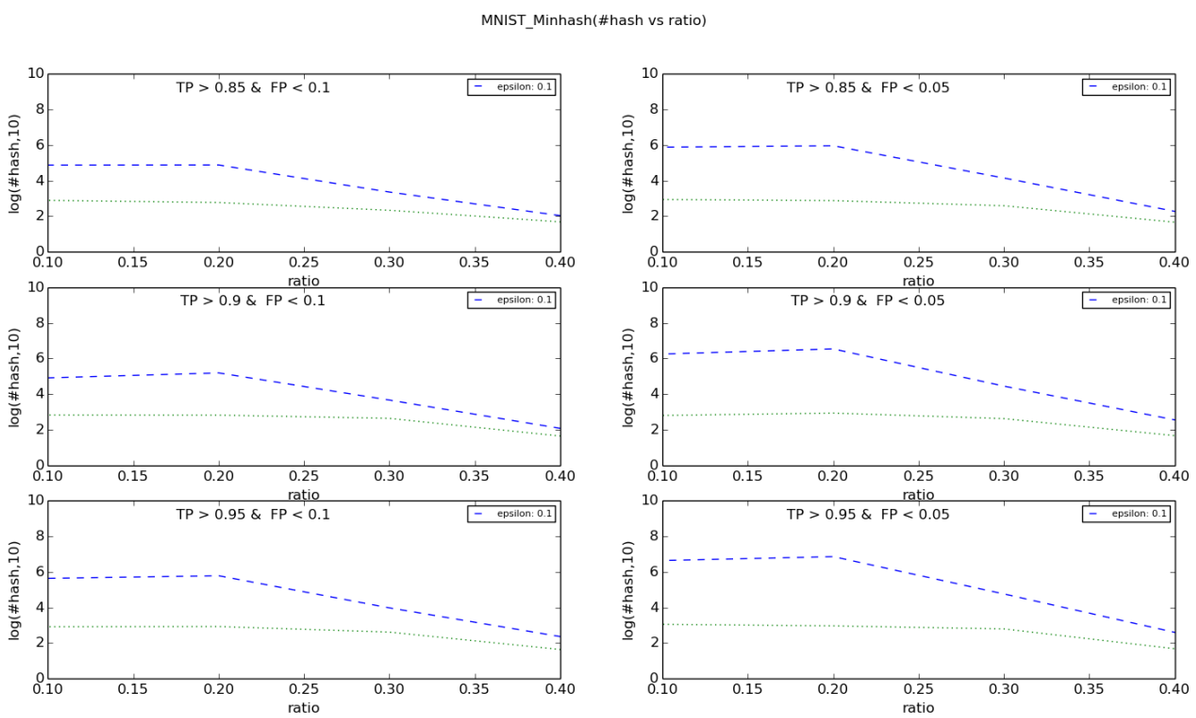

Figure 4: Minhash performance on MNIST Dataset

LSH Analysis

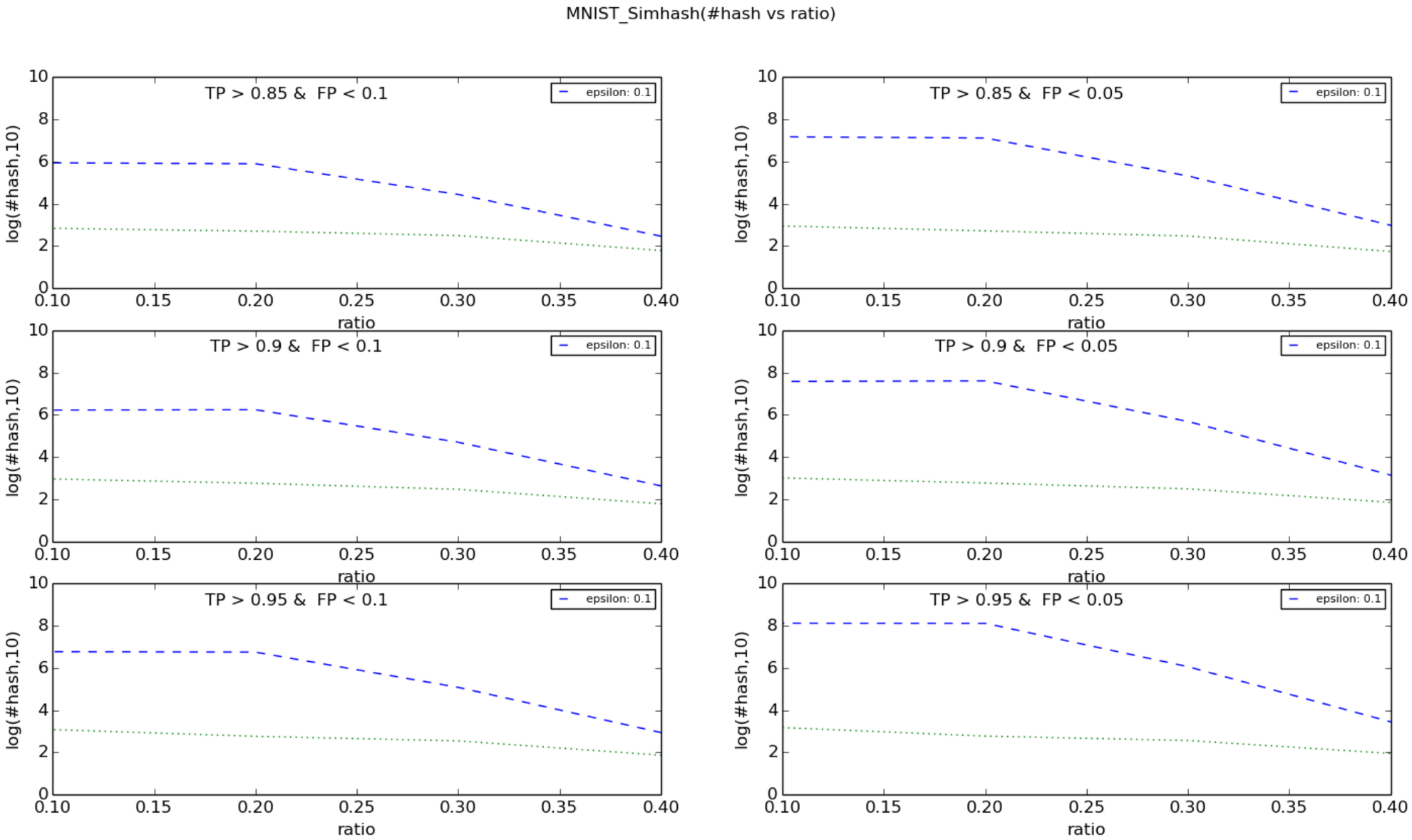

Figure 5: Simhash performance on MNIST Dataset

Two Level Banding

- If $(p_1 - p_2)$ is high then that LSH family takes less number of hash function evaluations.

- Two level banding
  - First level - increase the probability difference
  - Second level - Functions as normal (K,L)-banding scheme

- Any two points x, q are said to be hashed into the same location if the following conditions are satisfied

$$\exists j \text{ s.t. } b_j(x) \equiv b_j(q)\ 1 \leq j \leq L,$$
$$b_j(x) \equiv b_j(q) \text{ iff } \forall i\ O_{ij}(x) \equiv O_{ij}(q)\ 1 \leq i \leq K,$$
$$O_{ij}(x) \equiv O_{ij}(q) \text{ iff } \exists k \text{ s.t } h_{ijk}(x) = h_{ijk}(q)\ 1 \leq k \leq K_1$$

Two level banding analysis

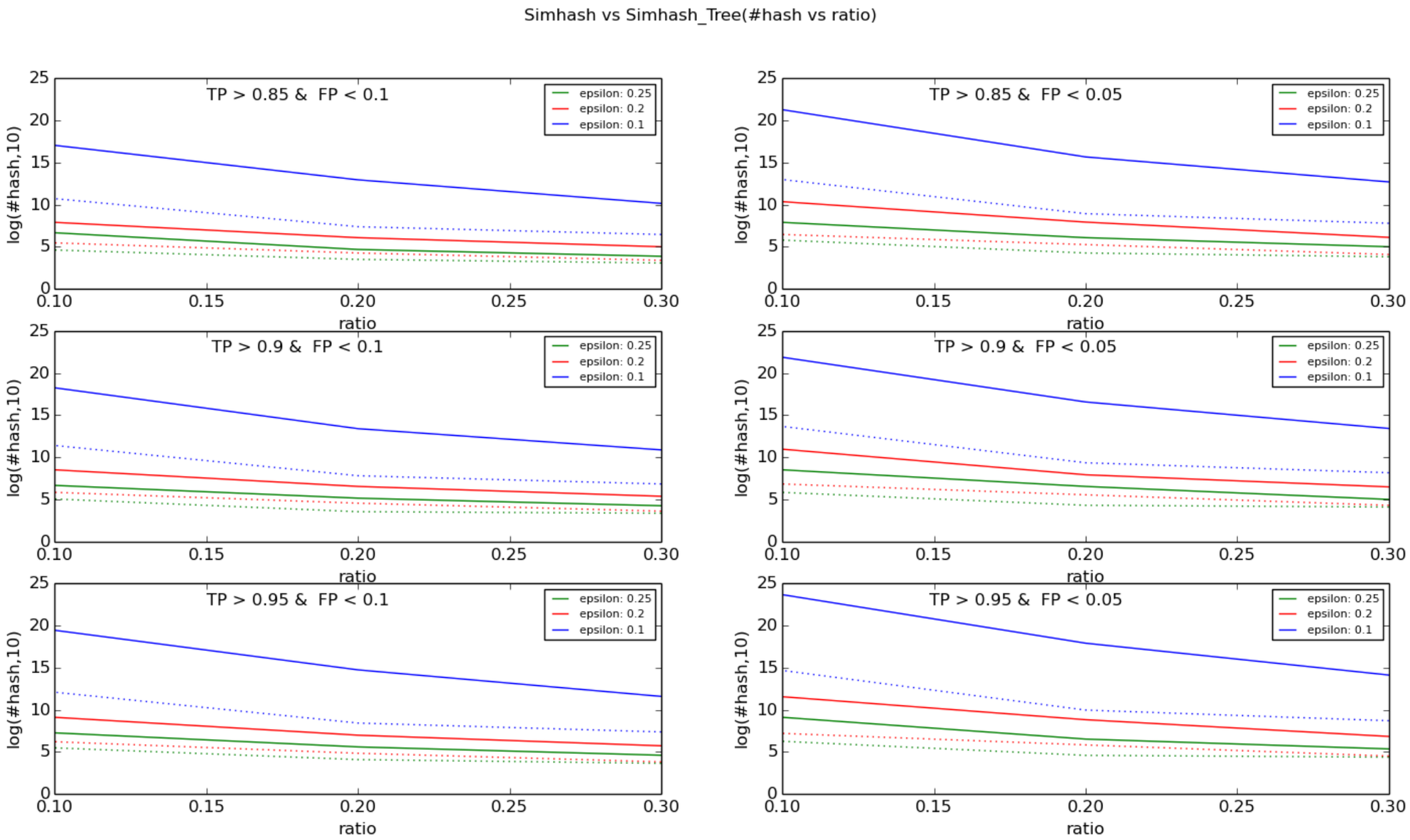

Figure 6: Simhash : Single vs Two Level Banding

Two level banding analysis

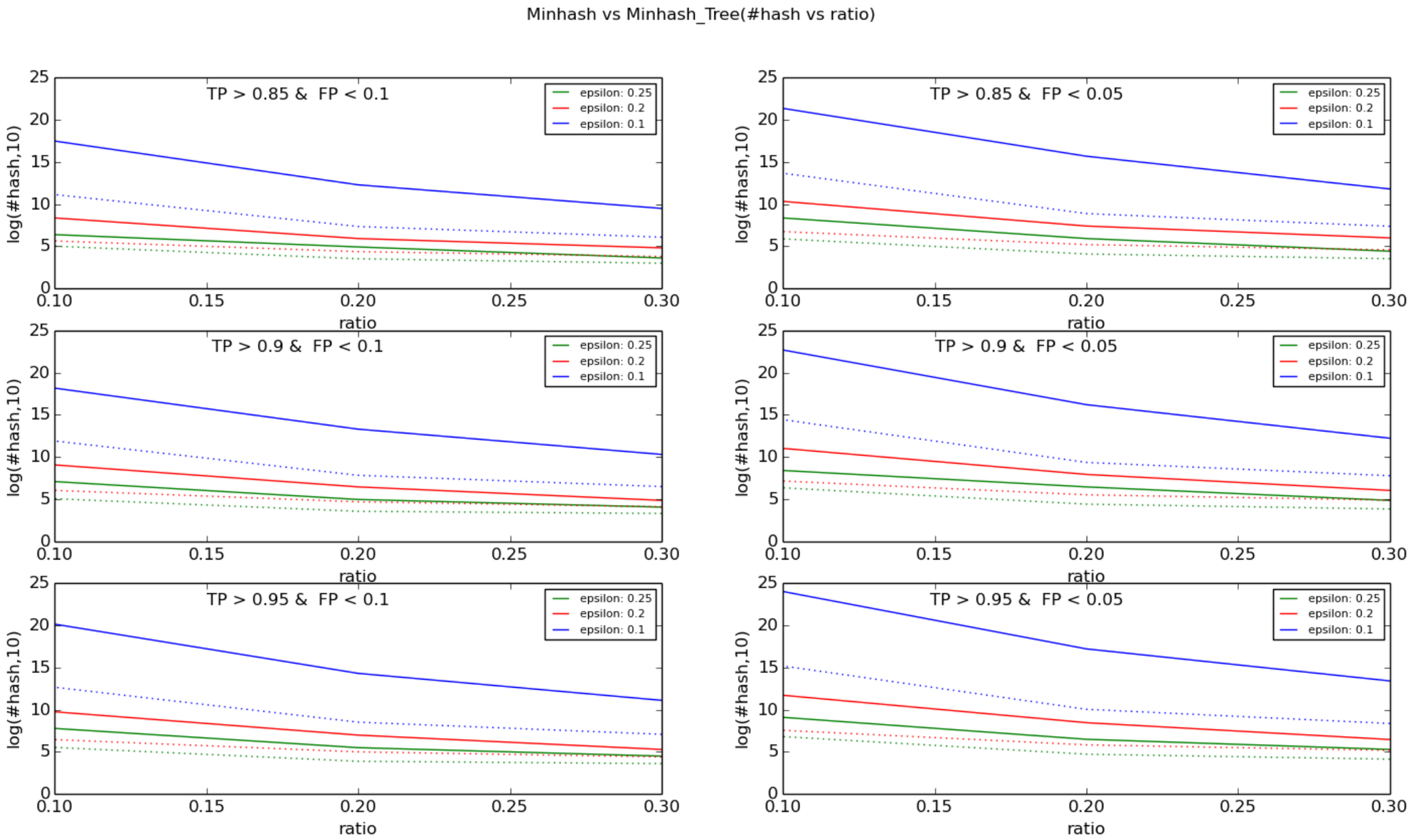

Figure 7: Minhash : Single vs Two Level Banding

Two level banding analysis

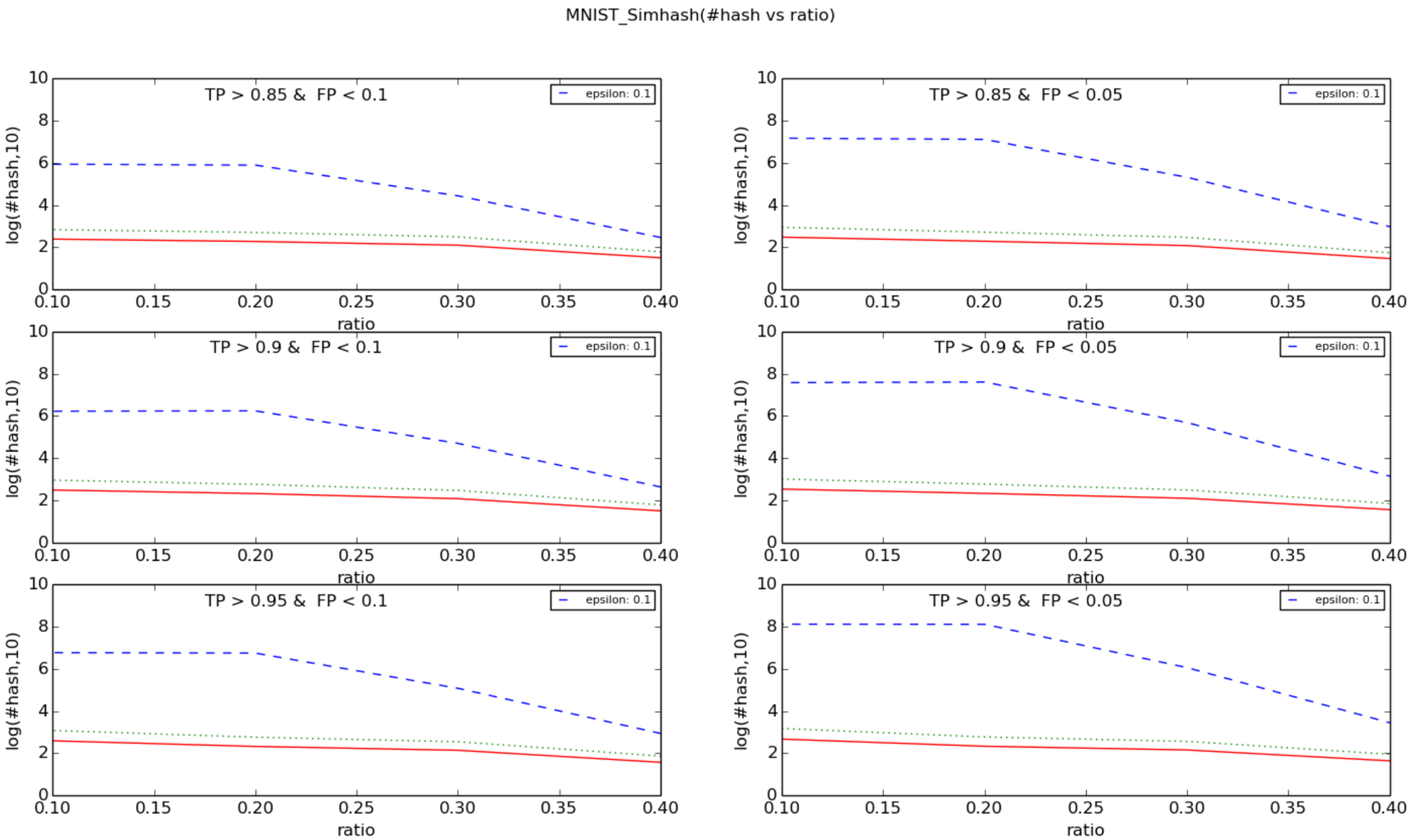

Figure 8: Simhash : Single vs Multi Level Banding(MNIST)

Two level banding analysis

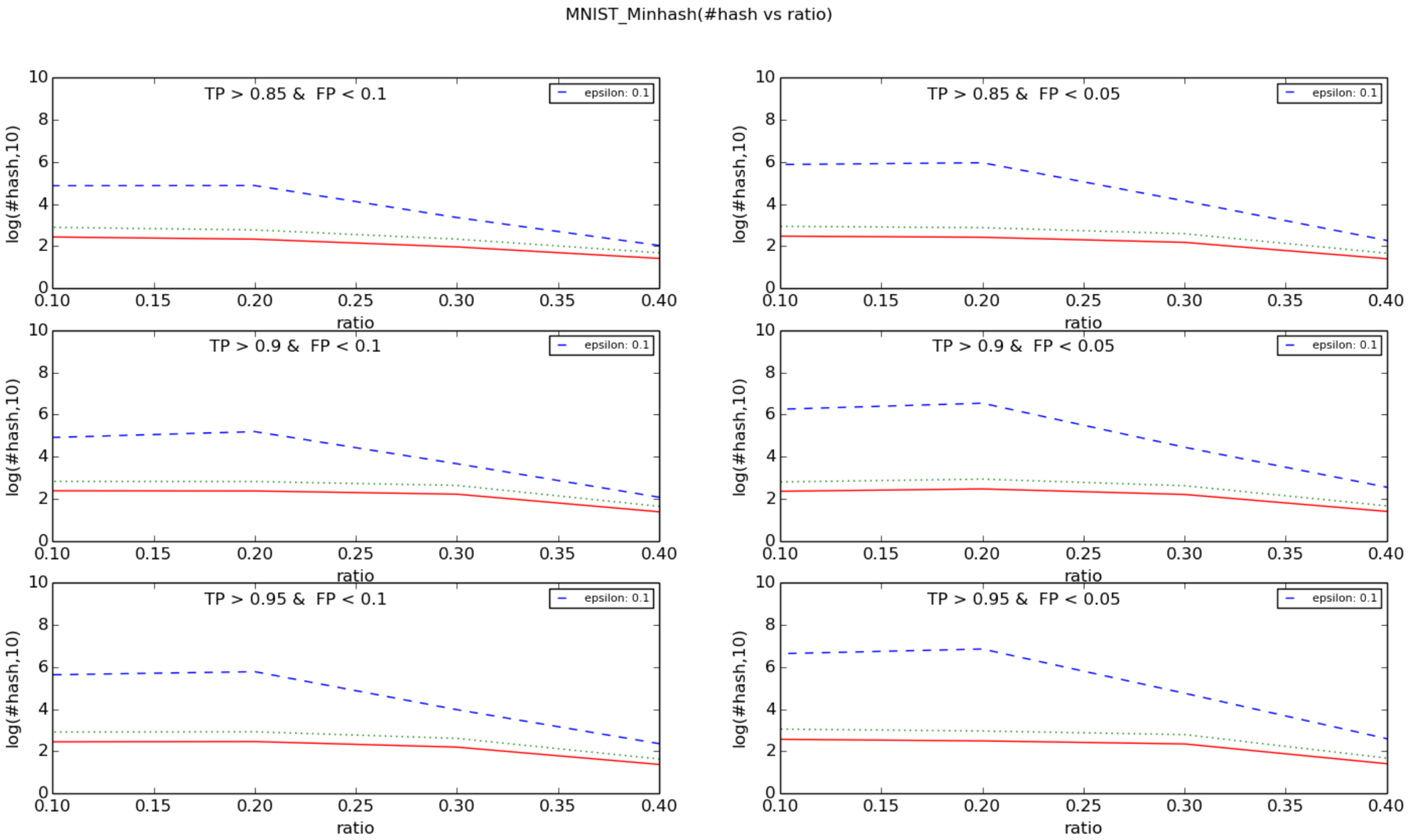

Figure 9: Minhash : Single vs Multi Level Banding(MNIST)

Frequent Pattern Mining

- Extracting all itemsets that occur with required frequency in a transactional database D.

- Frequency is called support

$$\text{Support(s)} = \frac{|X \cap Y|}{|D|}$$

- The Apriori (and its variants) is a famous algorithm
  - works from bottom to top
  - based on anti-monotone property of support
  - At each level k it joins items of previous level k-1.

$$C_k = L_{k-1} * L_{k-1}$$

- The overhead in the Apriori algorithm is the number of candidates it generates at every step and the number of I/O operations it requires to filter these candidates.

LSH functions for support

1. Minhash :

$$\left(\frac{s*n}{2M-s*n}, \frac{(1-\epsilon)s*n}{2M-(1-\epsilon)s*n}, \frac{s*n}{2M-s*n}, \frac{(1-\epsilon)s*n}{2M-(1-\epsilon)s*n}\right)$$

2. Simhash :

$$\left(1-\frac{\arccos\left(\frac{s*n}{M}\right)}{\pi}, 1-\frac{\arccos\left(\frac{(1-\epsilon)s*n}{M}\right)}{\pi}, 1-\frac{\arccos\left(\frac{s*n}{M}\right)}{\pi}, 1-\frac{\arccos\left(\frac{(1-\epsilon)s*n}{M}\right)}{\pi}\right)$$

Apriori Algorithm

Algorithm 1 Aprioiri Algorithm(D,s)

1: l = 1
2: $F = x|x\ is\ s - frequent\ in\ D$
3: $Output\ F$;
4: **while** $F \neq \phi$ **do**
5: $l = l + 1$;
6: **for** $I_a \in F$ **do**
7: **for** $I_b \in F$ **do**
8: add $I_a \cup I_b$ to C if $|I_a \cup I_b| =$ l
9: **end for**
10: **end for**
11: $F = \phi$
12: **for** $I \in C$ **do**
13: $Add\ I\ to\ F\ if\ support\ of\ I \geq s$
14: **end for**
15: $Output\ F$;

LSH-Apriori

Algorithm 2 LSH-Apriori(D,s,ϵ,tp,fp)

1: **for** $x \in F_p$ **do**
2: **for** $i = 1$ to L **do**
3: store x at $H_i(x)$
4: **end for**
5: **end for**
6: **for** $q \in F_q$ **do**
7: cand $\leftarrow \phi$
8: **for** $i = 1$ to L **do**
9: cand $\leftarrow$ cand $\cup$ items at $H_i(q)$
10: **end for**
11: **for** $x \in cand$ **do**
12: I $\leftarrow x \cup q$
13: *Add I to F if support of* $I \geq s$
14: **end for**
15: **end for**

Dataset & Evaluation Metrics

Dataset	# Items	# Transactions	M	avg_nonzeros
BMS1	497	59603	3658	301
BMS2	3340	77513	3766	107

Table 4: Datasets for Frequent Itemset Mining

- Candidate Ratio = $\frac{candidates\_of\_LSH-Apriori}{candidates\_of\_Apriori}$
- Accuracy = $\frac{Items\_of\_LSH-Apriori \cap Items\_of\_Apriori}{Items\_of\_Apriori} * 100$

LSH-Apriori Performance

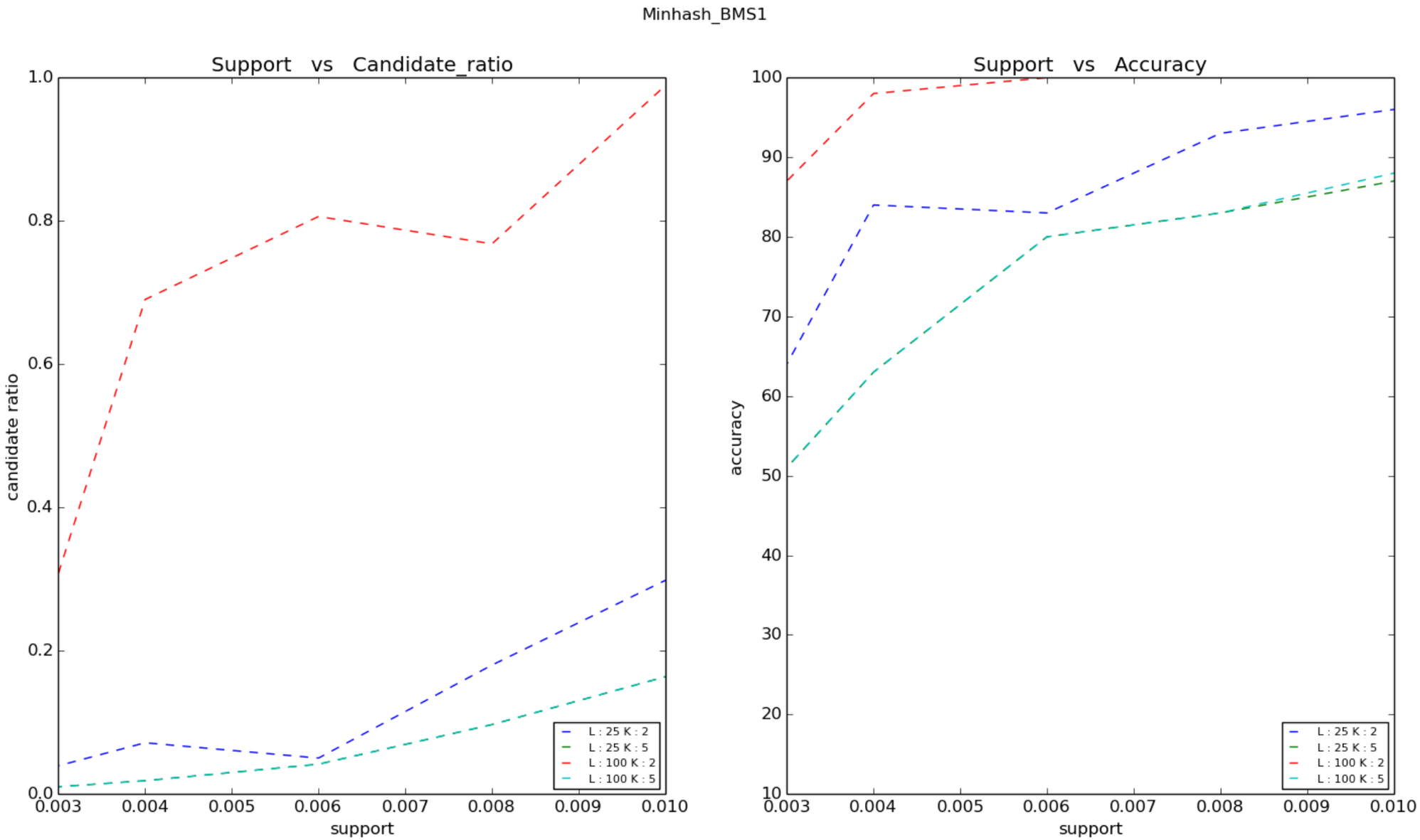

Figure 10: Minhash-Apriori performance on BMS1 dataset

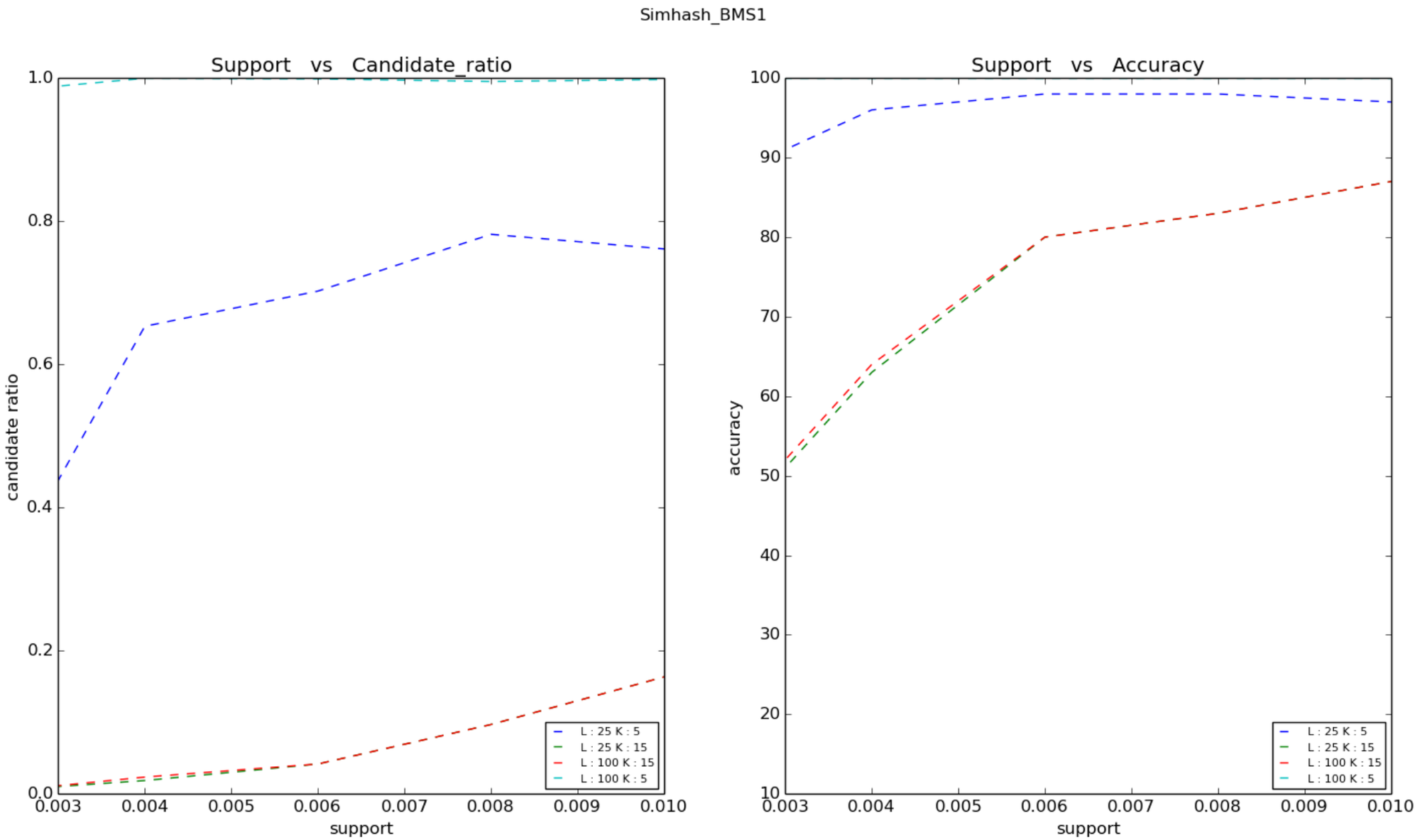

Figure 11: Simhash-Apriori performance on BMS1 dataset

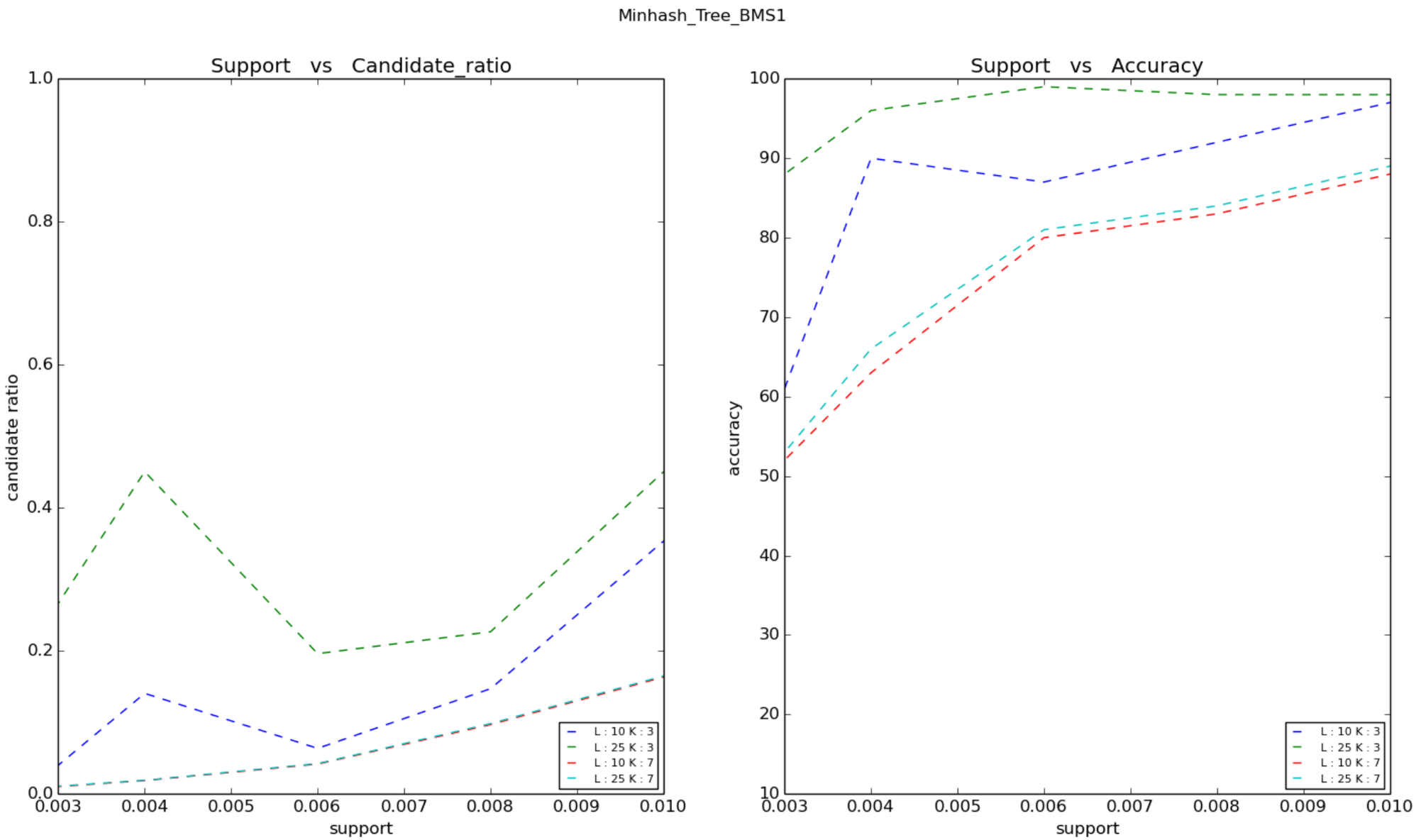

Figure 12: Minhash_Tree performance on BMS1 dataset

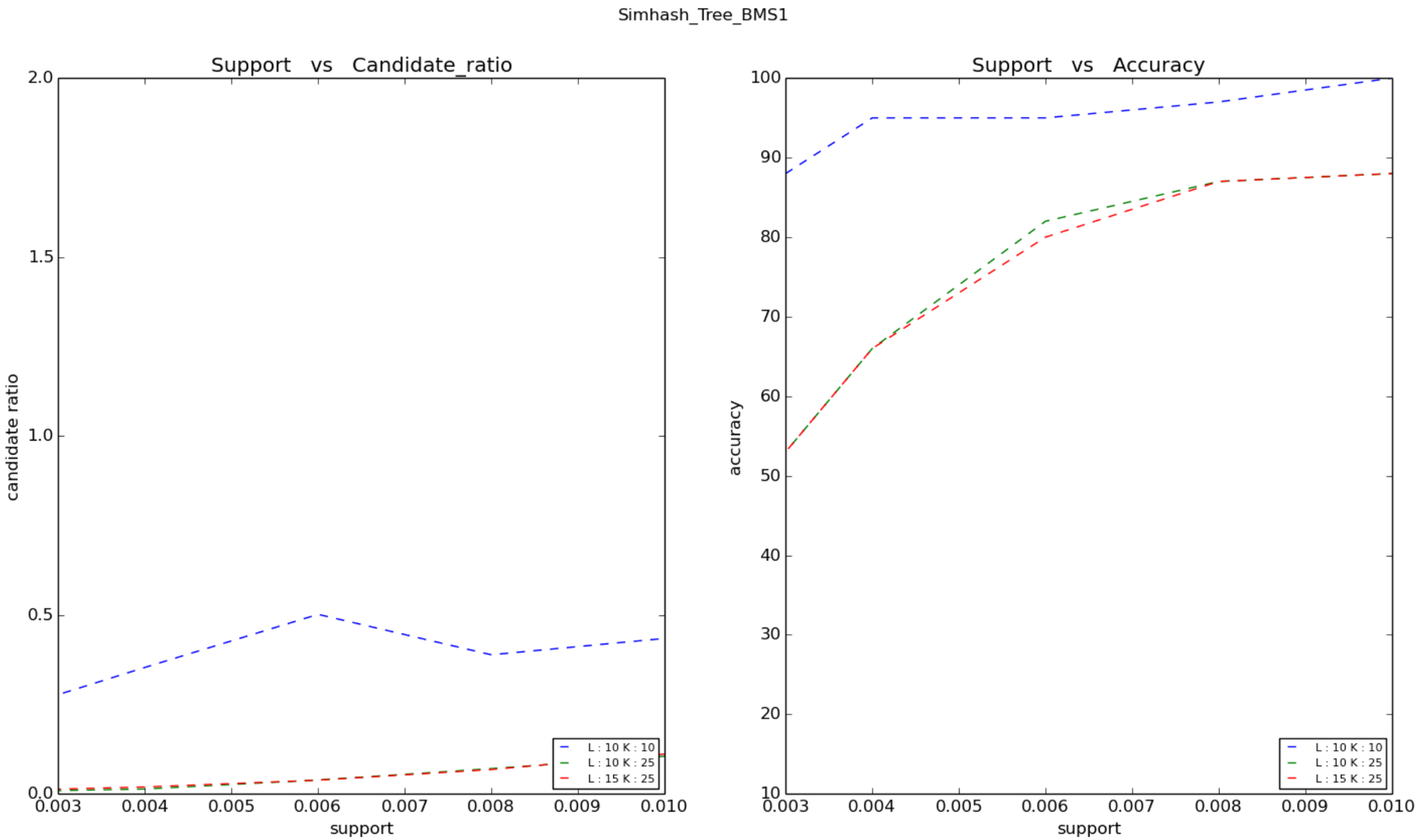

Figure 13: Simhash_Tree performance on BMS1 dataset

Conclusion

- The hash family with better probability difference($p_1 - p_2$) need less number of hash function evaluations to achieve the required tp and fp rate.
- The number of hash function evaluations required to achieve given tp and fp rate can be reduced using two level banding.

Thank You!!