CHAPTER 4

Experimental Results

4.1 Introduction

We evaluate the time efficiency of the proposed algorithms. Four parameters may affect the performance of the algorithms for discovering frequent superset [2]. The first is the effect of minimum support. The second is the number of transactions. The third is the number of items, and the fourth is the average length of transactions. For each parameter, we both test the larger and smaller dataset to show the abilities of these six algorithms, Baseline, Apriori-C, Eclat-C, DCT-Apriori, DCT-Eclat, and DCT-FPGrowth, under the dataset with different size. In addition, the number of candidates for each pass dominates the performance of Apriori-based methods, so we performed an experiment to evaluate the number of candidates in each pass for the Apriori-C and Baseline method.

We choose Apriori, Eclat and FP-Growth as the black box for DCT algorithms to compare with Apriori-C and Eclat-C. To inspect the performance of these six algorithms, Baseline, Apriori-C, Eclat-C, DCT-Apriori, DCT-Eclat, and DCT-FPGrowth under the four parameters, we performed several experiments to measure them on an Intel Xeon 2.4GHz computer with 2GB main memory, and running FreeBSD 5.2-CURRENT. We first describe

the generation of the synthetic datasets used in the assessment. Then we show the performance results of the six frequent superset discovery methods.

4.2 Synthetic Data Generation

The synthetic data was generated using the dataset generator from IBM Almaden Quest research group [16]. To create a dataset, we take the parameters for generation program shown in Table 4.1. The number of transactions is denoted by |D|, the average size of the transactions is denoted by |T|, and the number of entire different items is denoted by N. For example, a dataset D100kT60N100 means that it contains a hundred thousand transactions, a hundred different items, and the average size of transactions are sixty.

Table 4.1: Parameters for dataset generator.

Parameters	Description
|D|	The number of transactions
|T|	Average size of the transactions
N	Number of items

Table 4.2: Descriptions for the abbreviations of values of parameters.

Name	|D|	|T|	N	Size in Megabytes	Size of complement
D100kT60N100	100k	60	100	16	12
D100kT50N100	100k	50	100	13	15
D10kT60N200	10k	60	200	1.9	4.6
D100kT25N50	100k	25	50	6.1	7.3
D10kT25N50	10k	25	50	0.63	0.75
D10kT5N40	10k	5	40	0.12	0.79
D10kT5N10	10k	5	10	0.07	0.09
D10kT25N40	10k	25	40	0.6	0.5

Table 4.2 gives the descriptions for the abbreviations of values of parameters, and the summary of larger and smaller datasets we used in following experiments is shown in Table 4.3.

Table 4.3: The larger and smaller datasets used in our experiment studies.

Parameter	Larger Dataset	Small Dataset
Minimum support	D100kT60N100	D10kT10N20
|D|	D10kT50N10~D100kT50N100	D10kT5N15~D100kT5N15
N	D10kT60N100~D10kT60N200	D10kT5N10~D10kT5N40
|T|	D10kT30N100~D10kT70N100	D10kT10N50~D10kT30N50

4.3 Performance Analysis

4.3.1 Minimum Support

To evaluate the effect of different minimum support, we test two datasets, D10kT10N20 and D100kT60N100. Table 4.4 lists the execution times for taking the complement of these two datasets.

Table 4.4: Execution time in seconds for taking the complement of two dataset D10kT10N20 and D100kT60N100.

Dataset	CPU Time	I/O Time	Total Time
D10kT10N20	0.00	0.66	0.68
D100kT60N100	0.18	37.12	38.59

Table 4.5 depicts the execution time in seconds for the six algorithms at different support from 10% to 90%, using the D10kT10N20 dataset. Note that, in this table, we do not add the extra complement time for DCT methods. According to Table 4.5, although

DCT-FPGrowth is much faster than others, the complement time is even more then the mining time. It increases much execution time of DCT-FPGrowth. Besides, Eclat-based methods are also faster than all Apriori-based algorithms when the minimum support is larger than 80. Because when the minimum support is large, the length of maximum complement-frequent superset becomes short, and the depth-first search strategy has more overhead instead.

Table 4.5: Execution time in seconds for different minimum support (D10kT10N20).

Minimum Suppport (%)	10	20	30	40	50	60	70	80	90
Baseline	150.11	163.37	168.49	171.11	172.32	173.75	173.93	174.08	174.05
Apriori-C	1.66	0.4	0.21	0.07	0.06	0.05	0.05	0.05	0.02
DCT-Apriori	1.72	0.46	0.25	0.1	0.08	0.08	0.08	0.07	0.02
Eclat-C	0.26	0.12	0.08	0.06	0.05	0.05	0.04	0.04	0.04
DCT-Eclat	0.27	0.12	0.09	0.07	0.05	0.06	0.05	0.04	0.05
DCT-FPGrowth	0.05	0.04	0.03	0.03	0.03	0.02	0.03	0.02	0.00

Table 4.6: Execution time in seconds for different minimum support (D100kT60N100).

Minimum Suppport (%)	40	45	50	55	60	65	70
Apriori-C	146.67	64.06	26.27	6.20	2.03	1.00	0.44
DCT-Apriori	148.16	53.02	18.51	6.36	2.95	1.38	0.54
Eclat-C	2.17	1.09	0.62	0.39	0.29	0.21	0.18
DCT-Eclat	2.21	1.13	0.65	0.42	0.32	0.25	0.21
DCT-FPGrowth	1.44	0.37	0.23	0.17	0.14	0.10	0.10

With the larger dataset, we perform another experiment and Table 4.6 gives the execution time in seconds for the five algorithms except the Baseline in different minimum support. Because the Baseline method falls into memory exhausting, we can not show the

results of it. Comparing with the results of smaller dataset, the Apriori-based methods perform much worse, while the Eclat-based methods still have better performance. All of the algorithms have less execution time in lower minimum support. It is because that the lower the minimum support is, the more the shorter minimum frequent superset is, and in turn, the more the longer complement-frequent superset is.

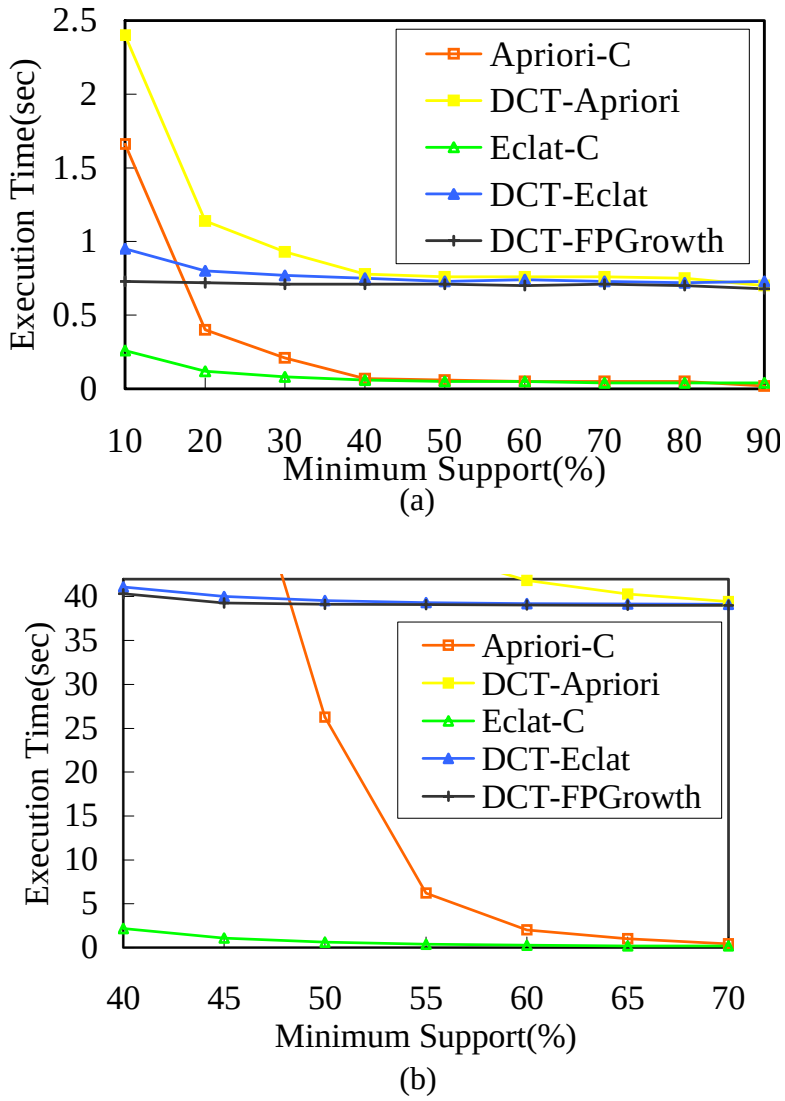

Figure 4.1: The execution time in different minimum support threshold for the database (a) D10kT10N20 and (b) D100kT60N100.

We also show the figures of experimental results in Figure 4.1. Both Eclat-based and

DCT-FPGrowth have good scalability for different minimum support values. In conclusion, Eclat-C is the best method in this test of different minimum support, and DCT-FPGrowth performances better than Apriori-based methods when the minimum support is small.

4.3.2 Number of Transactions

The second experiment is the evaluation of effect of the number of transactions. First, we list the execution time for generating the complement dataset in Table 4.7.

Table 4.7: The execution time for generating the complement dataset.

Dataset	CPU Time	I/O Time	Total Time
D10kT5N15	0.01	0.39	0.41
D20kT5N15	0.00	0.80	0.85
D30kT5N15	0.02	1.19	1.24
D40kT5N15	0.02	1.59	1.66
D50kT5N15	0.00	1.99	2.41
D60kT5N15	0.02	2.37	2.41
D70kT5N15	0.00	2.78	2.80
D80kT5N15	0.03	3.18	3.32
D90kT5N15	0.04	3.57	3.85
D100kT5N15	0.04	3.97	4.18
D10kT50N100	0.00	3.44	3.45
D20kT50N100	0.03	6.87	6.90
D30kT50N100	0.03	10.36	10.48
D40kT50N100	0.04	13.74	13.79
D50kT50N100	0.05	17.28	17.36
D60kT50N100	0.08	20.63	20.81
D70kT50N100	0.04	24.15	24.31
D80kT50N100	0.02	27.60	27.69
D90kT50N100	0.14	31.27	31.63
D100kT50N100	0.16	35.42	35.90

Table 4.8 shows the execution time except the complement time for the six algorithms in different number of transactions from 10k to 100k, the test dataset is T5N15 and the minimum support is set to 60%. The experiment of larger dataset is given in Table 4.9, the dataset is T50N100 and the minimum support is also set to 60%. The execution time increases when the number of transactions increases.

Table 4.8: Execution time in seconds for different number of transactions from 10k to 100k (dataset=T5N15, minsup=60%).

Number of Transactions (k)	10	20	30	40	50	60	70	80	90	100
Baseline	2.8	3.04	3.27	3.32	3.42	3.55	3.65	3.68	3.73	3.8
Apriori-C	0.02	0.05	0.08	0.11	0.13	0.16	0.19	0.22	0.25	0.28
DCT-Apriori	0.06	0.11	0.18	0.24	0.29	0.35	0.42	0.48	0.54	0.6
Eclat-C	0.02	0.04	0.07	0.09	0.13	0.14	0.16	0.2	0.23	0.26
DCT-Eclat	0.03	0.08	0.12	0.17	0.21	0.26	0.31	0.36	0.4	0.46
DCT-FPGrowth	0.02	0.03	0.04	0.06	0.08	0.10	0.11	0.12	0.13	0.15

Table 4.9: Execution time in seconds for five algorithms in different number of transactions from 10k to 100k and the datasets are T50N100.

Number of Transactions (k)	10	20	30	40	50	60	70	80	90	100
Apriori-C	47.15	95.25	139.48	180.47	222.70	265.28	305.70	348.16	387.80	429.59
DCT-Apriori	49.57	101.96	140.90	192.02	229.54	267.16	317.80	362.30	400.94	457.75
Eclat-C	0.75	1.38	2.05	2.59	3.23	3.85	4.38	4.97	5.57	6.16
DCT-Eclat	0.75	1.38	2.04	2.58	3.24	3.82	4.38	4.95	5.54	6.37
DCT-FPGrowth	0.28	0.52	0.73	0.98	1.21	1.41	1.64	2.13	2.08	2.29

Figure 4.2 depicts the scalability of the six algorithms in small dataset. We can see that the execution times of Apriori-C and Eclat-C algorithms are increasing linearly with the number of transactions. The slope of Apriori-C is a little larger than Eclat-C. When the

number of transactions is more than 80, the DCT methods become even worse than Baseline method. This is because they have to take the complement of dataset before mining, and when |D|>80, they waste too much time on taking the complement of dataset. However, for the DCT methods, DCT-FPGrowth is better than others.

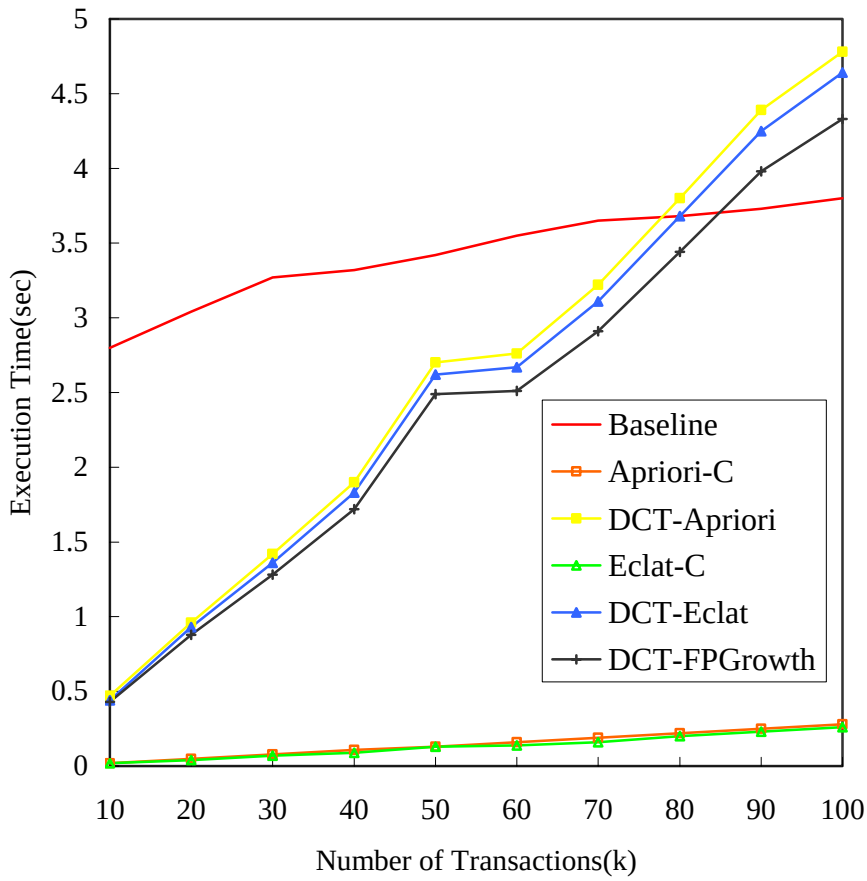

Figure 4.2: The scalability for six algorithms, Baseline, Apriori-C, DCT-Apriori, Eclat-C, DCT-Eclat, and DCT-FPGrowth in different number of transactions.

Figure 4.3 shows the scalability of Eclat-C and DCT-FPGrowth when the dataset is large. The execution time of them is growing linearly with the number of transactions. However, the slop of Eclat-C is smaller than DCT-FPGrowth, and we recognize that Eclat-C

is the better method while the dataset is huge.

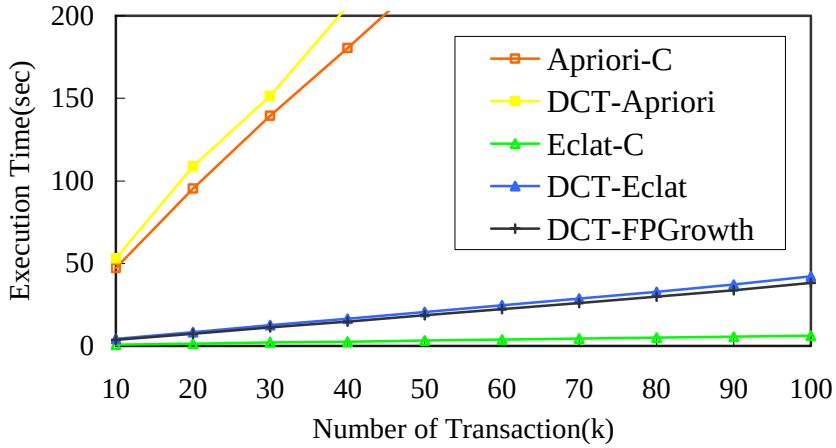

Figure 4.3: The scalability in different number of transaction when the dataset is large.

4.3.3 Number of Items

The third parameter is the total number of items in the transaction database. Basically, when the number of items increases, the size of the complement of an itemset increases, too. Also, we list the execution time for transforming dataset into complement in Table 4.10. In this situation, the variation of execution time for transforming dataset into complement is smoother than the previous experiment of different number of transactions while the parameters grow up. This is an advantageous point of DCT method.

The experiment results are shown in Table 4.11 for the execution times of different number of items from 10 to 40, and use small dataset D10kT5 with the minimum support 60%. When the number of items is more than 25, Baseline falls into memory exhausting, and we fill in the field with NA.

Table 4.10: The execution time for transforming dataset into complement.

Dataset	CPU Time	I/O Time	Total Time
D10kT5N10	0.00	0.30	0.31
D10kT5N15	0.00	0.40	0.40
D10kT5N20	0.00	0.48	0.51
D10kT5N25	0.00	0.57	0.60
D10kT5N30	0.00	0.65	0.69
D10kT5N35	0.00	0.73	0.77
D10kT5N40	0.01	0.80	0.84
D10kT60N100	0.03	3.67	3.70
D10kT60N120	0.00	4.59	4.59
D10kT60N140	0.02	4.57	4.59
D10kT60N160	0.00	5.10	5.19
D10kT60N180	0.02	5.43	5.51
D10kT60N200	0.01	5.84	5.91

Table 4.11: Execution time in second for different number of itmes from 10 to 40 (dataset=D10kT5, minsup=60%).

Number of Items	10	15	20	25	30	35	40
Baseline	0.08	2.78	226.2	NA	NA	NA	NA
Apriori-C	0.02	0.02	0.03	0.06	0.44	7.61	55.53
DCT-Apriori	0.02	0.05	0.09	0.22	0.99	6.88	62.75
Eclat-C	0.02	0.01	0.03	0.06	0.1	0.28	0.94
DCT-Eclat	0.01	0.04	0.54	0.08	0.14	0.33	0.99
DCT-FPGrowth	0.01	0.02	0.03	0.04	0.06	0.09	0.19

The result of other experiment with larger dataset is given in Table 4.12, the dataset is D10kT60, and the minimum support is set to 60%. For large dataset, DCT-FPGrowth is the best algorithm, and even adding the complement time, it performs better than Eclat-C while the number of items larger than 140. No matter what the dataset is, both of the two experimental results of small and large datasets show that when the number of items increases,

the execution time increases acutely. Figure 4.5 and 4.6 also show the graphs of the results.

Table 4.12: Execution times in second for different number of items from 100 to 200 (dataset=D10kT60, minsup=60%).

Number of Items	100	120	140	160	180	200
Apriori-C	2.04	219.34	4470.14	NA	NA	NA
DCT-Apriori	3.35	197.10	NA	NA	NA	NA
Eclat-C	0.31	3.15	38.79	737.16	NA	NA
DCT-Eclat	0.29	3.17	38.56	731.66	NA	NA
DCT-FPGrowth	0.13	0.65	5.34	106.56	3593.15	NA

Although we just change the number of items, the average size of transactions of their complement dataset also been changed, too.

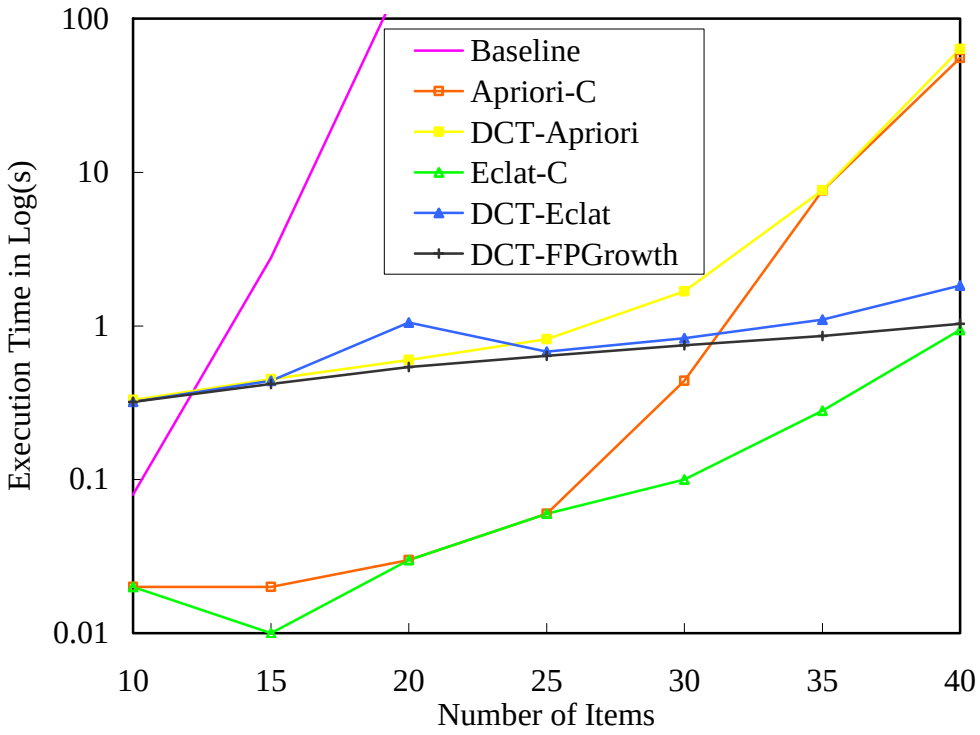

Figure 4.4 Results of the different number of itmes (dataset=D10kT5, minsup=60%).

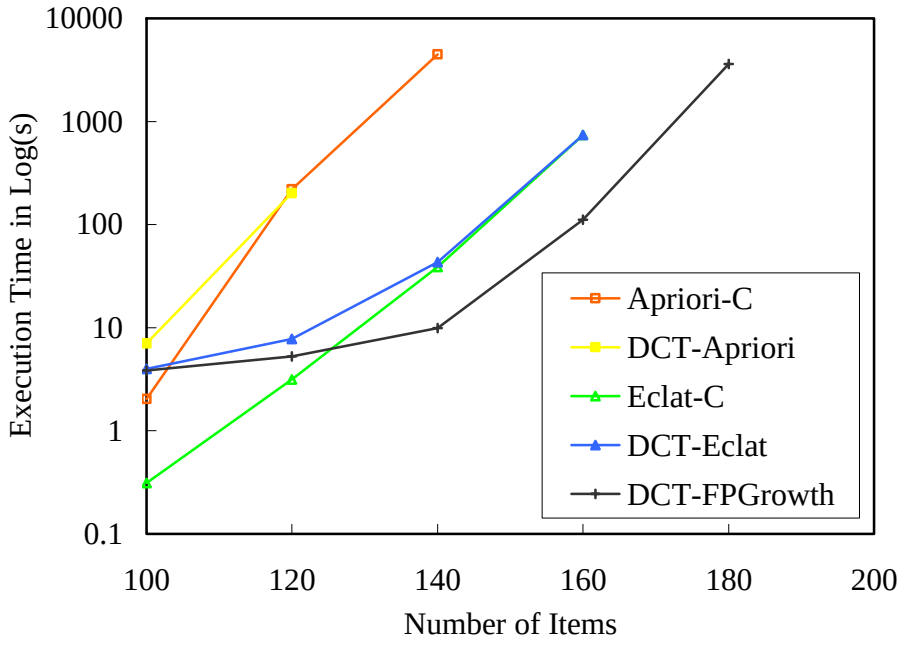

Figure 4.5 Results of the different number of itmes (dataset=D10kT60, minsup=60%).

Example 4.1 Consider a dataset *D* with *d* transactions, and there are *n* different items. If the average size of transaction is *t*, that is $\frac{l_1 + l_2 + ... + l_d}{d} = t$, where l_i is the length of the *i*-th transaction, the average size of transaction of complement dataset *D'* is

$$\frac{(n-l_1)+(n-l_2)+...+(n-l_d)}{d} = \frac{dn-(l_1+l_2+...+l_d)}{d} = n - \frac{l_1+l_2+...+l_d}{d} = n-t\,.$$

Owing to that the proposed algorithm are related to the complement view of point, we test the datasets, D10kT40N100, D10kT60N120,…, and D10kT140N200. They all have the fixed (N-|T|)=60, the average size of transactions of complement dataset. Intuitively, the average size of transactions influences the time of one database scan. With the size of

transactions growing up, we have to pay more efforts to scan the database. However, Table 4.13 shows the different results of execution time against our recognition. When the number of items increasing, the performance of these five algorithms become more efficient. The result is opposite to the previous result with fixed |T|.

Considering the occurrence probability of candidate 1-complement-supersets, $\frac{|T|}{N}$. If the probability increases, the candidates become fewer, maximum complement-frequent superset becomes shorter, and the execution time becomes less. Actually, the ratio of |T| to N increases with N.

Proof Given four natural numbers N_1, N_2, T_1, and T_2, where $N_1 < N_2$, and $(N_1\text{-}T_1) = (N_2\text{-}T_2) = k$, $k > 0$.

$$\because N_1 < N_2 \Leftrightarrow \frac{k}{N_1} > \frac{k}{N_2} \Leftrightarrow 1-\frac{k}{N_1} < 1-\frac{k}{N_2} \Leftrightarrow \frac{N_1-k}{N_1} < \frac{N_2-k}{N_2} \Leftrightarrow \frac{T_1}{N_1} < \frac{T_2}{N_2}$$

$\therefore$ The ratio of |T| to N increases with N.

For example, the probability of N=80 and |T|=40 is 0.5, and N=100 and |T|=60 is 0.6. In fact, this is a tradeoff between transaction size and candidates occurrence probability. Table 4.14 gives the result of the other dataset, and only Eclat-C becomes worse with the number of items increasing. This is because the reduction of the number of candidates is less important than the size of transactions. The datasets are D10kT40N80, D10kT60N100, …, and D10kT140N180. The execution time of generating complement datasets is listed in Table 4.15.

Table 4.13: Execution time for different number of items from 100 to 200 (N-|T|=60, |D|=10k, minsup=60%).

Number of Items	100	120	140	160	180	200
Apriori-C	801.34	319.65	135.50	75.74	65.87	2.82
DCT-Apriori	734.26	234.99	115.01	53.76	52.76	3.33
Eclat-C	4.58	3.33	1.56	1.07	1.03	0.65
DCT-Eclat	4.64	3.25	1.54	0.97	0.91	0.44
DCT-FPGrowth	1.19	0.70	0.46	0.40	0.35	0.28

Table 4.14: Execution time for different number of items from 80 to 180 (N-|T|=40, |D|=10k, minsup=60%).

Number of Items	80	100	120	140	160	180
Apriori-C	2.52	2.02	1.53	1.28	1.20	1.29
DCT-Apriori	4.09	2.94	1.83	1.36	1.05	1.07
Eclat-C	0.28	0.30	0.38	0.37	0.39	0.45
DCT-Eclat	0.29	0.28	0.29	0.24	0.22	0.21
DCT-FPGrowth	0.15	0.14	0.13	0.12	0.12	0.12

Table 4.15: Execution time for generating complement dataset.

Dataset	CPU Time	I/O Time	Total Time
D10kT40N100	0.02	3.16	3.19
D10kT60N120	0.03	4.11	4.15
D10kT80N140	0.00	5.10	5.11
D10kT100N160	0.00	6.05	6.11
D10kT120N280	0.01	6.99	7.01
D10kT140N200	0.01	7.93	7.94
D10kT40N80	0.01	2.74	2.77
D10kT60N100	0.02	3.67	3.73
D10kT80N120	0.00	4.65	4.68
D10kT100N140	0.01	5.58	5.62
D10kT120N160	0.00	6.53	6.61
D10kT140N180	0.05	7.42	7.50

While Table 4.13 and Table 4.14 do not consider the complement time of DCT algorithms, Figure 4.7 and 4.8 add the complement time to the execution time. The complement time of DCT-Eclat and DCT-FPGrowth make the performances becoming worse while the average size of transactions increases. However, the influence of occurrence probability on performance is more significant for Apriori-based algorithms than other depth-first algorithms, because only the performances of Apriori-based algorithms have dependence on the number of candidates. In addition, from the two figures, Figure 4.7 and Figure 4.8, the influence of occurrence probability increasing for the former is greater than the later, because the slopes of Apriori-C and DCT-Apriori in Figure 4.7 are larger then the ones in Figure 4.8. Besides, the slope of Eclat-C becomes positive in Figure 4.8.

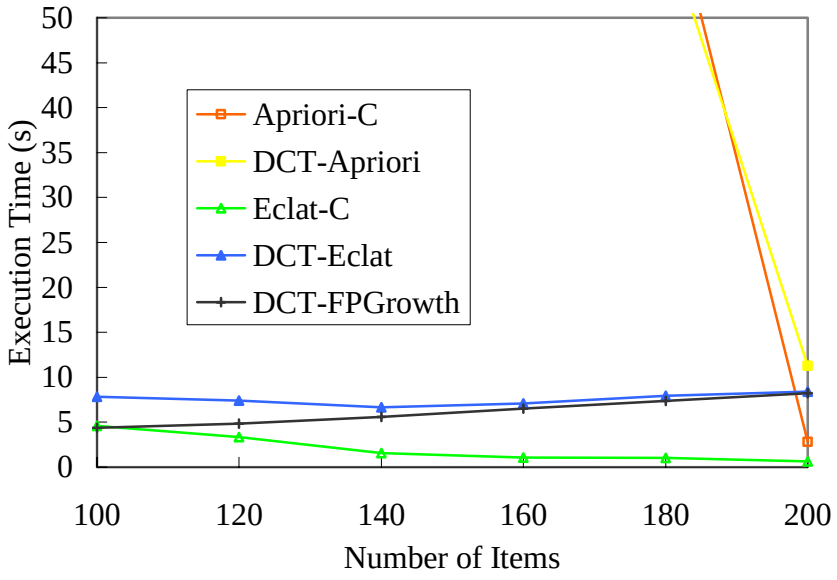

Figure 4.6: Execution time in seconds for different number of items (N-|T|=60, |D|=10k, minsup=60%).

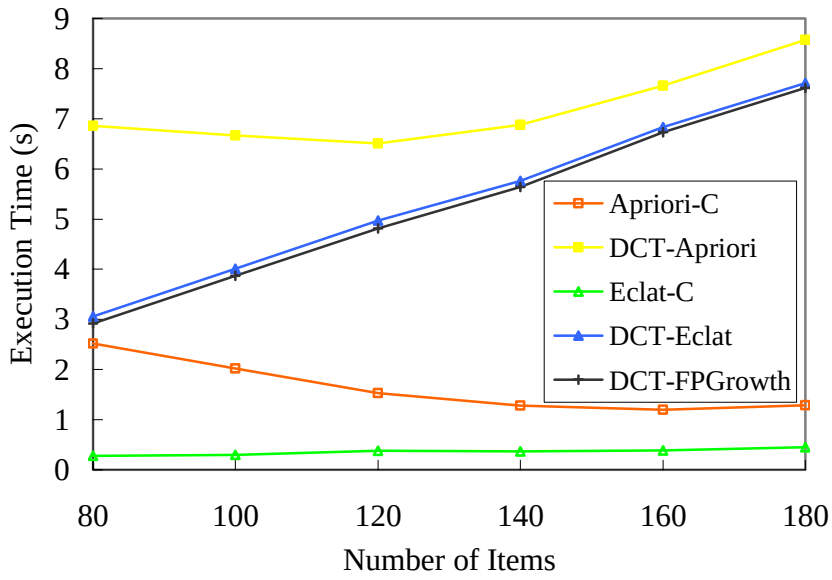

Figure 4.7: Execution time in seconds for the different number of items (N-|T|=40, |D|=10k, minsup=60%).

4.3.4 Average Size of Transactions

The forth is the evaluation of the average size of transactions. When the average size of transactions increases, the average size of transactions of the complement dataset decreases. And for DCT algorithms, they mine frequent subset from a smaller database, the execution becomes faster. For Apriori-C and Eclat-C, when the average size of transactions increases, the appearance probability of supersets increase, and leads to the number of candidate complement-frequent supersets decrease.

Table 4.16 and Table 4.17 show the results of different datasets, and we can see that DCT-Apriori and DCT-Eclat have better performances when the average size of transactions is larger. Note that, we just consider the execution of the black box for DCT algorithms currently. Besides, the algorithms with depth-first search strategy, Eclat-based and

DCT-FPGrowth, are better then Apriori-based methods when the average size of transactions is small. Because when the average size of transactions is small, the complement-frequent superset becomes longer, and the depth-first methods have more superiority.

Table 4.16: Execution time in seconds for different average size of transactions from 10 to 30 (dataset=D10kN50, minsup=60%).

Average Size of Transactions	10	15	20	25	30
Baseline	NA	NA	NA	NA	NA
Apriori-C	473.37	26.42	1.08	0.24	0.17
DCT-Apriori	518.99	19.25	1.71	0.37	0.17
Eclat-C	4.88	0.53	0.22	0.12	0.11
DCT-Eclat	4.92	0.57	0.23	0.13	0.09
DCT-FPGrowth	0.80	0.19	0.11	0.06	0.05

Table 4.17: Execution time in seconds for different average size of transactions from 30 to 70 (dataset=D10kN100, minsup=60%).

Average Size of Transactions	30	35	40	45	50	55	60	65	70
Apriori-C	NA	2361.89	612.26	188.7	57.26	11.13	2.15	0.89	0.70
DCT-Apriori	NA	NA	591.03	168.47	43.72	7.92	2.97	1.06	0.52
Eclat-C	84.08	16.41	4.49	1.58	0.73	0.41	0.31	0.25	0.23
DCT-Eclat	84.51	16.46	4.56	1.63	0.75	0.41	0.28	0.20	0.15
DCT-FPGrowth	13.63	3.15	1.10	0.50	0.29	0.20	0.14	0.10	0.08

The execution time for generating complement dataset is shown in Table 4.18. Figure 4.9 and 4.10 show the scalabilities for different average size of transactions with the execution time of complement for DCT methods in small and larger datasets. According to Figure 4.9 and 4.10, when the average size of transaction is large, DCT-Apriori has better performances than Apriori-C. Furthermore, DCT-FPGrowth becomes the best algorithm when then dataset is large.

Table 4.18: Execution time for generating complement datasets.

Dataset	CPU Time	I/O Time	Total Time
D10kT10N50	0.00	1.26	1.35
D10kT15N50	0.00	1.40	1.43
D10kT20N50	0.00	1.53	2.36
D10kT25N50	0.02	1.64	1.69
D10kT30N50	0.02	1.75	1.83
D10kT30N100	0.01	3.16	3.18
D10kT35N100	0.02	3.44	3.47
D10kT40N100	0.01	3.44	3.46
D10kT45N100	0.00	3.81	3.81
D10kT50N100	0.00	3.78	3.79
D10kT55N100	0.01	3,83	3.90
D10kT60N100	0.01	3.74	3.76
D10kT65N100	0.01	3,96	3.98
D10kT70N100	0.00	4.41	4.42

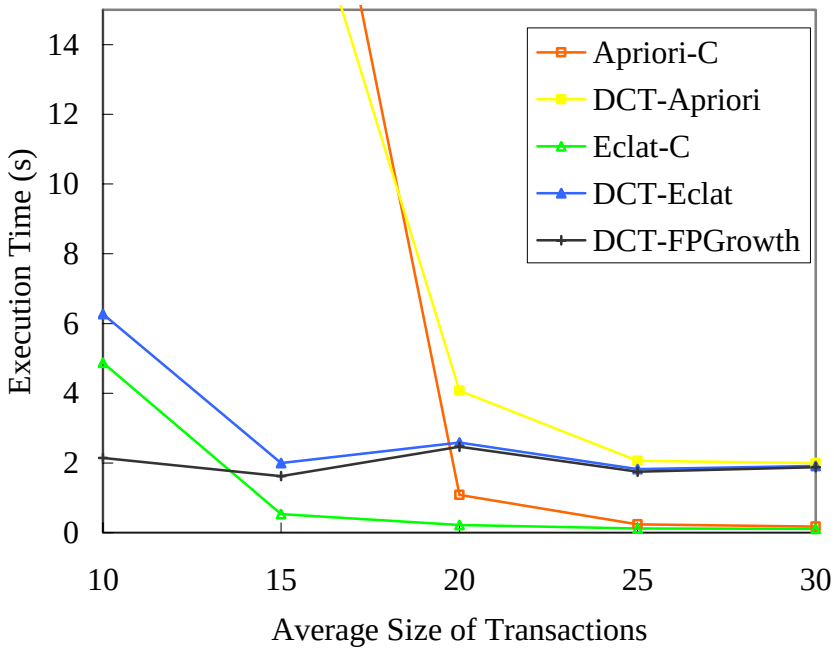

Figure 4.8: Execution time for different average size of transactions (smaller dataset).

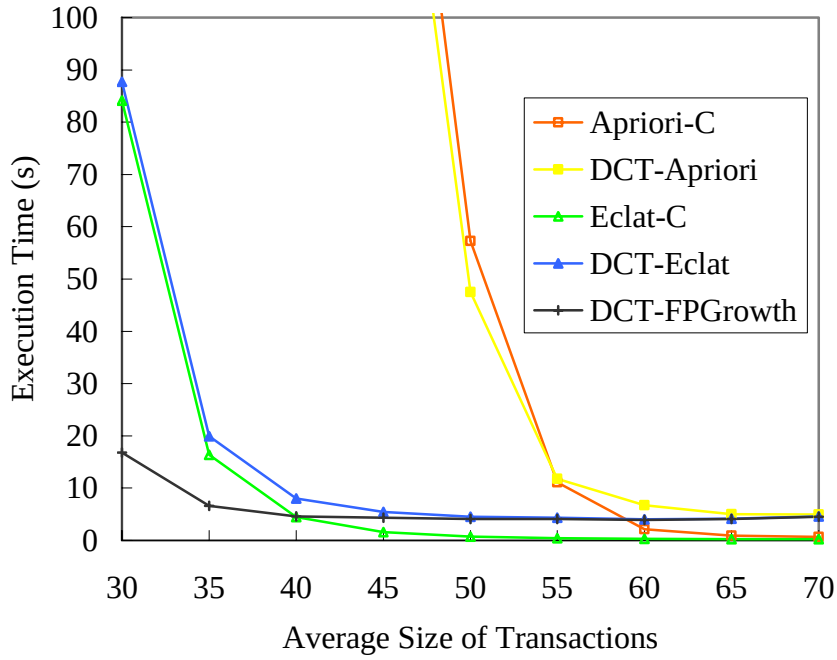

Figure 4.9: Execution time for different average size of transactions (larger dataset).

4.3.5 Number of Candidates

Finally, we evaluate the performance of Baseline and Apriori-C by comparing the number of candidates generated in each passes shown in Figure 4.8 and 4.9. The test dataset is D10kT5N20 with minimum support 50%. Obviously, the Baseline method prunes much fewer candidates than Apriori-C, and needs more passes to finish mining. We think that number of candidates and number of passes are main reasons for Apriori-C to have much better performance. Note that the definition of candidates in the Baseline and Apriori-C are different. In the Baseline algorithm, it means candidate superset, but in Apriori-C, it means candidate complement-superset.

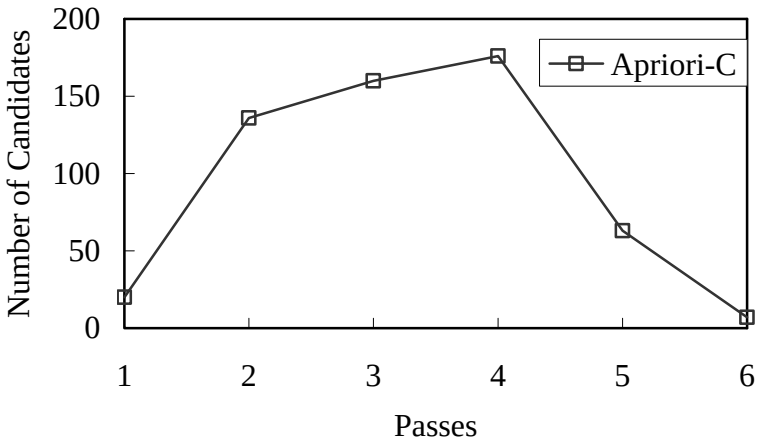

Figure 4.10: Number of Candidates in each pass (Apriori-C).

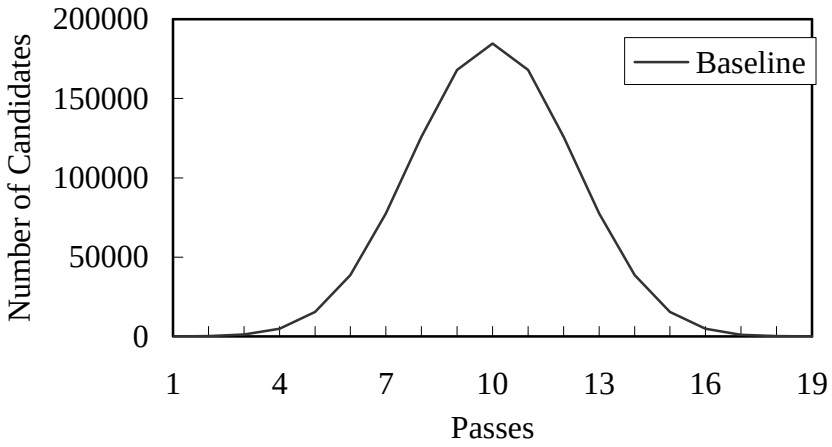

Figure 4.11: Number of Candidates in each pass (Baseline).