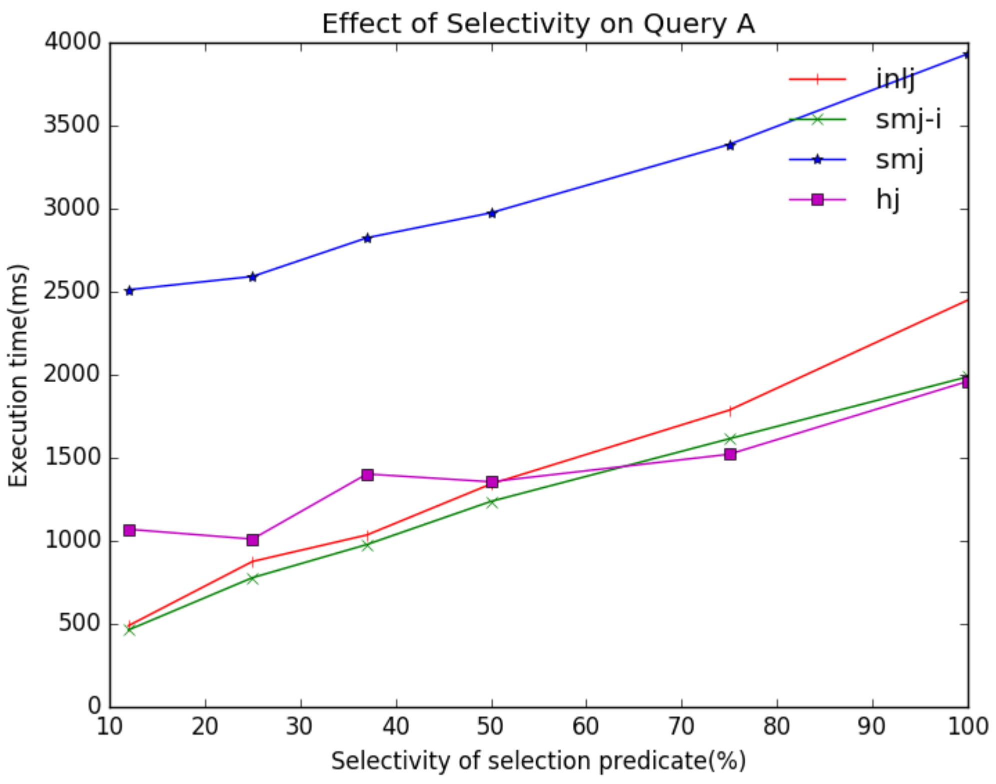
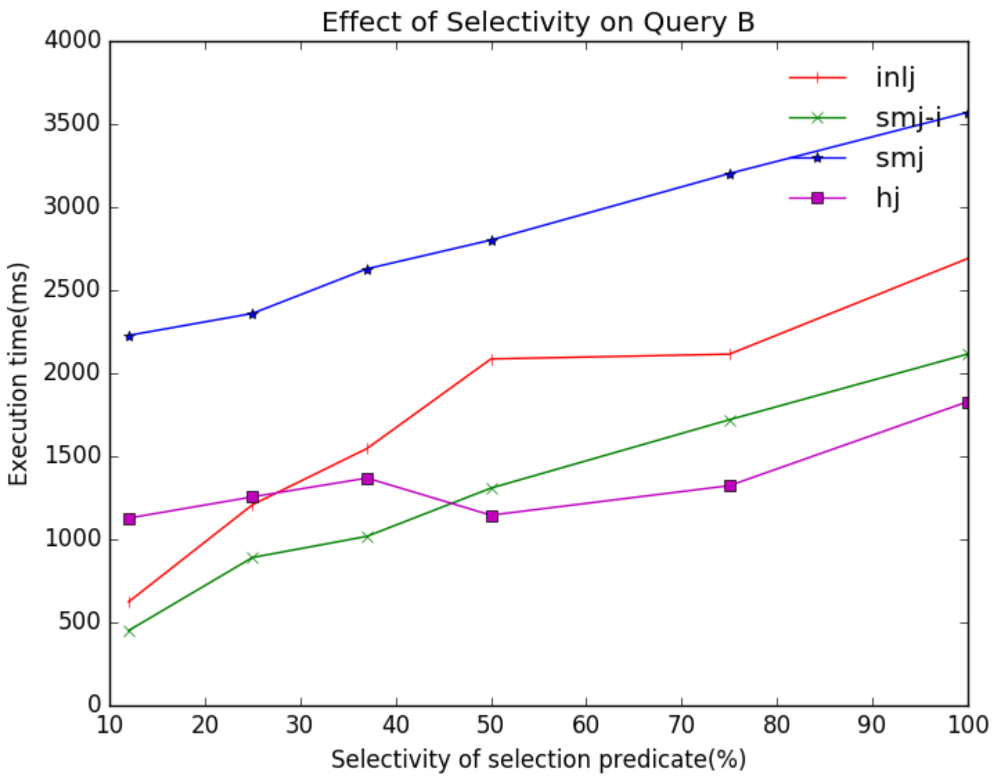
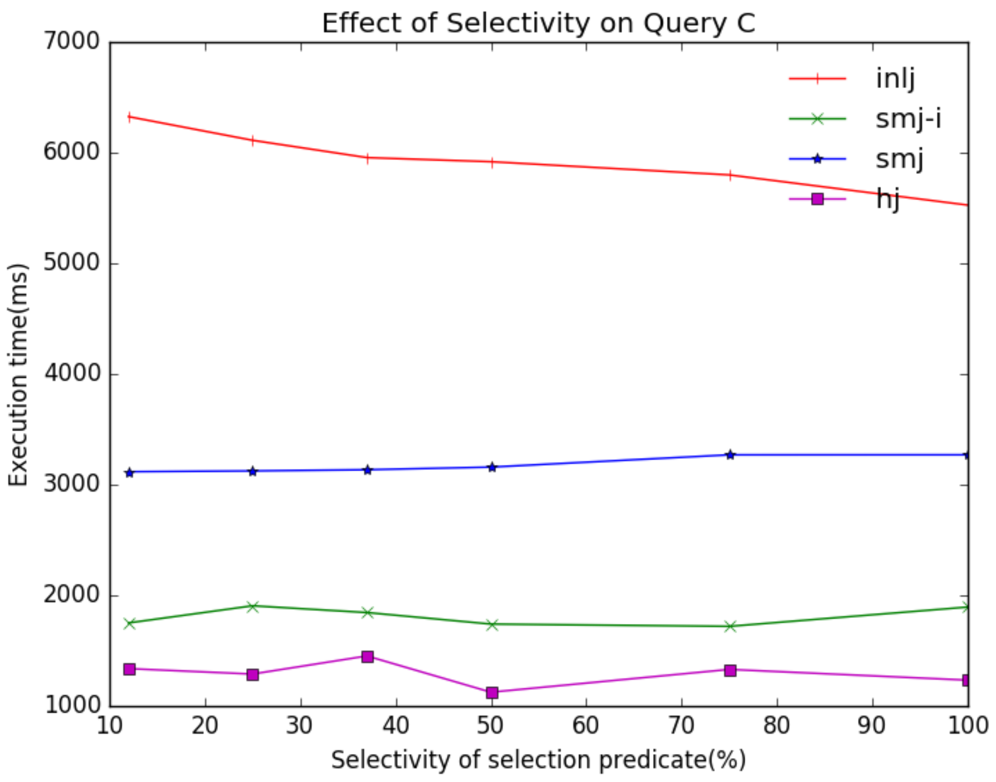
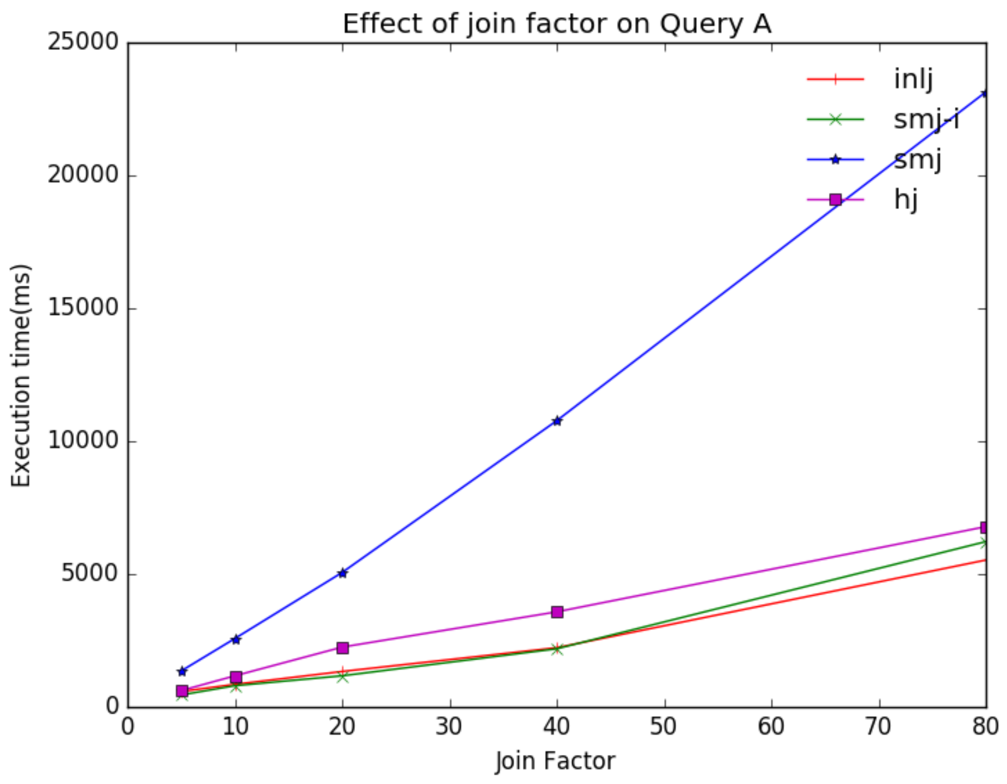
Experiment 1

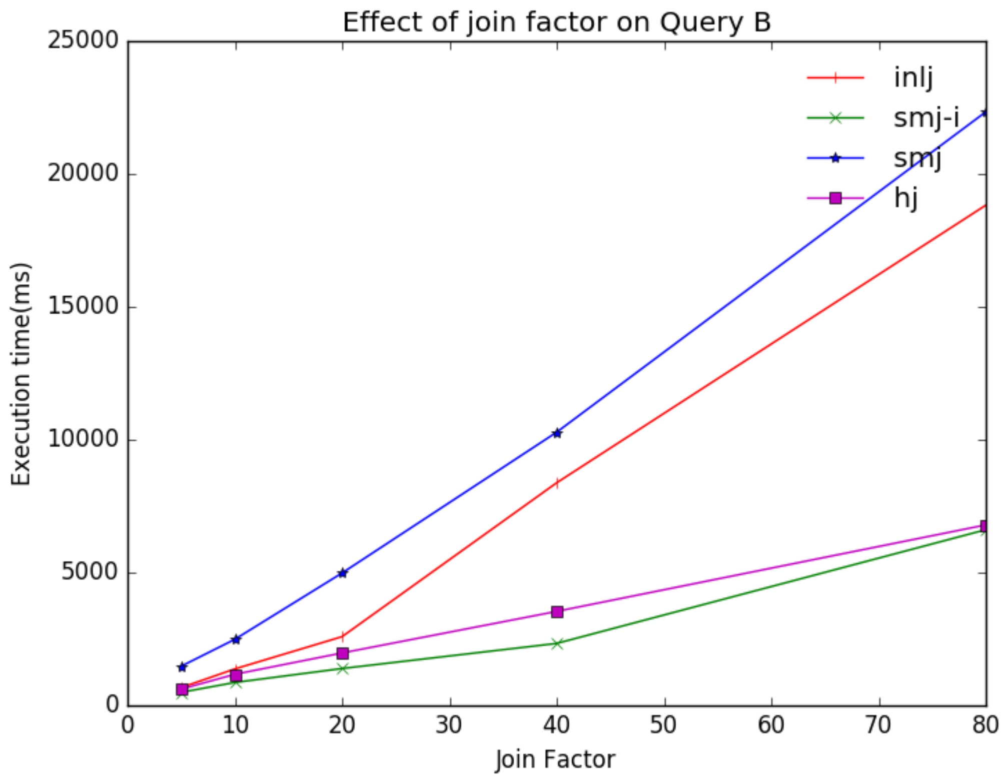


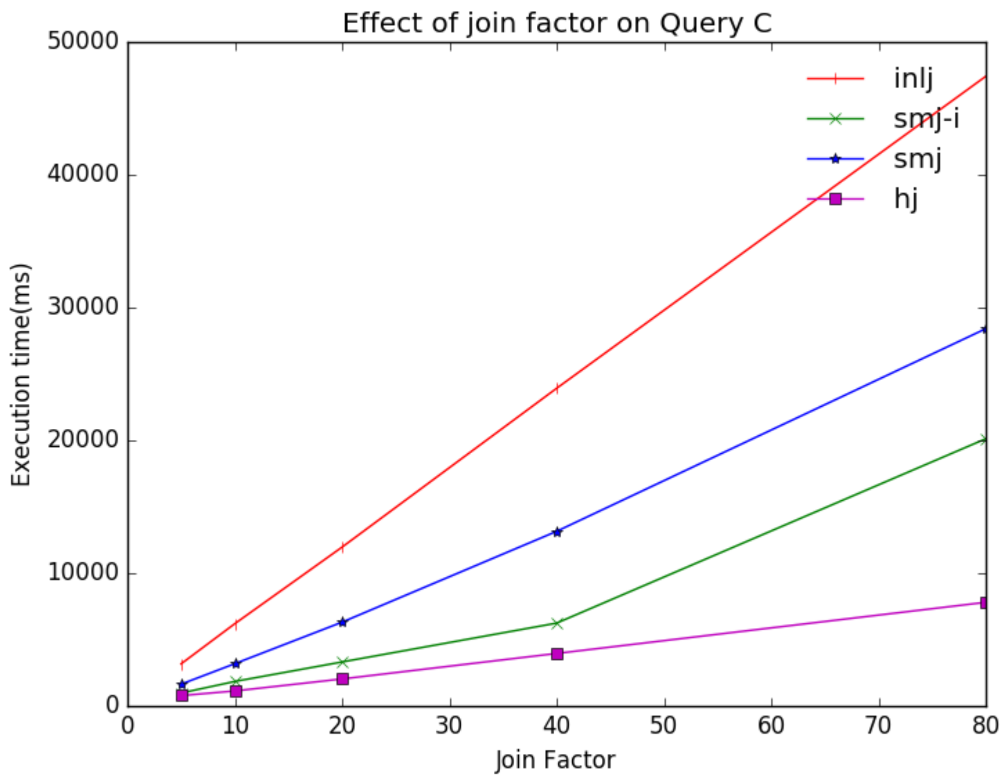



Effect of Selectivity on Query C
Execution time(ms)
Selectivity of selection predicate(%)
inlj
smj-i
smj
hj
1000
2000
3000
4000
5000
6000
7000
10
20
30
40
50
60
70
80
90
100

Experiment 2

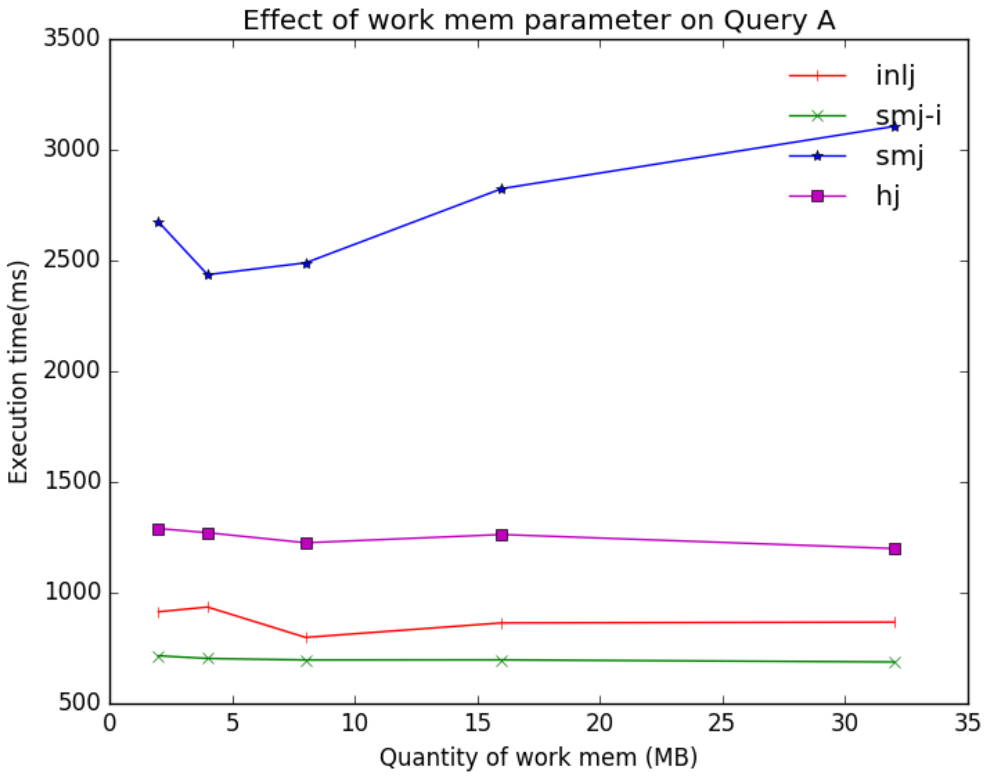


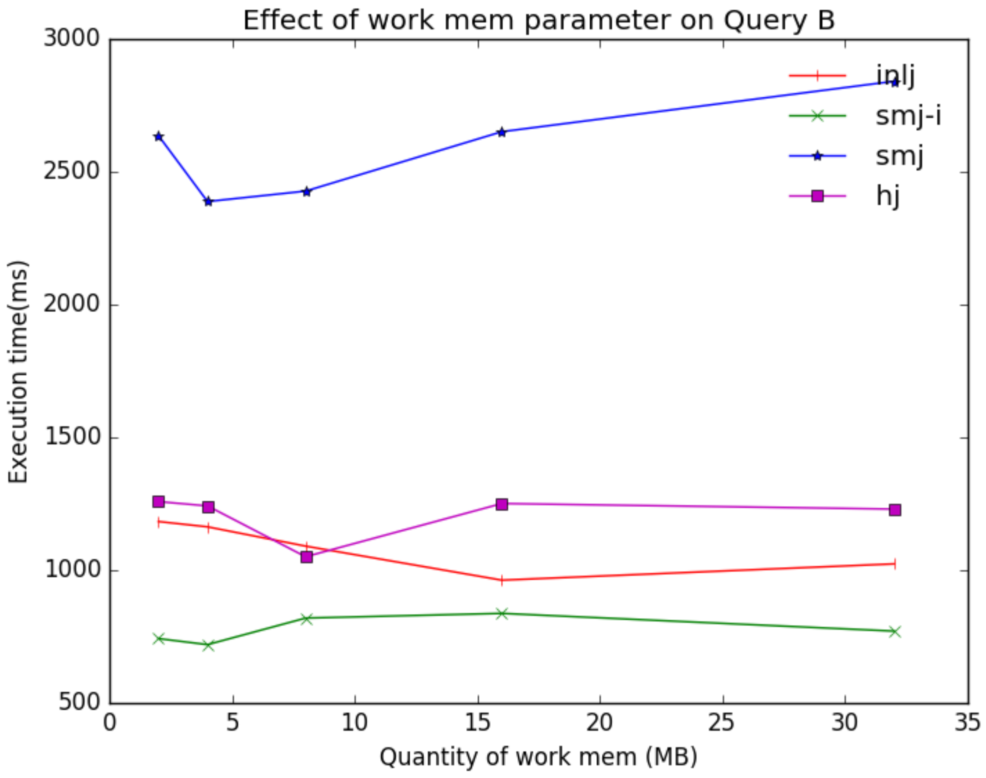


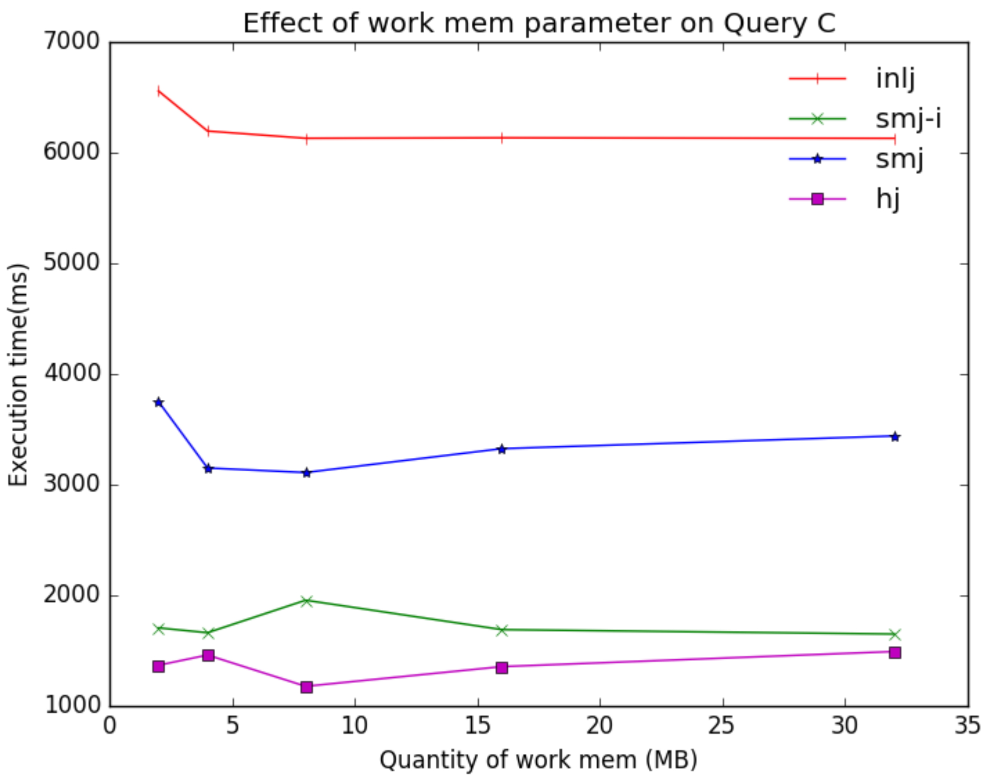


Effect of join factor on Query C
Execution time(ms)
Join Factor
50000
40000
30000
20000
10000
0
0
10
20
30
40
50
60
70
80
inlj
smj-i
smj
hj

Experiment 3






Effect of work mem parameter on Query C
Execution time(ms)
Quantity of work mem (MB)
inlj
smj-i
smj
hj
7000
6000
5000
4000
3000
2000
1000
0
5
10
15
20
25
30
35

Effect of selectivity of predicate on queries (expt 1)
From the graph, we can see that increasing in selectivity (more is selected) generally leads to a longer execution time for query A and B, but does not increase execution time for query C. However, at lower selectivity (less is selected), query C takes longer than A and B. The difference can be explained by some optimisations that the query optimiser use to optimise query A and B, like hash aggregate and unique operator, but not C.

Effect of join factor on queries (expt 2)
Across the board, higher join factor results in longer execution time. However, different algorithms increase execution time at a different rate for different queries. For query C, INLJ algorithm performs the worst as join factor increases. This is because of the lack of optimisation as mentioned previously. Query B also exhibit similar growth in execution time for INLJ. Even though, the number of rows scanned in query B is similar to query A, query B takes far longer in the `nested loop` stage. This can be due to the need to compute hashes for each row for the semi join algorithm.

Effect of work_mem on queries (expt 3)
The effect of work_mem on queries are mixed. For HJ , SMJ-I, and INLJ, the effect of increasing work_mem appears negligible, and only has moderate impact on query C – even then, only when work_mem is increased from 2MB to 4MB. This can be because the low memory need of these algorithms. For example, hash tables only need ~1000KB of memory. Interestingly, SMJ exhibit a U-curve graph for query A and B. The graph shows that increasing memory from 2-to-4-to-8MB helps with execution time. However, further increase leads to longer execution time. For the sort key on s.y, external merge sort is utilised. For the left part of the graph, the decrease in execution time can be explained by the drastic drop in temp reads/writes from 2000+/4000+ respectively to 600+/2500+ as we increase the work_mem from 2 to 4/8MB, but temp reads increase again as we increase work_mem. Given that the external merge sort only requires ~21MB of disk space, this seems to be able to comfortably fit in the work_mem when we set it to 32MB. However, the query optimiser still decide to carry out an external merge sort. One reason could be that the query optimiser is ‘reserving’ work_mem for future operations and thus conservatively choosing to perform external merge sort for key s.y.