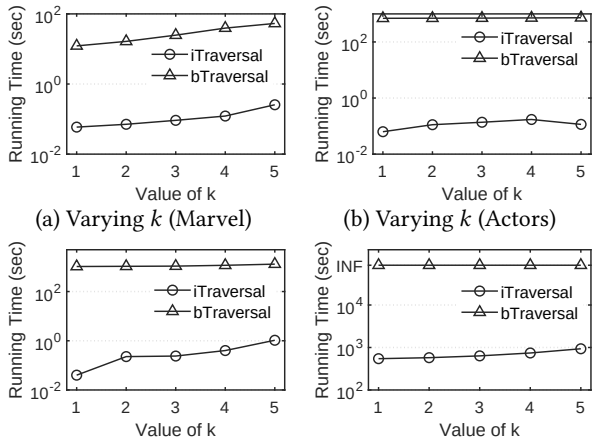
A ADDITIONAL EXPERIMENTAL RESULTS

We present the additional experimental results and performance study in this section.

A.1 Comparison among Algorithms

Additional results of running time (real datasets). The additional results of running time when returning the first 1,000 MBPs are shown in Figure 15. In general, they show the similar clues with those on Writer and DBLP, and `iTraversal` is up to four orders of mannitude faster than `bTraversal` on additional datasets.



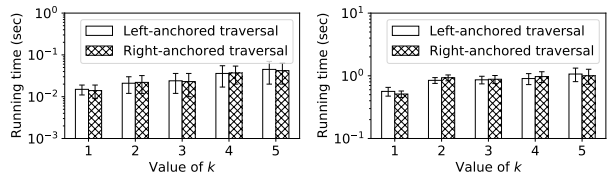
(a) Varying k (Marvel) (b) Varying k (Actors)

(c) Varying k (IMDB) (d) Varying k (Google)

Figure 15: Additional results of running time (real datasets)

A.2 Performance Study of `iTraversal`

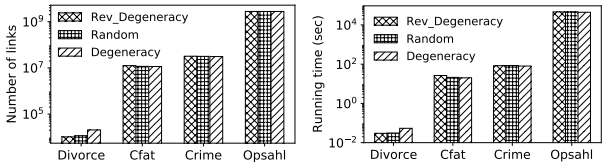
Left-anchored traversal vs Right-anchored traversal (Additional results). The results are shown in Figure 16. We compare these two options by varying k and measuring the running of time of returning the first 1,000 MBPs. We observe from the results that these two options have similar results.



(a) Varying k (Writer) (b) Varying k (DBLP)

Figure 16: Left-anchored vs right-anchored traversals

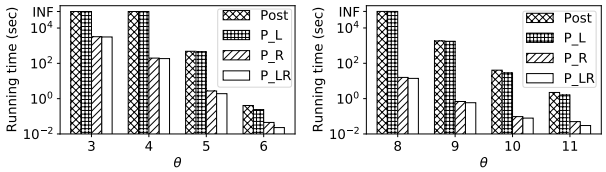
Performance study of orderings of vertices (Additional study). We note that `iTraversal` runs based on a pre-set ordering of vertices. For example, when forming almost-satisfying graphs or extending a local solution to be a maximal k-biplex, vertices are iteratively included based on an underlying ordering. Therefore, we study the effects of different orderings of vertices, namely, (1) `Random`: a random ordering of vertices, (2) `Degeneracy`: the degeneracy ordering that can be obtained by executing the existing algorithm for topological ordering, and (3) `rev-Degeneracy`: the reverse degeneracy ordering. We measure the number of links of the solution graph and the running time. The results are shown in Figure 17. We observe that these orderings have similar performances. Moreover, no heuristic metric motivates us to select a specific ordering of vertices.



(a) Solution graphs ($k = 1$). (b) Running time ($k = 1$).

Figure 17: Comparisons among different orderings.

Performance study of `iTraversal` for enumerating large MBPs (Additional study). We consider two types of pruning techniques proposed in Section 5, and compare four different versions of `iTraversal` for enumerating large MBPs, namely (1) `Post`: `iTraversal` without any pruning technique, (2) `P_L`: `iTraversal` with the left-side pruning, (3) `P_R`: `iTraversal` with the right-side pruning, and (4) `P_LR`: `iTraversal` with the left-side and right-side pruning. The results of running time are shown in Figure 18. We see that `P_L` and `P_R` are both faster than the basic version `Post`. `P_L` achieves small improvements compared with `P_R`. This is because the left-side pruning, derived based on the exclusion strategy, only works when the size of the set $\mathcal{L}\backslash\mathscr{E}(H)$ is less than θ. Moreover, `P_LR` combining all pruning techniques performs the best.



(a) Varying θ (k=1, Writer) (b) Varying θ (k=1, DBLP)

Figure 18: Comparison among pruning for listing large MBs.

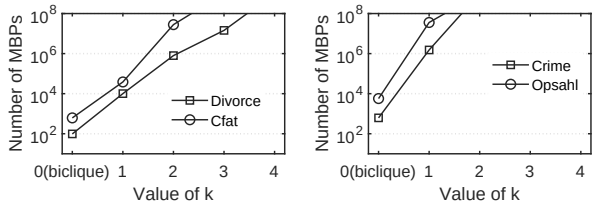


Figure 19: Results of # of MBPs (varying k)

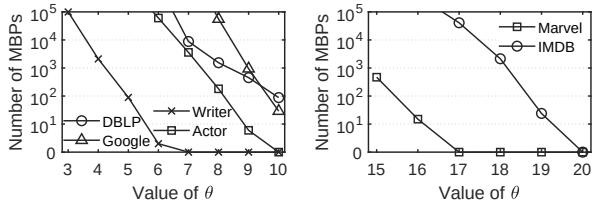


Figure 20: Results of # of large MBPs (varying θ)

A.3 Dataset Statistics

We finally analysis the properties of datasets used, according to the number of MBPs w.r.t. different values of k and θ. Since enumerating all MBPs within large datasets is time-consuming, we report the total number of MBPs (w.r.t. different k) only for four small datasets (i.e., Divorce, Cfat, Crime and Opsahl), and report the total number of large MBPs (w.r.t. different θ) for the remaining six large datasets. We show the results of the effect of varying k and θ on the number of large MBs in Figure 19 and Figure 20, respectively. In general, we have the number of (large) MBPs exponentially increase and decrease as k or θ grows.