Approximating Probabilistic Group Steiner Trees in Graphs: Supplemental Materials

This supplement is available online [1]. The road map of this supplement is as follows. In Section S1, we discuss that the existing dynamic programming methods for solving the classical group Steiner tree problem do not suit finding optimal solutions to the probabilistic group Steiner tree problem. In Section S2, we conduct additional experiments where edge weights are defined as pairwise Jaccard distances.

S1. SOME DISCUSSION ON THE DYNAMIC PROGRAMMING APPROACH TO FINDING GROUP STEINER TREES

The experiments in the main content indicates that two dynamic programming algorithms, DPBF in [2] and PrunedDP++ in [3], can solve the classical group Steiner tree problem to optimality within reasonable amounts of time. With this in mind, one may wonder whether we can extend these dynamic programming algorithms to efficiently solve the probabilistic group Steiner tree problem to optimality. Here, we show that, since the probabilistic group Steiner tree problem has a harsher NP-hard condition than the classical problem, we are unable to do this. The details are as follows.

We describe the dynamic programming model behind DPBF and PrunedDP++ for solving the classical group Steiner tree problem as follows. Let $\mathbf{p}$ be a set of vertex groups. Let $T(v, \mathbf{p})$ be a minimum-weight tree that roots at vertex v and contains at least one vertex in each group in $\mathbf{p}$. The dynamic programming model behind DPBF and PrunedDP++ is as follows (details in [2]).

$$T(v, \mathbf{p}) = \min\{T_g(v, \mathbf{p}), T_m(v, \mathbf{p})\}, \tag{S1}$$

where

$$T_g(v, \mathbf{p}) = \min_{u \in N(v)} \{(v, u) \cup T(u, \mathbf{p})\}, \tag{S2}$$

$$T_m(v, \mathbf{p}_1 \cup \mathbf{p}_2) = \min_{\mathbf{p}_1 \cap \mathbf{p}_2 = \emptyset} \{T(v, \mathbf{p}_1) \cup T(v, \mathbf{p}_2)\}, \tag{S3}$$

and $N(v)$ is the set of adjacent vertices of v. In Equation (S1), $T_g(v, \mathbf{p})$ is a tree generated via a grow process, while $T_m(v, \mathbf{p})$ is a tree generated via a merge process. Equation (S2) describes the grow process: $T_g(v, \mathbf{p})$ is generated by combining edge (v, u) with $T(u, \mathbf{p})$ for such $u \in N(v)$ that the weight of the combined tree is minimal. Equation (S3) describes the merge process: $T_m(v, \mathbf{p}_1 \cup \mathbf{p}_2)$ is generated by combining $T(v, \mathbf{p}_1)$ and $T(v, \mathbf{p}_2)$ for such $\mathbf{p}_1 \cap \mathbf{p}_2 = \emptyset$ that the weight of the combined tree is minimal. We show an example in Figure S1, where $\Gamma = \{g_1, g_2\}$, $g_1 = \{v_1, v_4\}$, $g_2 = \{v_2, v_3\}$, all edge weights are 1, and edge (v_1, v_2) is the optimal solution to the classical group Steiner tree problem. Let $\mathbf{p}_1 = \{g_1\}$, $\mathbf{p}_2 = \{g_2\}$, $\mathbf{p} = \{g_1, g_2\}$. In DPBF or PrunedDP++, for each vertex in a group, we initialize this single vertex as the minimum-weight tree that roots at itself and contains at least one vertex in this group. For example, we initialize $T(v_1, \mathbf{p}_1) = \{v_1\}$ and $T(v_2, \mathbf{p}_2) = \{v_2\}$. Then, we grow $T_g(v_2, \mathbf{p}_1) = (v_1, v_2) \cup T(v_1, \mathbf{p}_1) = (v_1, v_2)$. We merge $T(v_2, \mathbf{p}) = T_g(v_2, \mathbf{p}_1) \cup T_g(v_2, \mathbf{p}_2) = (v_1, v_2)$, which is the found optimal solution to the classical group Steiner tree problem.

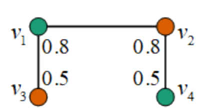
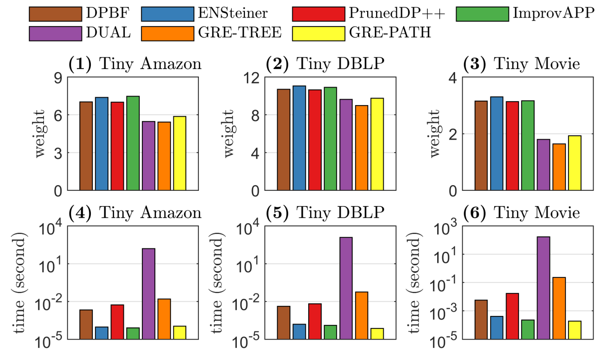
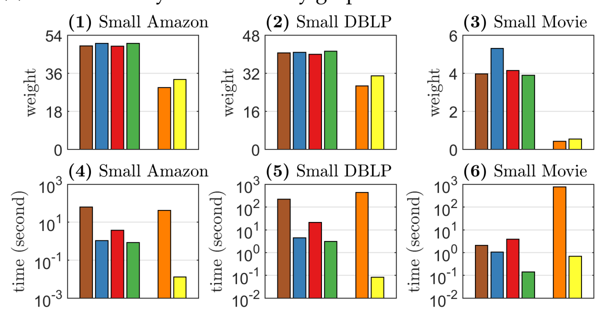


Fig. S1. An illustration of the dynamic programming approach to finding group Steiner trees.

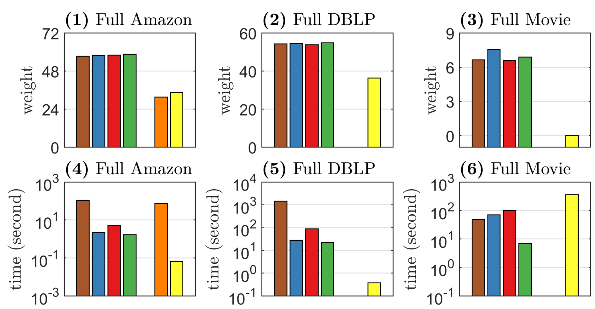
Subsequently, let us consider the probabilistic case. Let $p_{g_1}(v_1) = p_{g_2}(v_2) = 0.8$, $p_{g_2}(v_3) = p_{g_1}(v_4) = 0.5$, and $b = 0.9$. The whole input graph G in Figure S1 is the optimal solution to the probabilistic group Steiner tree problem. Intuitively, the extension of the above dynamic programming model to the probabilistic case would be to let $T(v, \mathbf{p})$ be a minimum-weight tree that roots at vertex v and satisfactorily covers every group in $\mathbf{p}$. Then, for each vertex in a group, we need to initialize the minimum-weight tree that roots at this vertex and satisfactorily covers



(a) DUAL can only be used in tiny graphs with dozens of vertices.



(b) GRE-TREE does not scale well for small DBLP and Movie.



(c) Experiment results in full graphs.

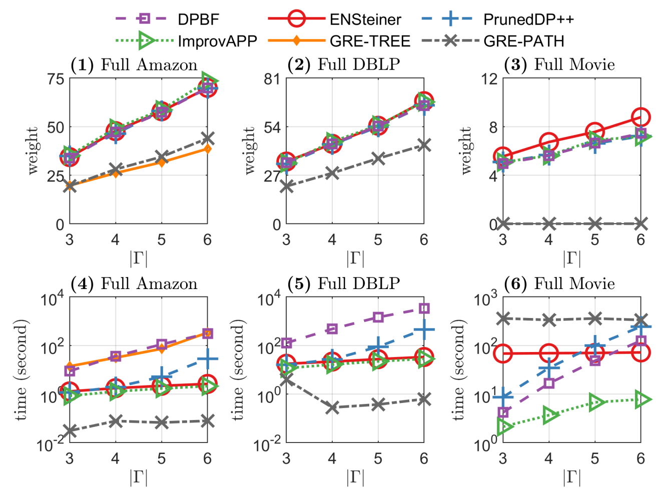
Fig. S2. Experiment results in graphs with different sizes.

this group. However, Theorem 1 in the main content shows that the probabilistic group Steiner tree problem is NP-hard even when $|\Gamma| = 1$, which means that it is NP-hard to initialize each of the above trees in the probabilistic case, *e.g.,* it is NP-hard to initialize $T(v_1, \mathbf{p}_1)$ and $T(v_2, \mathbf{p}_2)$ in Figure S1. In comparison, it is trivial to initialize such trees in the classical case, since the classical problem is trivial when $|\Gamma| = 1$ (notably, in the probabilistic case, each initialized tree may contain multiple vertices, while in the classical case, each initialized tree only contains a single vertex). As a result, it is too slow to conduct the initialization process in the probabilistic case. Therefore, due to the harsher NP-hard condition of the probabilistic group Steiner tree problem, we cannot extend the above dynamic programming model to efficiently solve the probabilistic group Steiner tree problem to optimality.

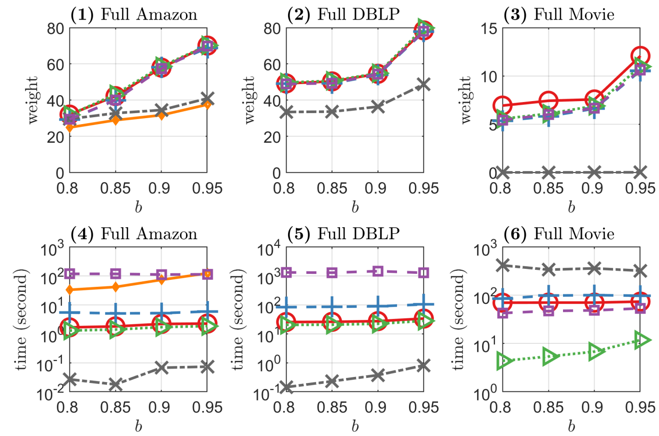
S2. ADDITIONAL EXPERIMENT RESULTS

In the experiments in the main content, we set edge weights to 1. Here, we conduct additional experiments by setting edge weights to pairwise Jaccard distances (*e.g.,* [4, 5]), *i.e.,* for edge e between vertices u and v, set the weight of e as $c(e) = 1 - \frac{|V_u \cap V_v|}{|V_u \cup V_v|}$, where V_u and V_v are the sets of vertices adjacent to u and v, respectively. We show that the experiment conclusions in the main content also hold for the following additional experiments.

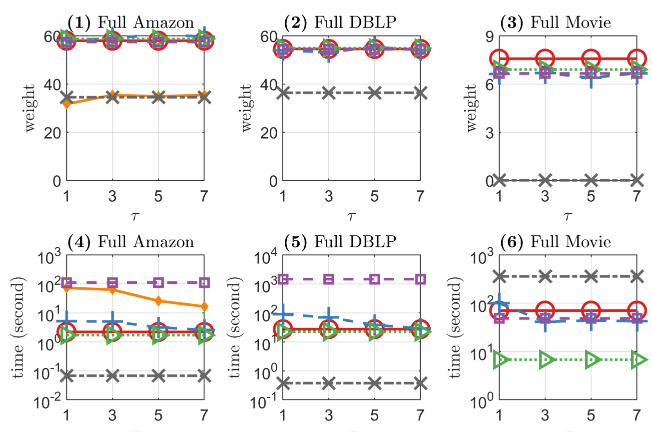
DUAL **is mainly of theoretical interests.** First, we show that DUAL can only be used in tiny graphs with dozens of vertices in Figure S2a, where $|V| = 45$ for "Tiny Amazon", $|V| = 90$ for "Tiny DBLP", and $|V| = 70$ for "Tiny Movie". We observe that, in Figures S2a (4-6), DUAL is significantly slower



(a) Variation of the number of vertex groups: $|\Gamma|$.



(b) Variation of the threshold value: b.

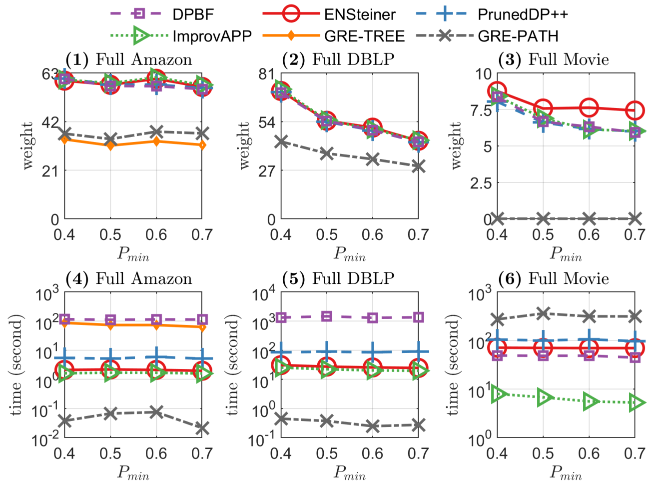


(c) Variation of the approximation parameter: τ.

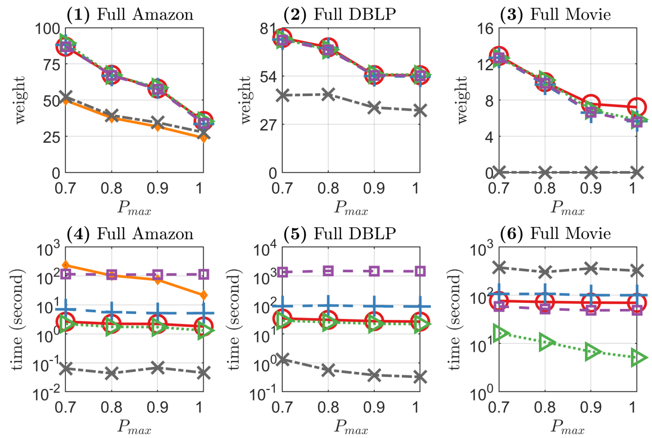
Fig. S3. Experiment results of varying $|\Gamma|$, b, and τ.

than the other algorithms. These experiments show that DUAL can only be used in tiny graphs with dozens of vertices, and is mainly of theoretical interests. Due to this reason, we do not apply DUAL in the following experiments.

GRE-TREE **is useful when group sizes are small.** We evaluate the performance of GRE-TREE in Figure S2b, where $|V| = 188,552$ for "Small Amazon", $|V| = 448,891$ for "Small DBLP", and $|V| = 2,423$ for "Small Movie". We observe that GRE-TREE is significantly slower than the other



(a) Variation of the minimum positive probability value: P_{min}.



(b) Variation of the maximum positive probability value: P_{max}.

Fig. S4. Experiment results of varying P_{min} and P_{max}.

algorithms for Movie, since group sizes are large for Movie. Like the experiments in the main content, it is too slow to implement GRE-TREE in the full DBLP and Movie graphs. Thus, we only implement GRE-TREE for Amazon, but not for DBLP and Movie, in the following experiments.

Experiment results in full graphs. We evaluate the solution quality and speed of algorithms using the full datasets in Figure S2c. We observe that, in Figures S2c (1-3), the solution weights of GRE-TREE and GRE-PATH are significantly lower than those of the baseline algorithms. This shows the effectiveness of GRE-TREE and GRE-PATH for finding probabilistic group Steiner trees. Like the experiments in the main content, GRE-PATH has a higher efficiency than the baseline algorithms for Amazon and DBLP, but a lower efficiency than the baseline algorithms for Movie.

Variation of the number of vertex groups: $|\Gamma|$. We vary the number of vertex groups: $|\Gamma|$ in Figure S3a. Like the experiments in the main content, in Figures S3a (1-3), the solution weights increase with $|\Gamma|$, and the superior solution qualities of the proposed GRE-TREE and GRE-PATH over the baseline algorithms hold well as $|\Gamma|$ varies. Also like the experiments in the main content, in Figures S3a (4-6), DPBF, PrunedDP++ and GRE-TREE do not scale well to $|\Gamma|$, while ENSteiner, ImprovAPP and GRE-PATH have stronger scalabilities to $|\Gamma|$. Notably, the reason why the running times of GRE-PATH may decrease with $|\Gamma|$, *e.g.,* Figure S3a (5), is that the time complexity of GRE-PATH is proportional to $|g_{min}|$, which decreases with $|\Gamma|$.

Variation of the threshold value: b. We vary the threshold value: b in Figure S3b. Like the experiments in the main content, the solution weights generally increase with b, and the running times of the proposed algorithms may increase with b. Also like the experiments in the main content, in Figures S3b (4-6), the running times of baseline algorithms often do not change much with b. Differently, the running time of ImprovAPP increases with b in Figure S3b (6). The reason is that it first finds a classical group Steiner tree without considering b, and then merging shortest paths into this tree for satisfactorily covering all vertex groups. Since the running time of finding a classical group Steiner tree is small for Movie, while the running time of the merging process increases with b, the running time of ImprovAPP increases with b for Movie.

Variation of the approximation parameter: τ. We vary the parameter τ in GRE-TREE and PrunedDP++ in Figure S3c. Like the experiments in the main content, in Figure S3c (1), the solution weight of of GRE-TREE increases with τ. Also like the experiments in the main content, in Figures S3c (4-6), the running times of GRE-TREE and PrunedDP++ decrease with τ.

Variation of the minimum and maximum positive probability values: P_{min} and P_{max}. We vary the minimum and maximum positive probability values: P_{min} and P_{max} in Figure S4. Like the experiments in the main content, in Figures S4a (1-3) and S4b (1-3), the solution weights often decreases with P_{min} and P_{max}. In figures S4b (4-6), the running times of GRE-TREE and ImprovAPP may decrease with P_{max}, as these two algorithms may merge fewer trees or paths as probability values increase.

REFERENCES FOR THE SUPPLEMENT

1. "Supplement," (2022). https://github.com/rucdatascience/PGST/blob/main/Supplement.pdf.
2. B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding top-k min-cost connected trees in databases," in *IEEE International Conference on Data Engineering,* (IEEE, 2007), pp. 836–845.
3. R.-H. Li, L. Qin, J. X. Yu, and R. Mao, "Efficient and progressive group Steiner tree search," in *Proceedings of the 2016 International Conference on Management of Data,* (ACM, 2016), pp. 91–106.
4. T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining,* (ACM, 2009), pp. 467–476.
5. Y. Sun, X. Xiao, B. Cui, S. Halgamuge, T. Lappas, and J. Luo, "Finding group steiner trees in graphs with both vertex and edge weights," Proc. VLDB Endow. **14**, 1137–1149 (2021).