Maximizing Submodular Functions under Submodular Constraints (Supplementary Material)

Madhavan R. Padmanabhan[1] **Yanhui Zhu**[1] **Samik Basu**[1] **A. Pavan**[1]

[1]Department of Computer Science,

Iowa State University,

Ames, Iowa, USA

A MISSING PROOFS

OBSERVATION 1. *Given a submodular function g and θ, it is NP-Hard to calculate K_g where $K_g = max\{|X| : g(X) \leq \theta$ and $|X| \leq k\}$.*

Proof. Let $K'_g = max\{|X| : g(X) \leq \theta\}$. We will first show that computing K'_g is NP-Hard. Consider the knapsack problem: We have n elements $S = \{1, 2, ..n\}$ with values $\{v_1, v_2, .., v_n\}$ and weights $\{w_1, w_2, .., w_n\}$. Given a capacity W, and a target γ, is there a set $S' \subseteq S$ such that $\sum_{i \in S'} v_i \geq \gamma$ and $\sum_{i \in S'} w_i \leq W$.

For each element i, construct a complete bipartite graph $G_i = (L_i, R_i, E_i)$ where $L_i = \{x_{i_1}, x_{i_2}, .., x_{iv_i}\}$ and $R_i = \{y_{i_1}, y_{i_2}, ..., y_{iw_i}\}$. L_i and R_i represent the values and weights of element i respectively. Let $L = \cup L_i$, $R = \cup R_i$ and $E = \cup E_i$. Let $G = (L, R, E)$. We define the neighborhood function $g(X)$ for all $X \subseteq L$ as the number of neighbors of X. $g(X)$ is known to be a submodular function. Assume that we can compute $K'_g = max\{|X| : g(X) \leq W\}$. Initially, let $X = \phi$. For a graph G_i, if we add a single node from L_i to X, we can add every node in L_i such that $g(X)$ increases by w_i. This is equivalent to selecting an element i, where the number of elements added increases the value by v_i and has a weight w_i. If $K'_g \geq \gamma$ then there exists $S' \subseteq S$ such that $\sum_{i \in S'} v_i \geq \gamma$ and $\sum_{i \in S'} w_i \leq W$. Next, we show that if S' exists then $K'_g \geq \gamma$. Let $S' = \{z_1, z_2., ..., z_l\}$. Based on our reduction we can construct $X = \{L_{z_1} \cup L_{z_2} .. \cup L_{z_l}\}$. Clearly $g(X) = \sum_{i \in S'} w_i \leq W$ and $|X| = \sum_{i \in S'} v_i \geq \gamma$. Thus, $K'_g \geq \gamma$. Since computing K'_g is NP-Hard, computing K_g is NP-Hard.

□

CLAIM 1.

$$\begin{aligned} OPT_{k,\theta} - f(X_k) \quad \leq \quad & \left(\frac{O-1}{O}\right)^{k-1} (OPT_{k,\theta} - f(X_1)) \\ & + \sum_{i=1}^{k-1} MCDG_i \end{aligned}$$

Proof.

$$\begin{aligned}
OPT_{k,\theta} - f(X_k) &\leq \frac{O-1}{O}\left(OPT_{k,\theta} - f(X_{k-1})\right) \\
&\quad + MCGD_{k-1} \\
&\leq \frac{O-1}{O}\left(\frac{O-1}{O}\left[OPT_{k,\theta} - f(X_{k-2})\right]\right. \\
&\quad \left. + MCGD_{k-2}\right) + MCGD_{k-1} \\
&= \left(\frac{O-1}{O}\right)^2 \left(OPT_{k,\theta} - f(X_{k-2})\right) \\
&\quad + \frac{O-1}{O} MCGD_{k-2} + MCGD_{k-1} \\
&\leq \left(\frac{O-1}{O}\right)^{k-1} \left(OPT_{k,\theta} - f(X_1)\right) + \sum_{i=1}^{k-1} MCDG_i
\end{aligned}$$

□

THEOREM 2. *There does not exist a polynomial time algorithm $\mathcal{A}$ for SCSK and SCSK-C such that it outputs a set X with guarantee $f(X) \geq d \cdot OPT - A$ where $d < 1, A > 0$ are universal constants.*

Proof. In Iyer and Bilmes [2013], it is shown that there exists no polynomial time algorithm for SCSK that achieves a constant factor approximation. We will use this to show that there also does not exist a (d, A)-multiplicative additive approximation algorithm with constants $d < 1, A > 0$ for SCSK.

Assume that there exists an algorithm $\mathcal{A}$ such that it outputs a set $f(X) \geq d \cdot OPT - A$.

Let $f'(X) = A \cdot f(X)$. Consider the same problem with f'. The optimum solution is $OPT' = A \cdot OPT$. The guarantee is still $f'(X) \geq d \cdot OPT' - A$. Thus $A \cdot f(X) \geq d \cdot A \cdot OPT - A$ and this implies that $f(X) \geq d \cdot OPT - 1$

Thus, algorithm $\mathcal{A}$ has the guarantee with $A = 1$. We can repeat the argument by defining $f'(X) = q \cdot f(X)$ leading to guarantee $f(X) \geq d \cdot OPT - \frac{1}{q}$. $\mathcal{A}$ has (d, A)-multiplicative additive guarantee for an arbitrarily small A.

Consider an instance $\mathcal{X}$ of SCSK. Let $\mathcal{I}$ be the family of feasible sets for this instance. Let $\mathcal{I}' = \{X \;\texttt{s.t}\; f(X) < d{\cdot}OPT\}$. Let $R \in \mathcal{I}'$ such that $\forall X \in \mathcal{I}', f(R) \geq f(X)$. Let $\epsilon = d \cdot OPT - f(R)$. $\mathcal{A}$ has (d, ϵ')-multiplicative additive guarantee where $\epsilon' < \epsilon$. In such an instance, the following inequalities hold.

$$f(\mathcal{A}(\mathcal{X})) \geq d \cdot OPT - \epsilon' > d \cdot OPT - \epsilon = f(R)$$

If $f(\mathcal{A}(\mathcal{X})) > f(R)$ then it must be the case that $A(\mathcal{X}) \in \mathcal{I} \setminus \mathcal{I}'$. By definition of $\mathcal{I}'$, $f(\mathcal{A}(\mathcal{X})) \geq d \cdot OPT$. If $\mathcal{I}' = \phi$, we consider the feasible element e^* that has the maximum value on f among all feasible elements. In such cases, if we output $max\{f(e^*), f(\mathcal{A}(\mathcal{X}))\}$, we obtain a $d-$approximate algorithm for SCSK. This is a contradiction as such an algorithm does not exist for SCSK Iyer and Bilmes [2013] which invalidates our assumption. We can extend the argument for SCSK-C by varying the cardinality from $1..n$ to arrive at a similar contradiction. □

THEOREM 3. *Let X be the solution produced by Algorithm 1, then*

$$f(X) \geq \frac{1}{c_f}\left(1 - (1 - \frac{c_f}{k})^k\right) OPT_{k,\theta} - A,$$

where A is the additive error same as in Theorem 1.

Proof. We combine the ideas from the proof of Theorem 3 with that of Lemma 5.2 in Conforti and Cornuéjols [1984]. At iteration $t-1$ of the Greedy Algorithm, we extend an ordered set $X_{t-1} = \{j_1, j_2, ..j_{t-1}\}$ by adding element j_t to obtain $X_t = \{j_1, j_2, ..j_t\}$. Define $\rho_i = f(j_i | \{j_1, j_2 ... j_{i-1}\})$. Let $X^*_{k,\theta}$ be an optimal solution. $l \leq k$ be the number of iterations of the Greedy Algorithm. From Lemma 2.1 in Conforti and Cornuéjols [1984], it follows that for $t = 1..l$:

$$OPT_{k,\theta} \leq c_f \sum_{j_i \in X_t \setminus X^*_{k,\theta}} \rho_i + \sum_{j_i \in X_t \cap X^*_{k,\theta}} \rho_i + \sum_{j_i \in X^*_{k,\theta} \setminus X_t} \rho_i$$

Recall that for $i = 0..l-1$:

$$\begin{aligned} f(X_{i+1}) = &\ f(X_i) + \text{MCG}(X_i, \theta) \\ f(X'_{i+1}) = & f(X_i) + \text{MCG}(X_i, 2\theta) \end{aligned}$$

This leads to the following observation:

$$\begin{aligned} & f(X'_{i+1}) - f(X_i) \\ & = \rho_{i+1} + \text{MCG}(X_i, 2\theta) - \text{MCG}(X_i, \theta) \end{aligned}$$

We define $\gamma_{i+1} = \text{MCG}(X_i, 2\theta) - \text{MCG}(X_i, \theta)$. Let $s = |X^*_{k,\theta} \setminus X_i|$. Thus, we have:

$$\begin{aligned} OPT_{k,\theta} \quad \leq \quad & c_f \textstyle\sum_{j_i \in X_t \setminus X^*_{k,\theta}} \rho_i + \textstyle\sum_{j_i \in X_t \cap X^*_{k,\theta}} \rho_i \\ & + \textstyle\sum_{j_i \in X^*_{k,\theta} \setminus X_t} f(X'_{t+1} | X_t) \\ \leq \quad & c_f \textstyle\sum_{j_i \in X_t \setminus X^*_{k,\theta}} \rho_i \\ & + \textstyle\sum_{j_i \in X_t \cap X^*_{k,\theta}} \rho_i + (O - s) \cdot f(X'_{t+1} | X_t) \\ = \quad & c_f \textstyle\sum_{j_i \in X_t \setminus X^*_{k,\theta}} \rho_i \\ & + \textstyle\sum_{j_i \in X_t \cap X^*_{k,\theta}} \rho_i + (O - s) \cdot [\rho_{t+1} + \gamma_{t+1}] \end{aligned}$$

Fix O, l and curvature c_f. Consider all problems $\mathcal{F}_{O,l,c_f}$ that has curvature c_f, optimal solution size O and the greedy set size l. Denote $\mathcal{F}(i_1, i_2, ...i_s) \subseteq \mathcal{F}_{O,l,c_f}$ as the set of problems meeting the following two conditions

1. The greedy algorithm selects the set $X = \{j_1, j_2, ...j_l\}$ where $l \leq O$.
2. $X \cap X^*_{k,\theta} = \{j_{i_1}, j_{i_2}, ...j_{i_s}\}$ for $0 \leq s \leq l$. The greedy algorithm selects s elements from the optimal solution within the first l iterations.

The greedy algorithm produces X such that $f(X) \geq \sum_{i=1}^{l} \rho_i \geq \sum_{i=1}^{l} \rho_i + \gamma_i$. Define $q_i = \rho_i + \gamma_i$ for $0 \leq i \leq l$.

Consider the ratio : $\frac{1}{O} \cdot \left(f(X) + \sum_{i=1}^{l} \gamma_i \right)$. We see that $\frac{1}{O} \cdot \left(f(X) + \sum_{i=1}^{l} \gamma_i \right) \geq R(\{i_1, i_2 ..., i_s\})$ where $R(\{i_1, i_2 ..., i_s\})$ is defined as:

$R(\{i_1, i_2 ..., i_s\}) = \min \sum_{i=1}^{l} q_i$ subject to

$$\forall i, q_i \geq 0,\ OPT_{k,\theta} \leq c_f \sum_{j_i \in X_t \setminus X^*_{k,\theta}} q_i + \sum_{j_i \in X_t \cap X^*_{k,\theta}} q_i + (O - s) \cdot q_i$$

From Conforti and Cornuéjols [1984], it follows that $R(\{i_1, i_2 ..., i_s\}) \geq R(\phi) \geq \frac{1}{c_f} \left[1 - (1 - \frac{c_f}{O})^l\right]$. Thus, we have:

$$f(X) + \sum_{i=1}^{l} \gamma_i \geq \frac{1}{c_f} \left[1 - \left(1 - \frac{c_f}{O} \right)^l \right] \cdot OPT_{k,\theta}$$

Since $O \leq k$

$$f(X) \geq \frac{1}{c_f} \left[1 - \left(1 - \frac{c_f}{k} \right)^l \right] \cdot OPT_{k,\theta} - \sum_{i=1}^{l} \gamma_i$$

The term $\sum_{i=1}^{l} \gamma_i$ is the additive error, which equals

$$\sum_{i=0}^{\ell-1} MCG(X_i, 2\theta) - \sum_{i=0}^{\ell-1} MCG(X_i, \theta)$$

Since $\sum_{i=0}^{\ell-1} MCG(X_i, \theta)$ equals $f(X)$, the additive error term is same as the one from Theorem 3. Finally, observe that when when $\ell = k$, the term $(1 - \frac{c_f}{k})^k$ tends to e^{-c_f}. $\square$

THEOREM 5. *Let f and g be two submodular functions and where f is monotone, let $h = f - g$. Consider Algorithm $\mathcal{B}$. If the subroutine $\mathcal{A}$ can solve* SCSK-C *exactly, then Algorithm $\mathcal{B}$ produces a set S such that $h(S) \geq h(OPT) - 1$. AlgorithmB makes by making $O(\lambda)$ calls to $\mathcal{A}$, where $\lambda = k \cdot max_{e \in V} g(e)$.*

Proof. Let us first assume that the range of g is non-negative integers and let OPT be the optimal solution for h. We claim that this algorithm produces an optimal solution for DIFF-C. Let $\gamma' = g(S^*)$. Since the range of g are non negative integers, Algorithm $\mathcal{B}$ will have queried $\mathcal{A}$ with paramaters f, g, k, γ'. Let the set produced at that iteration be P^*. Assume that P^* is not an optimal solution to DIFF-C i.e. such that $f(OPT) - g(OPT) > f(P^*) - g(P^*)$. We know that $f(P^*) \geq f(OPT)$, since Algorithm $\mathcal{A}$ produces an optimal solution. In addition, $g(P^*) \leq \gamma' = g(S^*)$ leading to $-g(P^*) \geq -g(S^*)$. Therefore, $f(P^*) - g(P^*) \geq f(S^*) - g(S^*)$ which contradicts our assumption. Now suppose that the range of g is not an integer and let $g(OPT) = \gamma''$ and let $\gamma' = \lceil \gamma'' \rceil$. Let P^* be the set output by $A(f, g, k, \gamma'$. Note that $f(P^*) \geq f(OPT)$ and thus $h(P^*) \geq f(P^*) - \gamma' \geq f(OPT) - \gamma'' - 1 = h(OPT) - 1$. □

B EXTENSIVE EXPERIMENTAL RESULTS

In this section, we present the full experimental results. We empirically examine the performance of SCSK-C and DIFF-C on the application of Information Diffusion in social networks.

Information Diffusion. The diffusion of information in a social network under various probabilistic diffusion models is captured as a submodular function Kempe et al. [2003]. For a (seed) set $X \subseteq V$, the submodulaer function $f(X)$ is the expected number of users influenced by X. On the other hand, there is often some cost function g associated with each seed set. Often cost functions are submodular in nature. The goal is to find a seed set of size $\leq k$ maximize f while minimizing g. Clearly, this problem can be formulated either as SCSK-C or as DIFF-C.

For our experiments, we design the submodular function g based on the reachability. For all nodes $u \in V$ in the network, we use $R_z(u)$ to denote the set of nodes reachable from u using $\leq z$ steps/edges. For any $S \subseteq V$, we have $R_z(S) = \cup_{v \in S} R_z(v)$. We define the function $g(X) = \alpha \cdot |R_z(S)|$ where $\alpha = \frac{c \cdot m}{n}$. The values c, z are tunable parameters that can be adjusted to suit the cost function. In our experiments, we use $z = 2, c = 0.02$.

Datasets. For the application of information diffusion, we collect six directed networks to conduct experiments: NetHept Net [2009], p2p-Gnutella31 Ripeanu et al. [2002], Facebook Leskovec and Mcauley [2012], Bitcoin Kumar et al. [2016], Wikipedia Leskovec et al. [2010] and DBLP Yang and Leskovec [2012]. The number of nodes of them ranges from 3,783 to 317,080.

B.1 EXPERIMENTS FOR SCSK-C

The main objective we seek in these experiments is to demonstrate that the approximation factors can be computed efficiently, which helps to gain an understanding of the quality of the solution. For the Natural greedy algorithm(Algorithm 2, we compute the additive error produced and also study how the additive error changes as the submodular budget θ increases.

Comparison Algorithms. We compare the solutions produced by the Greedy algorithm (Algorithm 2) with two variants. Note that during each iteration of this algorithm, the entire submodular budget θ is made available. We obtain a *budget-conscious* variant of this algorithm that allows iteration i to spend at most $\theta_i < \theta$ budget. Algorithm 4 describes this strategy. By following an analysis that is very similar to that of Theorem 1, we can show that this algorithm produces a set X for such that $f(X)$ is at least $(1 - 1/e)f(OPT) - A$, and A can be computed efficiently. We use the following budget-conscious algorithms. **Equal Partition:** Use $\theta/k, 2\theta/k, \cdots \theta$ as input to the budget-conscious Greedy algorithm. **Random Partition:** Select a random sequence of thresholds to use in the budget-conscious Greedy Algorithms.

□ Basic Greedy ○ Random Partition + Equal Partition

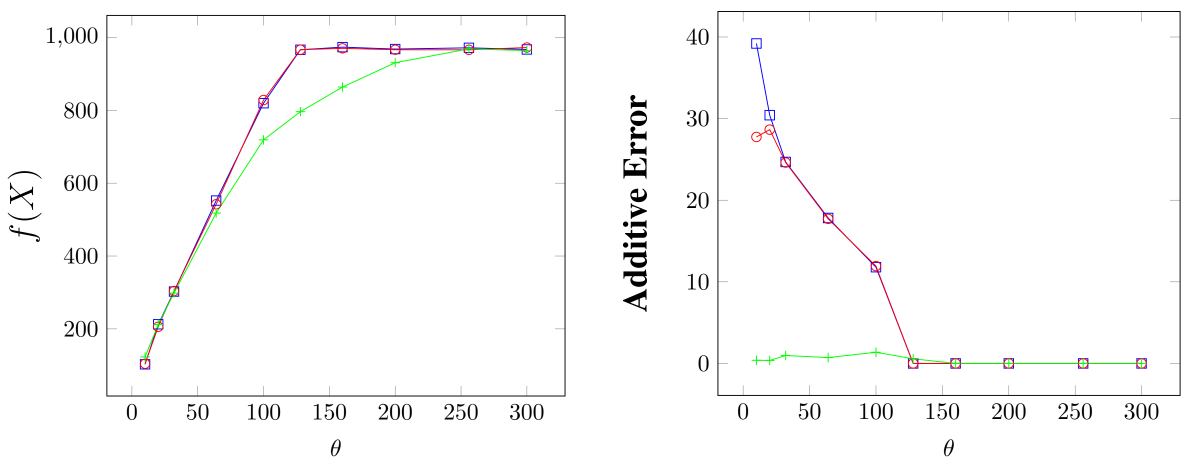


Figure 1: SCSK-C, NetHept; $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

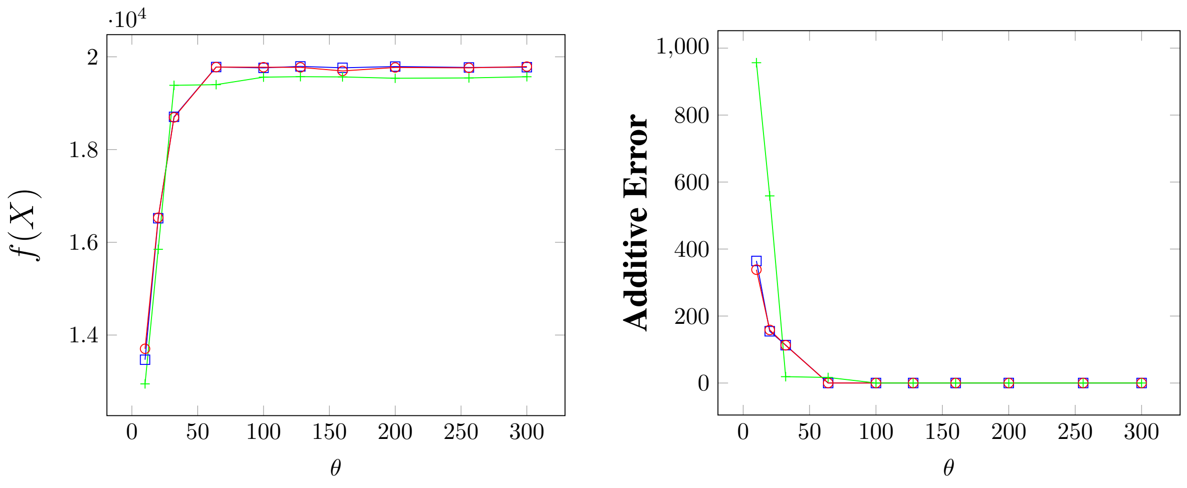
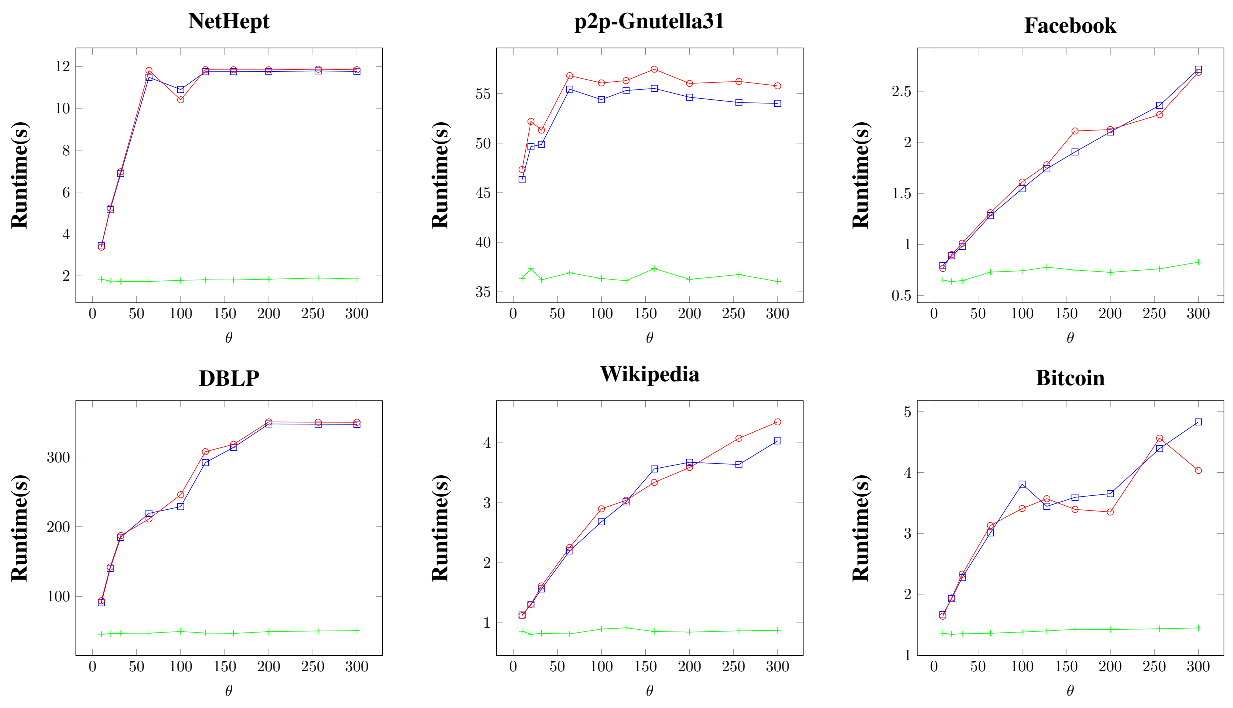


Figure 2: SCSK-C, p2p-Gnutella31, $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition



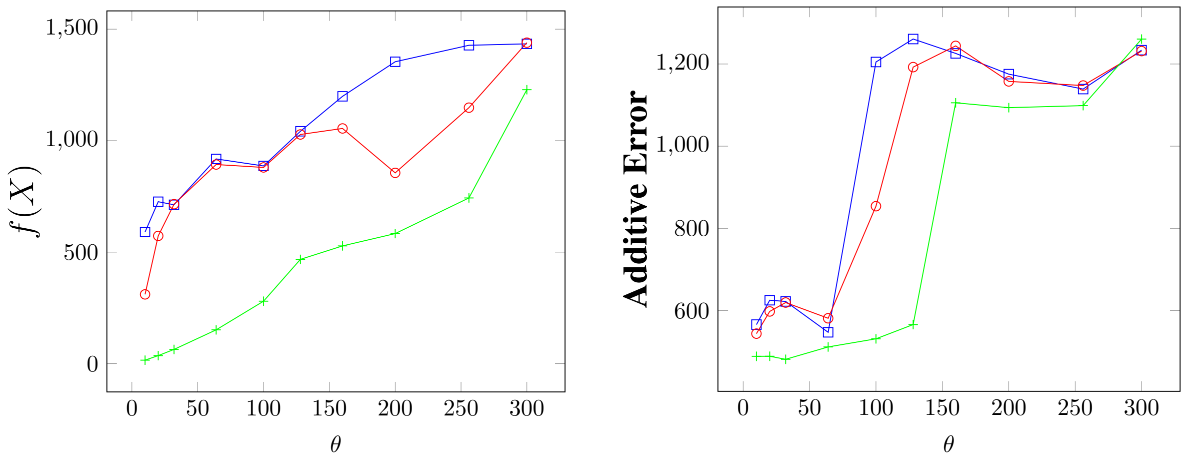


Figure 3: SCSK-C, Facebook, $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

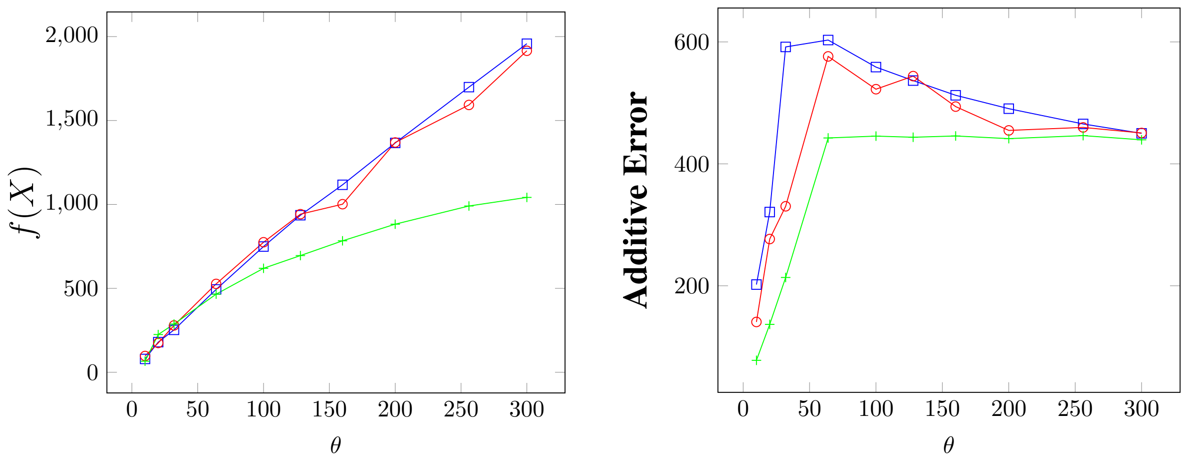


Figure 4: SCSK-C, Bitcoin, $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

Results Analyses. We chose $k = 50$ and varied the submodularity budget θ from 10 to 300. The results are shown in Figure 1 to 6. As can be seen, the Basic Greedy, Equal Partition and Random Partition algorithms produce very similar results, except for Facebook. It can be seen from Figure 1b, 2b, 6b, that as the submodular budget increases, the additive error decreases. Recall that the additive factor is approximately the difference between the quality of the seed sets produced with submodular constraints θ and 2θ. Thus all θ grows larges there may not be much difference between the constraints $g(X) \leq \theta$ and $g(X) \leq 2\theta$. It is likely that a set that satisfies the latter constraint will also satisfy the former constraint.

We now analyze the quality of the solutions produced. For NetHept, P2p-Gnutella31 and DBLP additive error is less than 10% of $f(X)$ most of the time and much smaller many times. When this happens we can conclude that for all these sets $f(X) \geq 0.53 f(OPT)$. For example, for NetHept, when $k = 50, \theta = 200$, the greedy algorithm produced a solution X of size 50, and the additive error is 0 and $f(X) = 968.21$. This implies that $f(X) \geq 0.63 f(OPT)$. Another example is DBLP, at $\theta - 20$, Basic Greedy produced a solution with value 13810 and the additive error is 1044. This implies that additive error is leass than which 7.5% of the optimal value. Thus we can be guaranteed that the value produced by the algorithm is at least $0.55 f(OPT)$. For graphs such as Facebook, Bitcoin, and Wikipedia additive errors are higher. For example, for Bitcoin with $\theta = 160$, the Basic Greedy produced a solution with value 110 whereas the additive error is 26. This implies that the value of the solution is at least $0.4 f(OPT)$. The density of the graphs could explain this phenomenon. The Average degrees of Facebook, Bitcoin and Wikipedia graphs are 43, 12, and 29 whereas for the other graphs, the average degree is less than 8.

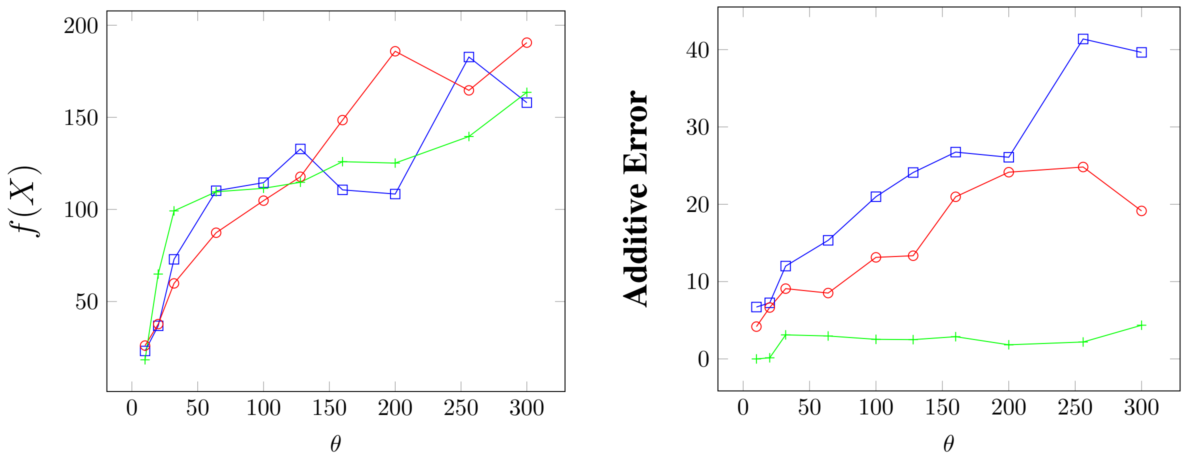


Figure 5: SCSK-C, Wikipedia, $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

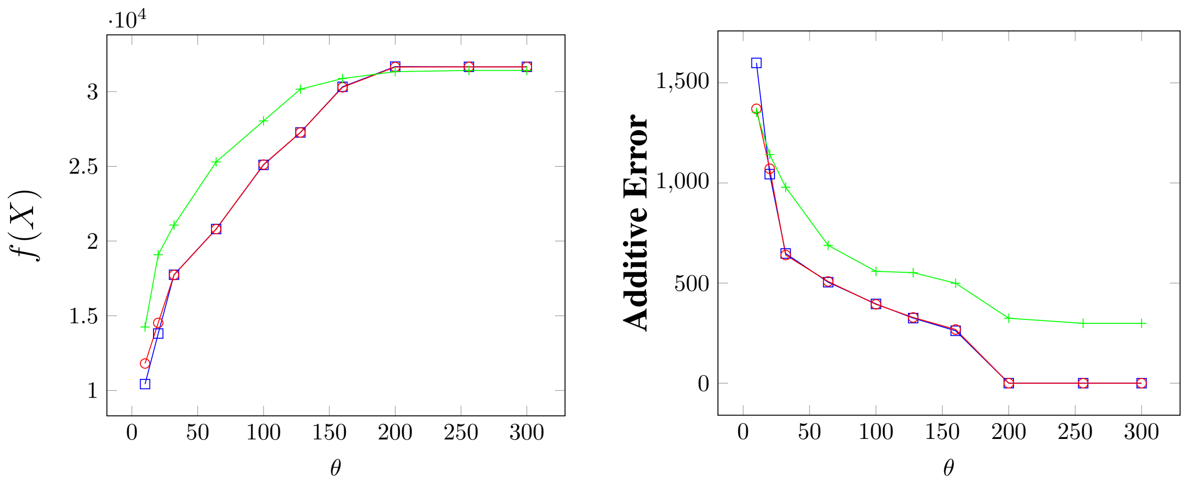


Figure 6: SCSK-C, DBLP, $k = 50$, a) θ vs. $f(X)$; b) θ vs. Additive Error with submodular cost on Basic Greedy, Random Partition and Equal Partition

For higher average degree graphs there is a larger difference between the constraints $g(X) \leq \theta$ and $g(X) \leq 2\theta$.

In terms of running time, these three algorithms are very fast. The comparisons are shown in Fig. 7. Comparing with Random Partition and Basic Greedy, Equal Partition is faster for the reason that it started from a small cost, which took a smaller number of iterations. In contrast, Random Partition can generate various cost sequences, and the submodular cost of each iteration for Basic Greedy is fixed.

B.2 EXPERIMENTS FOR DIFF-C

We use the Basic Greedy algorithm of SCSK-C as a subroutine of Algorithms LOG-APPROX and LINEAR-APPROX.

Baseline Algorithm. We compare our methods against the supSub method proposed by Iyer and Bilmes [2012]. This replaces the submodular function g with a surrogate modular function g' and attempts to maximize $f - g'$. In addition, this method iteratively updates the surrogate modular function g' the seed set until convergence. The work of Jin et al. [2021] presents the best know algorithm (called ROI-Greedy) to maximize $f - g'$, when f is submodular and g' is modular. In our implementation of supsub, we use this algorithm. We vary the cardinality constraint k from 10 to 100 to compare our LOG-APPROX and LINEAR-APPROX with supSub.

Results Analyses. As we see in Figure 9 to 13, Algorithm Log-Approx and Linear-Approx perform better than the supSub method. Interestingly, we observe that Log-Approx and Linear-Approx produced similar results on NetHept, p2p-Gnutella31, Bitcoin and DBLP. This observation suggests that when the cost function is submodular, it may suffice to just use Log-Approx since it is fast while sacrifices just a little bit of objective value. See Fig. 14 for running time comparisons. We can also see that supSub performed well on Wikipedia. However, supSub needs more time to converge on this network. Nevertheless, in general, there is still a significant gap between Log-Approx and SupSub.

□ Log-Approx ○ Linear-Approx + supSub

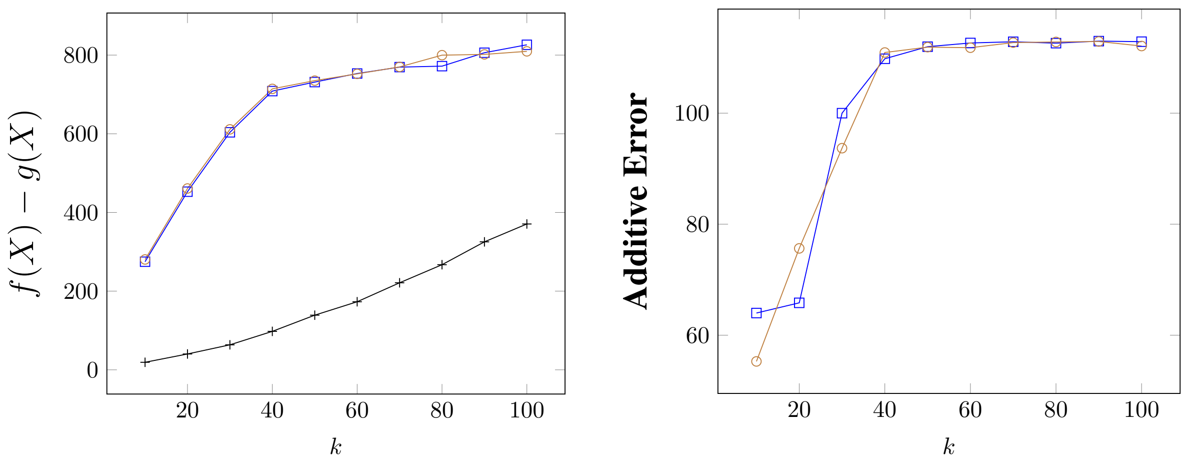


Figure 8: Diff-C, NetHept a) Budget vs Difference on Log-Approx, Linear-Approx and supSub; b) Budget vs Additive error on Log-Approx and Linear-Approx

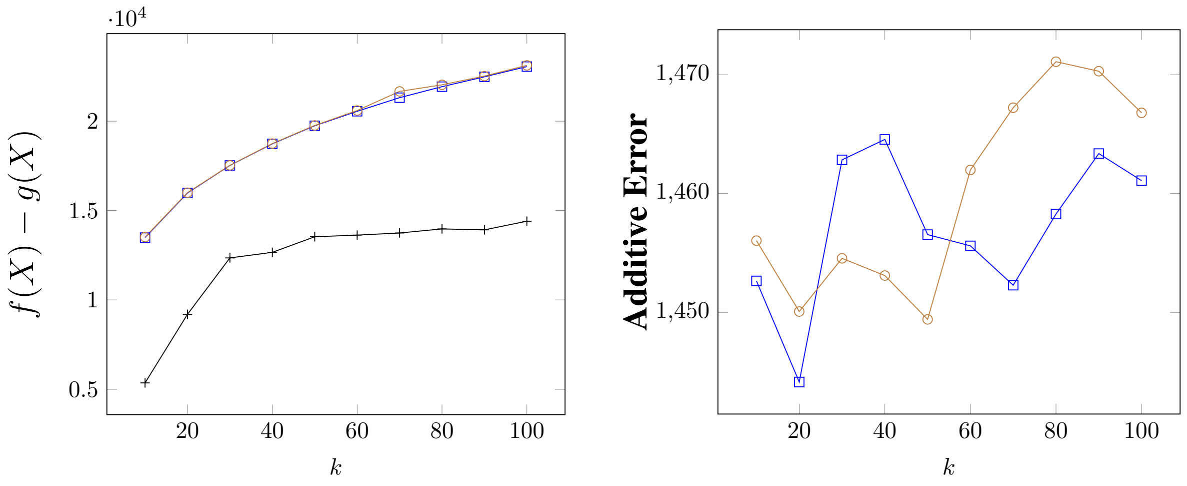


Figure 9: Diff-C, p2p-Gnutella31; a) Budget vs Difference on Log-Approx, Linear-Approx and supSub; b) Budget vs Additive error on Log-Approx and Linear-Approx

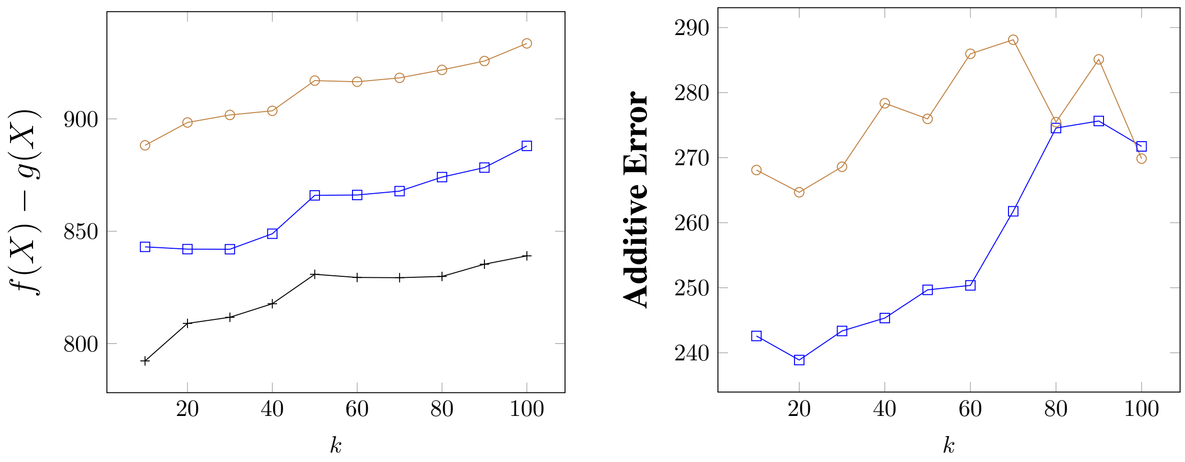


Figure 10: DIFF-C, Facebook; a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX and LINEAR-APPROX

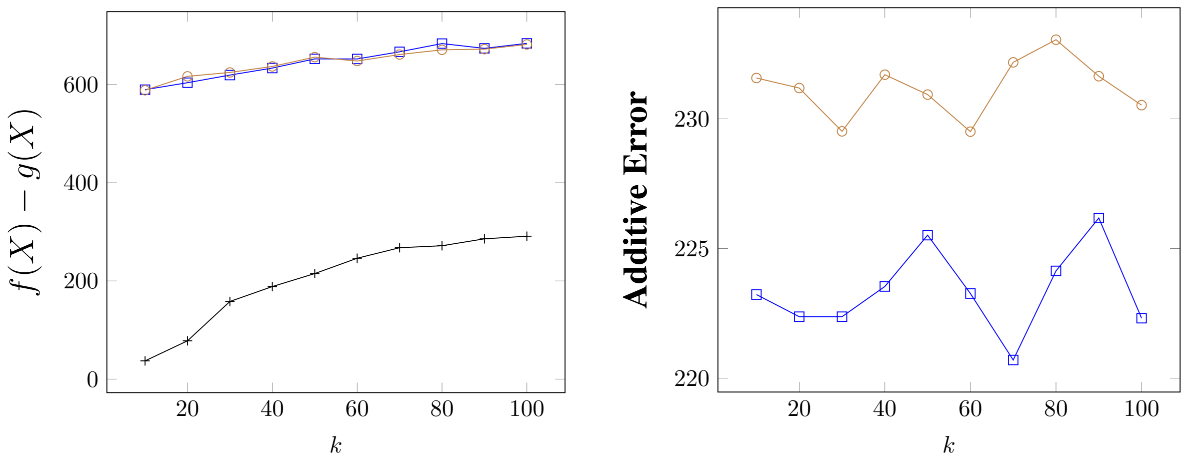


Figure 11: DIFF-C, Bitcoin; a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX and LINEAR-APPROX

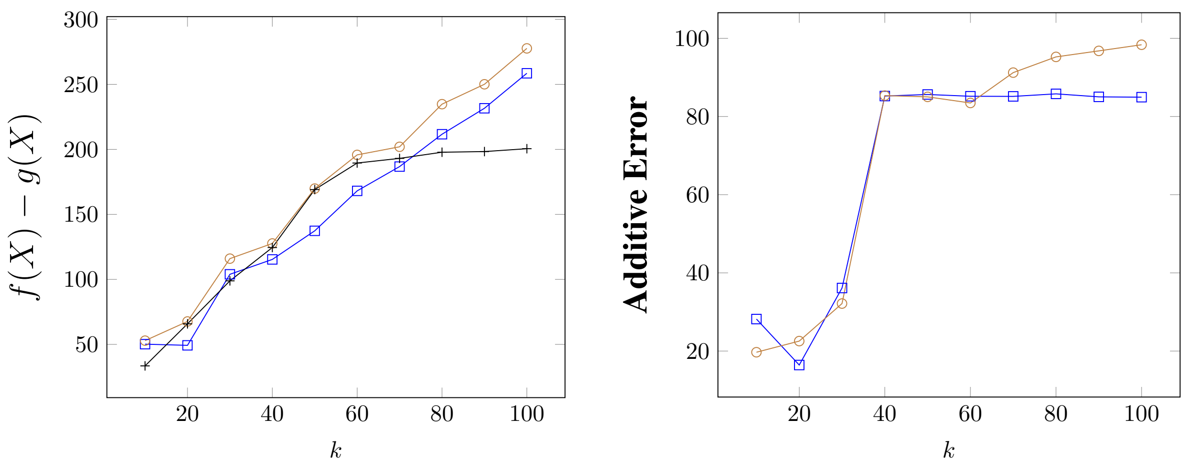


Figure 12: DIFF-C, Wikipedia; a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX and LINEAR-APPROX

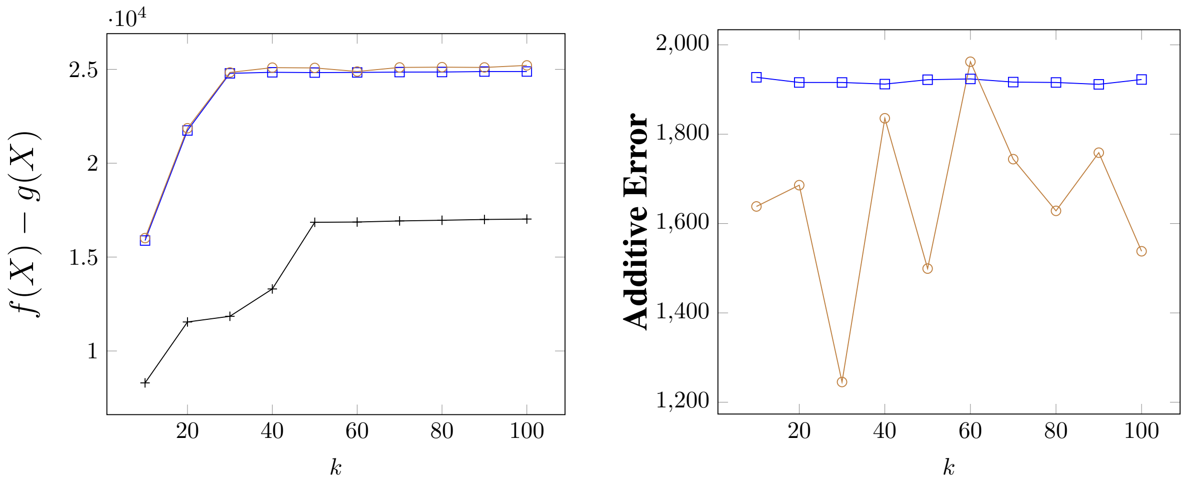


Figure 13: DIFF-C; DBLP a) Budget vs Difference on LOG-APPROX, LINEAR-APPROX and supSub; b) Budget vs Additive error on LOG-APPROX and LINEAR-APPROX

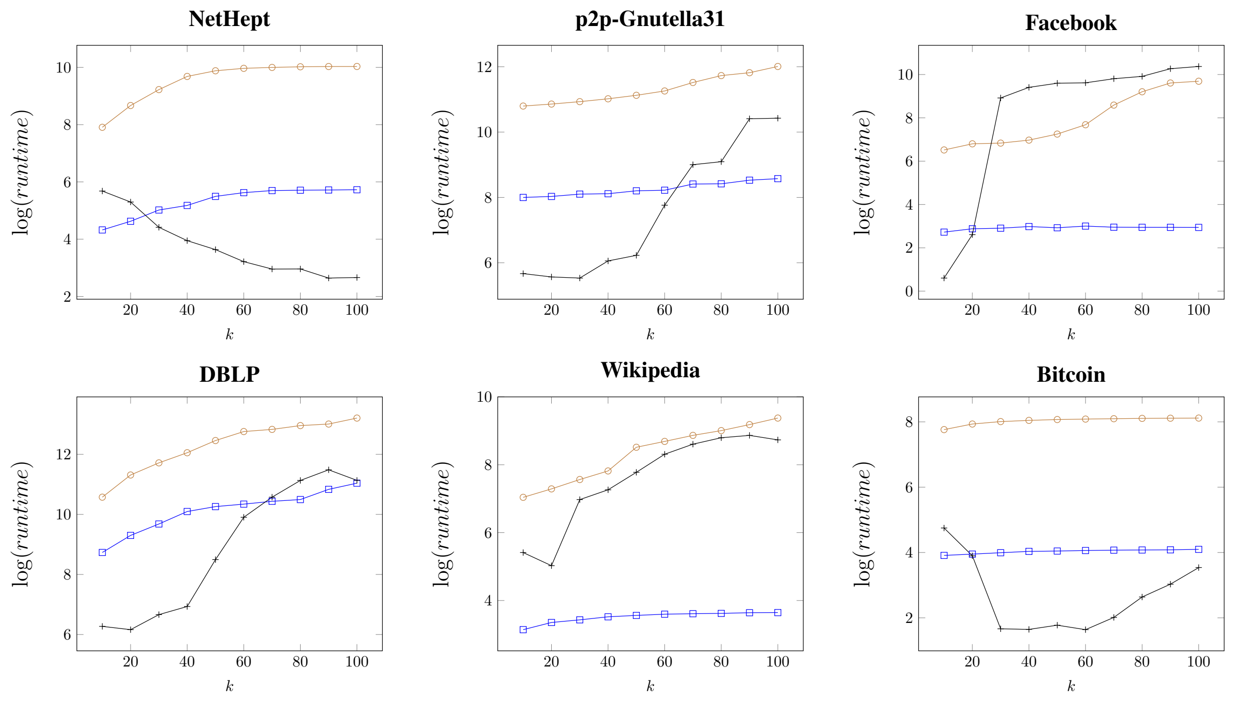


Figure 14: DIFF-C; k vs. Logarithm of Runtime(s) on LOG-APPROX, LINEAR-APPROX and supSub with Submodular Cost

References

Nethept. `https://microsoft.com/en-us/research/people/weic/`, 2009.

Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discret. Appl. Math.*, 7(3):251–274, 1984.

Rishabh Iyer and Jeff Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *International Conference on Neural Information Processing Systems*, pages 2436–2444, USA, 2013. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999792.2999884`.

Rishabh K. Iyer and Jeff A. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *CoRR*, abs/1207.0560, 2012. URL `http://arxiv.org/abs/1207.0560`.

Tianyuan Jin, Yu Yang, Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. Unconstrained submodular maximization with modular costs: Tight approximation and application to profit maximization. *Proceedings of the VLDB Endowment*, 14(10):1756–1768, 2021.

D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221–230. IEEE, 2016.

Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650, 2010.

Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028*, 2002.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, pages 1–8, 2012.