

# Don't Be Greedy: Leveraging Community Structure to Find High Quality Seed Sets for Influence Maximization

Rico Angell<sup>1(✉)</sup> and Grant Schoenebeck<sup>2</sup>

<sup>1</sup> University of Massachusetts, Amherst, USA  
[rangell@cs.umass.edu](mailto:rangell@cs.umass.edu)

<sup>2</sup> University of Michigan, Ann Arbor, USA

**Abstract.** We consider the problem of maximizing the spread of influence in a social network by choosing a fixed number of initial seeds — a central problem in the study of network cascades. The majority of existing work on this problem, formally referred to as the *influence maximization problem*, is designed for submodular cascades. Despite the empirical evidence that many cascades are non-submodular, little work has been done focusing on non-submodular influence maximization.

We propose a new heuristic for solving the influence maximization problem and show via simulations on real-world and synthetic networks that our algorithm outputs more influential seed sets than the state-of-the-art greedy algorithm in many natural cases, with average improvements of 7% for submodular cascades, and 55% for non-submodular cascades. Our heuristic uses a dynamic programming approach on a hierarchical decomposition of the social network to leverage the relation between the spread of cascades and the community structure of social networks. We present “worst-case” theoretical results proving that in certain settings our algorithm outputs seed sets that are a factor of  $\Theta(\sqrt{n})$  more influential than those of the greedy algorithm, where  $n$  is the number of nodes in the network.

## 1 Introduction

A *cascade* is a fundamental social network process in which a number of nodes, or agents, start with some property that they then may spread to neighbors. Network structure has been shown relevant for a wide array of real world cascade processes including the adoption of products [6], farming technology [13], medical practices [12], participation in microfinancing [4], and the spread of information over social networks [26].

---

The full version is located at <https://arxiv.org/abs/1609.06520>.

The authors gratefully acknowledge the support of the National Science Foundation under Career Award 1452915 and AIFT Award 1535912.

How to place a limited number of initial seeds, in order to maximize the spread of the resulting cascade, is a natural question known as INFLUENCE-MAXIMIZATION [16, 22, 23, 32, 35]. This problem requires as input a network, a cascade process, and the number of initial seeds. For example, which students can most effectively be enrolled in an intervention to decrease student conflict at a school [34]?

To study INFLUENCEMAXIMIZATION, we first need to understand how cascades spread. While many cascade models have been proposed [2, 31, 42], they can be roughly divided into two categories: *submodular* and *non-submodular*.

In submodular cascade models, such as the Independent Cascade model defined in Sect. 2 [22, 23, 32], a node’s marginal probability of becoming infected after a new neighbor is infected decreases when the number of previously infected neighbors increases [22]. In non-submodular cascade models the marginal probability of being infected may increase as more neighbors are infected. For example, in the Threshold model [20], each node has a threshold for the number of infected neighbors after which it too will become infected. If a node has a threshold of 2, then the first infected neighbor has zero marginal impact, but the second infected neighbor causes this node to become infected with probability 1. Unlike submodular cascades, non-submodular cascades require well-connected regions to spread [7].

For INFLUENCEMAXIMIZATION in submodular cascades, a straightforward greedy algorithm efficiently finds a seed set with influence at least a  $(1 - 1/e)$  fraction of the optimal; but for general non-submodular cascades, it is NP-hard even to approximate INFLUENCEMAXIMIZATION to within a  $n^{1-\epsilon}$  factor of optimal [22].

Unfortunately, empirical research shows that most cascades are non-submodular [3, 27, 36], and in this case little is known about INFLUENCEMAXIMIZATION other than worst-case hardness. INFLUENCEMAXIMIZATION becomes qualitatively different in the non-submodular setting. In the submodular case, one should put as much distance between the  $k$  initial adopters as possible, lest they erode each other’s effectiveness. However, in the non-submodular case, it may be advantageous to place the initial adopters close together to create synergy and yield more adoptions. Thus, the intuition that it is better to saturate one market first, and then expand implicitly assumes non-submodular influence. However, this synergy renders the problem intractable. Schoenebeck and Tao [37] show that even if the community structure is exactly hierarchical and is given to the algorithm, INFLUENCEMAXIMIZATION remains intractable to approximate in non-submodular settings. This shows that we cannot expect our algorithm to be provably optimal.

However, as we will illustrate, greedy approaches can perform poorly in these settings. Yet, much of the work following Kempe et al. [22], which proposed the greedy algorithm, has attempted to make *greedy approaches* efficient and scalable [8, 9, 11, 30, 39, 41]. New ideas seem necessary to design effective heuristics for non-submodular INFLUENCEMAXIMIZATION.

We observe that structural problems for networks—such as community detection—are also, in general NP-complete, but many efficient heuristics already exist [10, 21]. There are reasons to believe that such problems are not intractable in cases likely to arise in practice [1]. This work asks whether we can design heuristics for INFLUENCEMAXIMIZATION that work well for both submodular and non-submodular cascades, and what new algorithmic techniques might efficiently find hidden synergies necessary to maximize influence.

## 1.1 Contributions

We provide a new heuristic for solving INFLUENCEMAXIMIZATION designed to work for both submodular and non-submodular cascades. Our algorithm takes as input not only a network, but a hierarchical decomposition of the network. It then uses a dynamic programming technique to search for an influential seed set of nodes. We provide the following results concerning our algorithm<sup>1</sup>:

1. We show theoretically that in certain cases, our algorithm outputs seed sets that are a factor of  $\Theta(\sqrt{n})$  more influential than those of the state-of-the-art greedy algorithm, where  $n$  is the number of nodes in the network. This stylized example illustrates the intuition behind our algorithm, as well as the poor performance of greedy.
2. We empirically compare our algorithm to the greedy algorithm via simulations on real-world and synthetic networks for a variety of cascade models. Our algorithm appears to do at least as well as greedy and substantially better for non-submodular cascades. Our algorithm achieves average improvements of 7% for submodular cascades and 55% for non-submodular cascades, performing 266% better in one exceptional case.
3. We verify the importance of network structure by showing that the quality of the hierarchical decomposition impacts the quality of our algorithm’s output.
4. Finally, we define a generalization of our algorithm to a “message-passing” algorithm. While it provably returns seed sets of at least the quality of our dynamic programming algorithm, it typically only offers marginal improvement in the seed set quality while incurring a greater running time. However, we hope that the versatility the message-passing approach can help with future work than focuses on making more scalable versions of our algorithm. Due to space constraints, the message-passing algorithm can be found in the appendix.

## 1.2 Related Work

Following the work of Kempe et al. [22], which proposed the greedy algorithm, extensive work has constructed *efficient and scalable* algorithms and heuristics INFLUENCEMAXIMIZATION [8, 9, 11, 30, 33, 39, 41].

---

<sup>1</sup> See the full version for all of these results at <https://arxiv.org/abs/1609.06520>.

Wang et al. [41] partition the network into communities to inform a greedy-based algorithm in order to increase scalability. The heuristic algorithms presented in [8, 9] rely on input parameters from the user that sacrifice accuracy for speed. The authors state that fine tuning the input parameters can make solving INFLUENCEMAXIMIZATION fast and accurate. Borgs et al. [5] provably show fast running times when the influence function is the independent cascade model. Tang et al. [39] extend this work to provide an algorithm that maintains the same theoretical guarantees as the greedy algorithm presented in [22] and is efficient in practice. Lucier et al. [30] show how to parallelize (in a model based on Map Reduce) the subproblem of determining the influence of a particular seed. Additional work has been done to speed up algorithms for solving INFLUENCEMAXIMIZATION by providing techniques to efficiently compute the total influence of a seed set [11, 24].

Leskovec et al. [28] consider the analogous problem of effectively placing sensors in a network in order to effectively detect an outbreak in the network. They present the algorithm CELF that uses a greedy approach, but leverages the submodularity of the cascade to reduce the amount of time it takes to evaluate the spread of the cascade. Moreover, CELF is built upon by the work in [17, 18], which present modifications to CELF to make an even more cost effective solution to INFLUENCEMAXIMIZATION. Nguyen and Zheng [33] present an algorithm based on belief propagation for INFLUENCEMAXIMIZATION. The algorithm works by systematically removing edges until the resulting graph is a tree, and then running a belief propagation algorithm on the scaled-down network. The article shows that the performance of their algorithm is not substantially worse than that of the greedy algorithm.

In contrast to the aforementioned work, our goal is not to deliver an algorithm that is more efficient and scalable, but rather to present an algorithm that finds higher quality seed sets. Cordasco et al. [14] recently published an algorithm aimed at improving the quality of the seed set discovered, but is limited to deterministic influence patterns where each nodes has a fixed integer threshold. This algorithm greedily maximizes the total influence instead of the total number of infected nodes at the end of the cascade. They show empirically that their algorithm finds higher quality seed sets than the traditional greedy algorithm. On the other hand, our work is focused on the more general and traditional stochastic variant of INFLUENCEMAXIMIZATION. Additionally, with the exception of [33], the prior work is based on a greedy-like approach and suffers from the short-comings of this approach. Our algorithm uses a dynamic programming framework, and is fundamentally different.

Other variations of INFLUENCEMAXIMIZATION have also been considered, e.g. [19, 38].

## 2 Preliminaries

A real function  $f$  on sets is *submodular* if the marginal gain of from adding an element to a set  $A$  is at least as large as the marginal gain from adding the

same element to a superset  $B$  of  $A$ . Formally,  $f$  is submodular if for all  $A, B, u$  where  $A \subseteq B$  we have  $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$ .

*Cascade Model.* A cascade model is a triple  $(G, F, S)$  where  $G = (V, E)$  is an unweighted graph;  $F = \{f_v : \{0, 1\}^{|\Gamma(v)|} \rightarrow [0, 1]\}_{v \in V}$  is a collection of **local influence functions**, where  $\Gamma(v)$  is the set of  $v$ 's neighbors,  $f_v$  takes in the set of infected neighbors of a node  $v$ , and produces a real value which encodes the "influence" of this set on  $v$ ; and  $S$  is the subset of the vertices that are initially infected. The cascade will proceed in rounds. In round 0, the set  $S$  is infected and each of the remaining vertices is assigned a threshold value  $\theta_v \in [0, 1]$  drawn uniformly at random. At each subsequent round, a vertex  $v$  becomes infected if and only if  $f_v(T) \geq \theta_v$ , where  $T$  is the set of  $v$ 's infected neighbors. We will require  $f_v$  to be monotone for each  $v$ .

We denote the **global influence function** as  $\sigma(S)$  which is the *expected* total number of infected vertices due to the influence of the initial seed set  $S$ .

It can be shown that if  $f_v$  is submodular for each  $v$ , then the global influence function  $\sigma$  is submodular too [32]. Thus, we say that a model of cascade is **submodular** if  $f_v$  is submodular for each  $v$ , and is **non-submodular** otherwise.

The same research that shows the  $f$  usually fail to be submodular [3, 27, 36] shows that this submodularity fails in one particular way: the second adopting neighbor is, on average, more influential than the first; and that after this point, each subsequent adopting neighbor's marginal influence decreases. We call such functions **2-quasi-submodular**. Formally,  $f$  is **2-quasi-submodular** if for all  $A, B, A \subseteq B, |A|, |B| \geq 1$  and  $u \notin A, B$ , we have  $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$ ; and for  $v \neq u$ , we have  $f(\{u\}) - f(\emptyset) \leq f(\{u, v\}) - f(\{v\})$ .

Any nonzero submodular influence function  $f_v$  can be turned into a 2-quasi-submodular function by sufficiently decreasing the value of  $f_v(\cdot)$  on singleton sets.

For any local influence function  $f_v$ , we define the  **$q$ -deflated** version  $f_v^{q-defl}$  of  $f_v$  as follows:

$$f_v^{q-defl}(S) = \begin{cases} q \cdot f_v(S) & |S| = 1 \\ f_v(S) & \text{o.w.} \end{cases}$$

*Specific Cascade Models.* The two popular cascade models studied in the INFLUENCEMAXIMIZATION literature are the Independent Cascade model (ICM) and the Linear Threshold model (LTM). In the **Independent Cascade model**, each newly infected node infects each currently uninfected neighbor in the subsequent round with some fixed probability  $p$ . Thus, for all  $v$ ,

$$f_v^{ICM}(S) = 1 - (1 - p)^{|S|}.$$

In the **Linear Threshold model**, each node has a threshold  $\theta_v \in [0, 1]$ , each of  $v$ 's neighbors  $u$  has influence  $b_{u,v}$  on  $v$  such that  $\sum_{u \in \Gamma(v)} b_{u,v} \leq 1$ , and  $v$  becomes infected when the sum of the influences of the infected neighbors meets or surpasses  $v$ 's threshold.

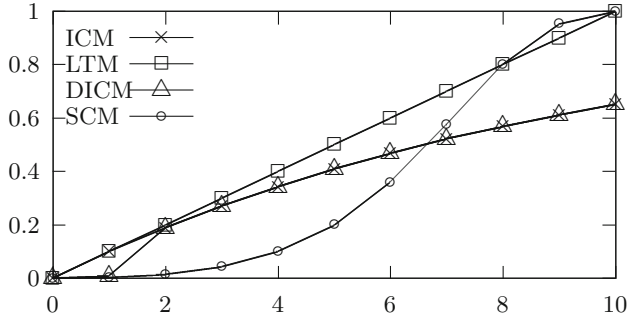
We define the **Deflated Independent Cascade model** (DICM) which takes two parameters:  $p, q \in [0, 1]$  to be the  $q$ -deflated version of the Independent Cascade model.

In the  **$S$ -Cascade model** (SCM) we have that

$$f_d^{SCM}(S) = \frac{\left(\frac{|S|}{2d}\right)^2}{\left(\frac{|S|}{2d}\right)^2 + \left(1 - \frac{|S|}{d}\right)^2}.$$

The fraction  $|S|/d$  is the fraction of infected neighbors of a given node. This is a modified version of the Tullock Cost function [40] with power 2.

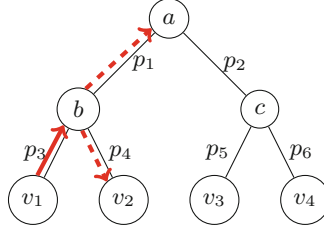
We note that the Independent Cascade model and the Linear Threshold model are submodular, while the  $q$ -Deflated Independent Cascade model (for  $q < 1 - p/2$ ) and  $S$ -Cascade are not. Figure 1 illustrates the local influence functions of the various cascade models.



**Fig. 1.** The local influence functions where the parameters of the ICM and DICM and the influence weights of LTM are all .1; and the vertex’s degree in LTM and SCM is 10.

**Synthetic Network Model.** Most existing synthetic models fail to have meaningful asymmetry between nodes, or significant community structure, or both. Therefore, we design our own synthetic network model. The **directed**  $(d, \ell, t)$ -**hierarchical network model** creates a random network on  $2^d$  nodes as follows: We create an edge-weighted complete binary tree of depth  $d$ , each leaf representing a vertex of the graph. The weights are drawn i.i.d from a Binomial( $\ell, 1/2$ ) distribution. Each node  $v$  issues  $t$  random edges, each generated via a random walk — illustrated in Fig. 2. Each random walk starts at  $v$ . At each step in the walk, an outgoing edge is chosen proportional to its weight (we disallow exiting the node along the same edge that the walk arrived at the node). The walk terminates when it arrives at a leaf node. If a terminating node is duplicated, we draw again, which keeps the graph simple.

As  $\ell$  grows larger, this approaches the hierarchical Kleinberg model [25]. But for moderately sized  $\ell$ , there is a non-trivial amount of asymmetry introduced into the graph — some subcommunities are more influential than others.



**Fig. 2.** Illustrative example of a random walk to generate an edge in our synthetic network model.

*Hierarchical Decomposition.* We define a **hierarchical decomposition** of a graph  $G$  to be a rooted full binary tree  $T = (V_T, E_T)$  where the leaves of  $T$  correspond to the vertices of  $G$ . For a tree node  $v \in V_T$ , define  $\mathbf{T}(v)$  to be the subset of vertices in  $G$  corresponding to the leaves of the subtree rooted at  $v$ . Let the **height** of  $v \in V_T$  be defined as the length of the path to  $v$ 's deepest descendent.

We use the recently proposed cost function of Dasgupta [15] to evaluate the quality of a hierarchical decomposition. Let  $\text{lca}_T(u, v)$  be the least common ancestor of  $u, v \in V$  in the tree  $T$ . Then we define

$$\text{Cost}(T) = \sum_{\{u,v\} \in E} |T(\text{lca}(u, v))|$$

which sums the number of leaves in the smallest subtree containing each edge.

*Influence Maximization.* An **InfluenceMaximization Instance** consists of a graph  $G = (V, E)$ , an influence function  $\sigma$ , and an integer  $k$ . Given an InfluenceMaximization Instance, the goal is to find a set  $S$  of  $k$  nodes as to maximize  $\sigma(S)$ .

The **greedy algorithm** [22] for an INFLUENCEMAXIMIZATION Instance start with a tentative seed set  $T = \emptyset$  and for  $k$  rounds, simply adds  $\arg \max_{v \in V} \sigma(T \cup \{v\})$  to  $T$ .

### 3 DPIM: Dynamic Programming Influence Maximization Algorithm

The Dynamic Programming Influence Maximization Algorithm (*DPIM*), formally specified in Algorithm 1, takes as input a graph  $G$ , and corresponding hierarchical decomposition  $T$ , an integer  $k$ , and a global influence function  $\sigma(\cdot)$  and outputs a subset of vertices  $S \subseteq V$  such that  $|S| = k$  and  $S$  is a highly influential set of seeds. *DPIM* seeks to maximize the total influence of a fixed-sized seed set  $S$  by performing dynamic programming upon  $T$ .

For each node  $v \in T$ , and each  $i \in \{0, 1, \dots, \min(|T(v)|, k)\}$ , the algorithm stores  $A[v, i]$ , a choice of  $i$  seeds in  $T(v)$  which seeks to maximize the total influence in  $G$ . Starting at the leaves of the tree, and moving up level by level until

reaching the root, DPIM processes each tree node. For each leaf node  $v \in T$ , we store  $A[v, 0] = \emptyset$  and  $A[v, 1] = \{v\}$ . For each internal node  $v \in T$ , which has children  $v_L$  and  $v_R$ , we set  $A[v, i] = A[v_L, j] \cup A[v_R, i - j]$  where  $j \in \{0, 1, \dots, i\}$  is selected as to maximize  $\sigma(A[v_L, j] \cup A[v_R, i - j])$ . Thus the algorithm optimally splits the seeds between a tree nodes two subtrees given how the algorithm has already determined that each subtree should allocate any particular number seeds.

---

**Algorithm 1.** *DPIM: Dynamic Programming Influence Maximization*  
Algorithm

---

**Input:**  $G = (V, E), T = (V_T, E_T), \sigma(\cdot), k$

**Output:**  $S \subset V$  such that  $|S| = k$

Let  $A[\cdot, \cdot] = V_T \times [k] \rightarrow 2^V$ , such that  $A[v_T, j]$  stores a choice of  $j$  seeds in  $T(v_T)$ .

Let  $r \in V_T$  be the root of  $T$  and  $h$  be its height.

**for** each height  $i = 0, 1, \dots, h$  **do**

**for** each node  $v \in V_T$  with height  $i$  **do**

**if**  $i = 0$  **then**

$A[v, 0] = \emptyset$

$A[v, 1] = \{v\}$

**else**

            Let  $v_L, v_R$  be the left and right children of  $v$ , respectively.

**for** each  $i = 0, 1, \dots, \min\{|T(v)|, k\}$  **do**

$j = \arg \max_{j \in \{0, 1, \dots, i\}} \sigma(A[v_L, j] \cup A[v_R, i - j])$

$A[v, i] = A[v_L, j] \cup A[v_R, i - j]$

**end**

**end**

**end**

**end**

**return**  $A[r, k]$

---

The analysis of the running time for *DPIM* is straightforward.

**Theorem 1.** *Given a graph  $G = (V, E)$  with  $|V| = n, |E| = m$ , fixed positive integers  $k, r$ , and a hierarchical decomposition  $T$ , *DPIM* calls the  $\sigma(\cdot)$  oracle  $O(nk^2)$  times.*

*Proof.* Observe that, for each node in  $T$ , *DPIM* makes  $O(k^2)$  queries to  $\sigma(\cdot)$ . The number of nodes in  $T$  is exactly  $2n - 1$ .

Hence, the number of oracle calls in *DPIM* is  $O(nk^2)$ .

Note that this is a factor of  $k$  more than the greedy algorithm, which requires only  $O(nk)$  calls to the oracle. The execution of a single query to  $\sigma(\cdot)$  can be approximated by repeatedly,  $r$  times, simulating the cascade process and returning the average number of infected vertices. This can be done in time  $O(mr)$  because simulating the cascade requires at most simulating the cascade



on each edge in  $G$ . However, there are often techniques to speed up the oracle access beyond simply running the cascade [5], but they are beyond the scope of this work.

## 4 Experimental Results About DPIM

In this section, we discuss empirical results of *DPIM*. See the full version for theoretical results and further empirical analysis.

### 4.1 Experimental Setup

We execute *DPIM* and the greedy algorithm from [22] on a variety of networks and cascades to test the relative quality of solutions.

**Cascade Models.** We adopt the two common submodular cascade models from the literature: the linear threshold model and the independent cascade model, defined in Sect. 2, and two non-submodular cascades:

- (I) *Independent Cascade*(IC): We uniformly assign the probability  $p = 1\%$ , thus  $v$  with  $\ell$  infected neighbors is infected with probability  $1 - (0.99)^\ell$ .
- (II) *Linear Threshold* (LT): For each node  $v$ , we assign each of  $u \in \Gamma(v)$  to have  $1/|\Gamma(v)|$  influence on  $v$ .
- (III) *Deflated Independent Cascade* (DIC): We uniformly assign the probability  $p = 1\%$ , thus  $v$  with  $\ell = 1$  infected neighbors is infected with probability  $0.1\%$  ( $q = 0.1$ ) and with  $\ell \geq 2$  infected neighbors is infected with probability  $1 - (0.99)^\ell$ .
- (IV) *S-Cascade model* (SCM): The influence on any given node  $v$  is

$$\frac{(x/2)^2}{(x/2)^2 + (1-x)^2},$$

where  $x$  is the fraction of  $v$ 's neighbors that are infected.

Both of these algorithms require access to an oracle for  $\sigma(\cdot)$ , which is also required to evaluate the effectiveness of the algorithms. To implement this oracle, we simulate the cascade 100 times, resampling the randomness for the cascade each time (using pseudorandomness from the standard C++ library) and return the average number of infections.

**Networks.** We use two real-world networks (from [29]) and two synthetic networks, summarized in Table 1. In the arXiv collaboration network (ca-GrQc), the vertices are authors of e-print scientific articles and edges represent coauthorship relations. The ego-Facebook network is largest such network provided by [29]. This network denotes the facebook friendship ties from a single person's (ego's) set of friends. The ego vertex has been removed. Furthermore, we generate two synthetic networks by first sampling from directed  $(d, \ell, t)$ -hierarchical network model using parameters  $(10, 50, 50)$  and  $(11, 50, 50)$  (synthetic-1 & synthetic-2, resp.), and then making the graph simple and undirected in the natural way.

**Table 1.** Networks used to evaluate the effectiveness of our algorithm.

Name	Nodes	Edges
synthetic-1	1,024	51,200
synthetic-2	2,048	102,400
ca-GrQc	5,276	28,827
ego-Facebook	1,034	53,498

**Algorithms for Hierarchical Decomposition.** Lastly, in order to evaluate our algorithm, we present 4 algorithms for generating a hierarchical decomposition of any network. The algorithms we used in our simulations are implemented as follows:

- (I) Random Pair: Each node starts in its own partition, and partitions are joined randomly until all of the nodes are contained in one partition.
- (II) Random Edge: Each node starts in its own partition, and partitions are joined by contracting a random edge between partitions. If no edges remain between the partitions, partitions are merged randomly until all of the nodes are contained in one partition.
- (III) Jaccard Similarity: Each node starts in its own partition, and pairs of partitions  $(A, B)$ , for  $A, B \subset V$  are joined based on which pair maximizes

$$\frac{|\Gamma(A) \cap \Gamma(B)|}{|\Gamma(A) \cup \Gamma(B)|},$$

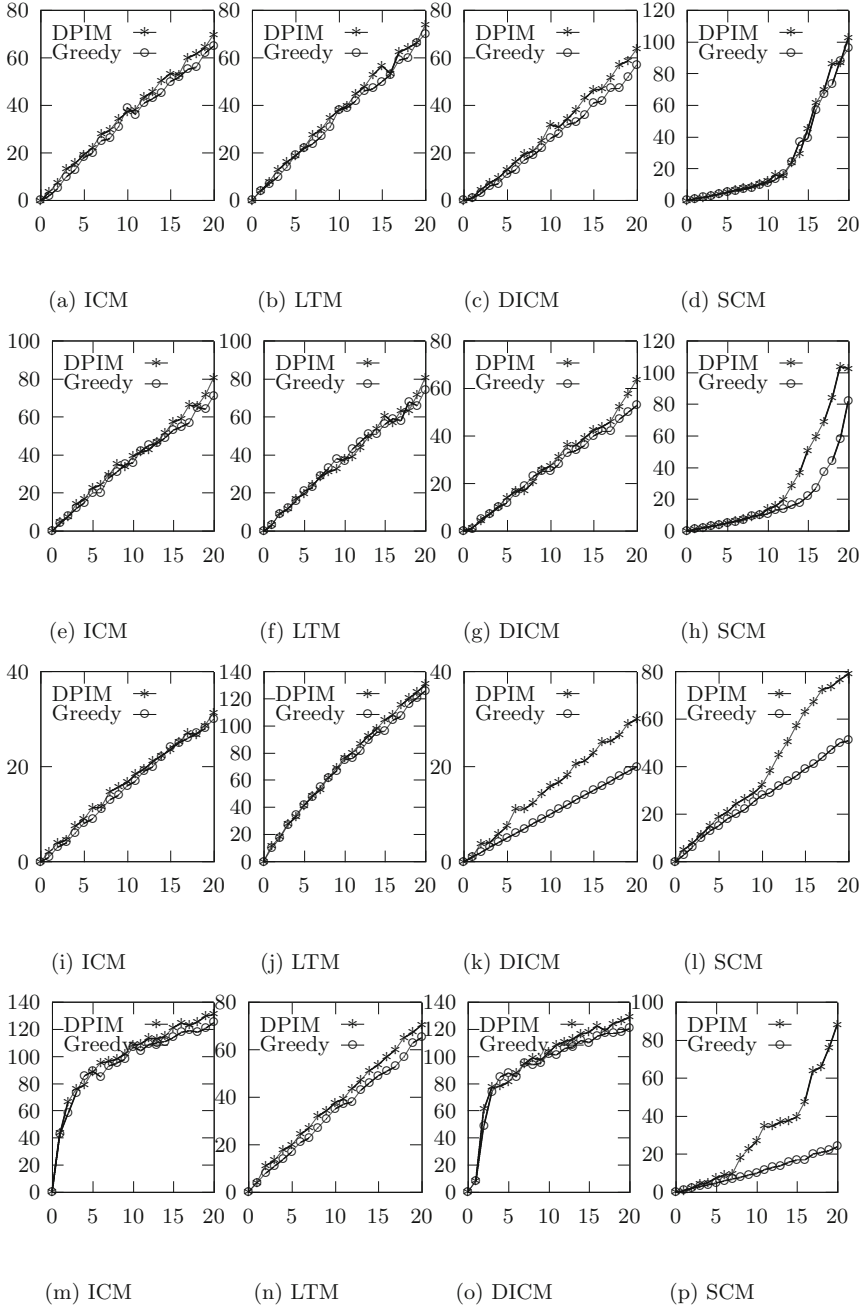
where  $\Gamma(X \subset V) = \bigcup_{v \in X} \Gamma(v)$ .

- (IV) METIS-based: The whole network starts as one partition; using METIS [21], partitions are recursively divided into two partitions until each partition contains only a single node.

## 4.2 Algorithm Evaluation

**Performance of DPIM.** The results of the simulations we ran are shown in Fig. 3. For each execution of *DPIM*, we used the METIS-based hierarchical decomposition algorithm to construct a hierarchical decomposition of the network.

Considering cascades across all four networks with seed set size 20, *DPIM* increases influence on average by 8% for ICM, 6% for LTM, 22% for DICM, and 88% for SCM. Surprisingly, *DPIM* performs marginally better than the greedy algorithm even for submodular cascades. As predicted, when the cascade is non-submodular, *DPIM* outperforms the greedy algorithm by a significant amount. However, gains were more impressive for synthetic-2, ca-GrQc, and ego-Facebook — including an 266% increase in influence for the SCM cascade on the ego-Facebook network — than for synthetic-1, where we see only marginal improvement even when the cascade is non-submodular. Table 2 contains,



**Fig. 3.** Comparison of performance: *DPIM* vs. Greedy. The rows from top to bottom correspond to synthetic-1, synthetic-2, ca-GrQc, and ego-Facebook, respectively. For each plot, the x-axis is  $k$  and the y-axis is the number of total infections at the end of the cascade.

for  $k = 20$ , the approximated expected total influence values for each simulation rounded to the nearest integer. Figure 3 includes charts of the improvement for all  $k$ .

In the full version of the paper, we present further empirical results and generalize *DPIM* to a message passing algorithm.

**Table 2.** Expected total influence of the final seed sets of size 20 chosen by both algorithms for each of the simulations (rounded to the nearest integer).

	synthetic-1		synthetic-2		ca-GrQc		ego-Facebook	
	Greedy	<i>DPIM</i>	Greedy	<i>DPIM</i>	Greedy	<i>DPIM</i>	Greedy	<i>DPIM</i>
ICM	65	70	71	80	30	31	125	131
LTM	70	74	74	81	126	130	65	70
DICM	57	64	53	64	20	30	121	129
SCM	96	102	82	103	51	79	24	88

## 5 Future Work

*DPIM* is typically not as fast as naive greedy, and to be useful in practice, it would greatly help if it were more scalable. We believe that this will prove to be the case. For example, we could stop the recursion before exploring the entire hierarchical decomposition. We might stop dividing if a subtree does not appear to have any additional community structure, and then run a heuristic (such as degree or greedy) to process the rest of the subtree. The intuition here is that dynamic programming works best where the network has strong community structure, so where no structure exists, it may not provide much added benefit. Additionally, the same techniques that have made the greedy algorithm more scalable might be adopted to our dynamic programming framework. Finally, we think that the versatility of when to schedule updates in the “message-passing” algorithm (which generalizes *DPIM*) may be useful in scaling our approach.

Another possible direction of future exploration is to try additional hierarchical decomposition techniques. Interestingly, *DPIM* can be seen as a way to *test* hierarchical decomposition techniques. Decompositions that perform better are intuitively finding a better decomposition.

## References

1. Arora, S., Ge, R., Sachdeva, S., Schoenebeck, G.: Finding overlapping communities in social networks: toward a rigorous approach. In: ACM EC 2012 (2012)
2. Arthur, W.B.: Competing technologies, increasing returns, and lock-in by historical events. *Econ. J.* **99**(394), 116–131 (1989). <http://www.jstor.org/stable/2234208>
3. Backstrom, L., Huttenlocher, D.P., Kleinberg, J.M., Lan, X.: Group formation in large social networks: membership, growth, and evolution. In: KDD 2006, pp. 44–54 (2006)
4. Banerjee, A., Chandrasekhar, A.G., Duflo, E., Jackson, M.O.: The diffusion of microfinance. *Science*, 341(6144) (2013)

5. Borgs, C., Brautbar, M., Chayes, J.T., Lucier, B.: Maximizing social influence in nearly optimal time. In: SODA 2014 (2014)
6. Brown, J.J., Reingen, P.H.: Social ties and word-of-mouth referral behavior. *J. Consum. Res.* **14**, 350–362 (1987)
7. Centola, D.: The spread of behavior in an online social network experiment. *Science* **329**(5996), 1194–1197 (2010)
8. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: KDD 2009. ACM (2009)
9. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: ICDM 2010, pp. 88–97. IEEE (2010)
10. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066111 (2004)
11. Cohen, E., Delling, D., Pajor, T., Werneck, R.F.: Sketch-based influence maximization and computation: scaling up with guarantees. In: CIKM 2014, pp. 629–638. ACM (2014)
12. Coleman, J., Katz, E., Menzel, H.: The diffusion of an innovation among physicians. *Sociometry* **20**, 253–270 (1957)
13. Conley, T.G., Udry, C.R.: Learning about a new technology: pineapple in Ghana. *Am. Econ. Rev.* **100**(1), 35–69 (2010)
14. Cordasco, G., Gargano, L., Mecchia, M., Rescigno, A.A., Vaccaro, U.: Discovering small target sets in social networks: a fast and effective algorithm. arXiv preprint [arXiv:1610.03721](https://arxiv.org/abs/1610.03721) (2016)
15. Dasgupta, S.: A cost function for similarity-based hierarchical clustering. In: STOC 2016, pp. 118–127. ACM, New York, NY, USA (2016). [http://doi.acm.org/10.1145/2897518.2897527](https://doi.acm.org/10.1145/2897518.2897527)
16. Domingos, P., Richardson, M.: Mining the network value of customers. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 57–66 (2001)
17. Goyal, A., Lu, W., Lakshmanan, L.V.: Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: WWW 2011. pp. 47–48. ACM (2011)
18. Goyal, A., Lu, W., Lakshmanan, L.V.: Simpath: an efficient algorithm for influence maximization under the linear threshold model. In: ICDM 2011, pp. 211–220. IEEE (2011)
19. Goyal, S., Kearns, M.: Competitive contagion in networks. In: STOC 2012, pp. 759–774 (2012)
20. Granovetter, M.: Threshold models of collective behavior. *Am. J. Sociol.* **83**(6), 1420–1443 (1978). <http://www.journals.uchicago.edu/doi/abs/10.1086/226707>
21. Karypis, G., Kumar, V.: METIS: unstructured graph partitioning and sparse matrix ordering system, Version 4.0. (2009). <http://www.cs.umn.edu/~metis>
22. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD 2003, pp. 137–146 (2003)
23. Kempe, D., Kleinberg, J., Tardos, É.: Influential nodes in a diffusion model for social networks. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1127–1138. Springer, Heidelberg (2005). [https://doi.org/10.1007/11523468\\_91](https://doi.org/10.1007/11523468_91)
24. Kimura, M., Saito, K.: Tractable models for information diffusion in social networks. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS, vol. 4213, pp. 259–271. Springer, Heidelberg (2006). [https://doi.org/10.1007/11871637\\_27](https://doi.org/10.1007/11871637_27)

25. Kleinberg, J.: Small-world phenomena and the dynamics of information. In: NIPS 2002, vol. 1, pp. 431–438 (2002)
26. Lerman, K., Ghosh, R.: Information contagion: an empirical study of the spread of news on Digg and Twitter social networks. ICWSM **10**, 90–97 (2010)
27. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. In: ACM EC 2006, pp. 228–237 (2006)
28. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: KDD 2007, pp. 420–429. ACM (2007)
29. Leskovec, J., Krevl, A.: SNAP datasets: stanford large network dataset collection, June 2014. <http://snap.stanford.edu/data>
30. Lucier, B., Oren, J., Singer, Y.: Influence at scale: distributed computation of complex contagion in networks. In: KDD 2015, pp. 735–744. ACM (2015)
31. Morris, S.: Contagion. *Rev. Econ. Stud.* **67**(1), 57–78 (2000). <http://restud.oxfordjournals.org/content/67/1/57.abstract>
32. Mossel, E., Roch, S.: Submodularity of influence in social networks: from local to global. *SIAM J. Comput.* **39**(6), 2176–2188 (2010)
33. Nguyen, H., Zheng, R.: Influence spread in large-scale social networks – a belief propagation approach. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012. LNCS, vol. 7524, pp. 515–530. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33486-3\\_33](https://doi.org/10.1007/978-3-642-33486-3_33)
34. Paluck, E.L., Shepherd, H., Aronow, P.M.: Changing climates of conflict: a social network experiment in 56 schools. *Proc. Nat. Acad. Sci.* **113**(3), 566–571 (2016). <http://www.pnas.org/content/113/3/566.abstract>
35. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: KDD 2012, pp. 61–70 (2002)
36. Romero, D.M., Meeder, B., Kleinberg, J.: Differences in the mechanics of information diffusion across topics : idioms, political hashtags, and complex contagion on twitter. In: WWW 2011. pp. 695–704. ACM (2011). <http://dl.acm.org/citation.cfm?id=1963503>
37. Schoenebeck, G., Tao, B.: Beyond worst-case (in)approximability of nonsubmodular influence maximization (2017). <http://www-personal.umich.edu/bstao/>
38. Seeman, L., Singer, Y.: Adaptive seeding in social networks. In: FOCS 2013, pp. 459–468. IEEE (2013)
39. Tang, Y., Xiao, X., Shi, Y.: Influence maximization: near-optimal time complexity meets practical efficiency. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 75–86. ACM (2014)
40. Tullock, G.: Towards a theory of the rent-seeking society. In: chap. Efficient Rent Seeking. Texas A&M University Press (1980)
41. Wang, Y., Cong, G., Song, G., Xie, K.: Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1039–1048. ACM (2010)
42. Watts, D.J.: A simple model of global cascades on random networks. *Proc. Nat. Acad. Sci.* **99**(9), 5766–5771 (2002). <http://www.pnas.org/content/99/5766.abstract>

Web and Internet Economics

13th International Conference, WINE 2017, Bangalore,  
India, December 17–20, 2017, Proceedings

Devanur, N.; Lu, P. (Eds.)

2017, XI, 408 p. 62 illus., Softcover

ISBN: 978-3-319-71923-8