6.883 Learning with Combinatorial Structure
Note for Lecture 18
Author: Ludwig Schmidt

1 Maximizing influence and diffusion

We started the lecture with a brief summary of the diffusion models discussed last time.[1] We then introduced a general diffusion model defined as follows.

1.1 A general diffusion model

Let $S_t \subseteq V$ be the set of nodes active at time t, starting with some initial set of active notes S_0. Moreover, each node $v \in V$ has a uniformly random activation threshold θ_v and an activation function $f_v : 2^V \to \mathbb{R}$ defined on all subsets of the nodes. Given the set of active nodes S_{t-1}, we update the set of active nodes as follows:

1. Set $S_t \leftarrow S_{t-1}$.
2. For all $v \in V \setminus S_{t-1}$, we check whether $f_v(S_{t-1}) \geq \theta_v$. If this is the case, the node v becomes active and we set $S_t \leftarrow S_t \cup \{v\}$.

This diffusion model generalizes both the linear threshold model and the independent cascade model discussed in the previous lecture [KKT03]. Morever, a theorem of Mossel and Roch shows that the expected "influence" induced by the initial set S_0 is monotone submodular as long as the activations and influence are monotone submodular. Here, the influence of a set S_0 is defined via a monotone submodular set function $w : V \to \mathbb{R}$. In particular, we are interested in the expected influence after n steps of the above diffusion process, which defines the following quantity:

$$\sigma_w(S_0) \;=\; \mathbb{E}[w(S_n)] \;.$$

Formally, we have the following result.

Theorem 1 ([MR10])**.** *If all activation functions f_v and the weight function w are monotone submodular, then $\sigma_w(S_0)$ is monotone and submodular.*

1.2 Further questions

Next, we discussed further questions related to diffusion processes.

[1] See notes for Lecture 17.

Learning. A natural question is how to learn a diffusion function. One appraoch is to first learn the underlying diffusion model, which is related to graph learning and sparse estimation problems. Given the learnt diffusion model, we can then evaluate the diffusion function for new sets. Another recent approach [DLBS14] bypasses the model learning stage and directly learns the resulting diffusion function, which is sufficient in cases where we are only interested in estimating the influence of certain sets but not the entire graph structure. The approach of [DLBS14] is based on the fact that many influence functions are coverage functions, which can be approximated by a sum of simpler binary reachability functions.

Evaluation of influence functions. Even if the diffusion model is given, the influence function $\sigma_w(S_0)$ is still hard to compute in many cases because σ_w is defined as an expectation. Researchers have proposed several different approaches for efficiently evaluating influence functions.

- Approximation via hypergraphs [BBCL14, TXS14, TSX15].
- Sketching [CDPW14].
- Approximating Riemann sum [LOS15].
- Sparsification [MBC$^+$11].
- Heuristics, e.g., degree discount [CWY09].

2 Making greedy faster

Yet another approach for evaluating influence functions more efficiently is making the greedy algorithm itself faster. Besides influence functions, a faster greedy algorithm is useful in many other submodular maximization problems over large domains. While the "rectangular" running time of $O(nk)$ for the regular greedy algorithm is sufficient for small- and medium-scale problems, larger instances with large values of n and k can become computationally expensive. Therefore, several ways of speeding up the greedy algorithm have been proposed.

- The lazy greedy algorithm – we will explore this algorithm in the next problem set [Min78].
- Instead of picking only the single best element in each iteration of the greedy algorithm, we can also pick all elements above a certain treshold [BV14]. This leads to a time complexity of $O(\frac{n}{\varepsilon} \log \frac{n}{\varepsilon})$.

- We can achieve an even better running time of $O(n \log \frac{1}{\varepsilon})$ using randomization: instead of considering all elements, we pick the best element from a small sample [MBK+15].
- Gain maximization as optimization [PJB14].
- Parallel and distributed algorithms for monotone function maximization [KMVV13, MKSK13, WIB14, PJG+14].
- Algorithms working in an online or streaming setting [BMKK14].

2.1 Stochastic greedy

The stochastic greedy algorithm addresses the usual problem of maximizing a monotone submodular function $F : 2^V \to \mathbb{R}$ subject to a cardinality constraint:

$$\operatorname*{arg\,max}_{S \subseteq V, |S| \leq k} F(S) \ .$$

The stochastic greedy algorithm repeats the following steps for $i = 1, \ldots, k$, starting with an initially empty set of selected elements $S_0 = \{\}$.

1. Randomly pick a set $T_i \subseteq V \setminus S_{i-1}$ of size $|T_i| = \frac{n}{k} \log \frac{1}{\varepsilon}$.
2. Find the best element a_i in T:

$$a_i \ = \ \operatorname*{arg\,max}_{a \in T_i} F(a \,|\, S_{i-1}) \ = \ \operatorname*{arg\,max}_{a \in T_i} F(\{a\} \cup S_{i-1}) \ .$$

3. Add the element a_i to the set of selected elements:

$$S_i \leftarrow S_{i-1} \cup \{a_i\} \ .$$

Compared to the standard greedy algorithm, the stochastic greedy algorithm only searches over a smaller set of candidate elements T in each iteration. Hence the time complexity of the stochastic greedy algorithm is clearly better than that of standard greedy. The total number of oracle calls $O(n \log \frac{1}{\varepsilon})$ stated above directly follows from the cardinality of the sets T_i.

In addition to the good running time, the stochastic greedy algorithm still achieves a good expected approximation ratio. In particular, we have

$$\mathbb{E}[F(S_k)] \ \geq \ \left(1 - \frac{1}{e} - \varepsilon\right) \cdot \text{OPT} \ = \ \left(1 - \frac{1}{e} - \varepsilon\right) \max_{S \subseteq V, |S| \leq k} F(S) \ .$$

The expectation is over the randomness of the algorithm. Note that the approximation ratio is only worse by an additive ε compared to the guarantee for the standard greedy algorithm. Moreover, the stochastic greedy algorithm has good empirical performance (see plot in the slides).

We now proof the main lemma for the stochastic greedy algorithm. Similar to the standard greedy algorithm, we show that every iteration of the algorithm makes at least a certain amount of progress (now in expectation).

Lemma 1 ([MBK+15])**.** *The expected gain of one iteration of the stochastic greedy algorithm is at least*

$$\mathbb{E}[F(S_i) - F(S_{i-1})] \;\geq\; \frac{1-\varepsilon}{k} \cdot \mathbb{E}\left[(F(S^*) - F(S_{i-1}))\right]$$

where S^* is an optimal set, i.e., we have $F(S^*) = \mathrm{OPT}$.

Proof. Let T_i be the random set picked by the stochastic greedy algorithm. Using $1 + x \leq \exp(x)$ and $|V \setminus S_{i-1}| \leq n$, we then get Then we have

$$\begin{aligned}
\mathbb{P}[T_i \cap (S^* \setminus S_{i-1}) = \varnothing] \;&=\; \left(1 - \frac{|S^* \setminus S_{i-1}|}{|V \setminus S_{i-1}|}\right)^m \\
&\leq \exp\left(-m\frac{|S^* \setminus S_{i-1}|}{|V \setminus S_{i-1}|}\right) \\
&\leq \exp\left(-\frac{m}{n}|S^* \setminus S_{i-1}|\right) \;.
\end{aligned}$$

Now, note that $1 - \exp(-\frac{mk}{n}x)$ is a concave function of $x \in [0,1]$. Combined with $0 \leq \frac{|S^* \setminus S|}{k} \leq 1$, we use this get the following lower bound:

$$\begin{aligned}
\mathbb{P}[T_i \cap (S^* \setminus S_{i-1}) \neq \varnothing] \;&\geq\; 1 - \exp\left(-\frac{m}{n}|S^* \setminus S_{i-1}|\right) \\
&\geq \left(1 - \exp\left(-\frac{mk}{n}\right)\right)\frac{|S^* \setminus S_{i-1}|}{k} \\
&\geq (1-\varepsilon)\frac{|S^* \setminus S_{i-1}|}{k} \;.
\end{aligned}$$

Next, we bound the marginal increase of the best element in T_i. The stochastic greedy algorithm picks the best element $a_i \in T_i$, and hence the marginal gain of element a_i is at least as good as a random element in T_i. Since we pick the set T_i uniformly at random, every element of $S^* \setminus S_{i-1}$ has the same probability of being in T_i. So conditioned on the event that $T_i \cap (S^* \setminus S_{i-1}) \neq \varnothing$, the expected marginal gain of element a_i is at least as good

as that of a uniformly random element from $S^* \setminus S_{i-1}$. Since the function F is monotone, all marginal gains are non-negative and we get

$$\begin{aligned}
\mathbb{E}[F(a_i \,|\, S_{i-1}) \,|\, S_{i-1}] \;&\geq\; \mathbb{P}[T_i \cap (S^* \setminus S_{i-1}) \neq \varnothing] \cdot \mathbb{E}[F(a_i \,|\, S_{i-1}) \,|\, T_i \cap (S^* \setminus S_{i-1}) \neq \varnothing,\, S_{i-1}] \\
&\geq\; (1-\varepsilon)\frac{|S^* \setminus S_{i-1}|}{k} \;\cdot\; \frac{1}{|S^* \setminus S_{i-1}|} \sum_{e \in S^* \setminus S_{i-1}} F(e \,|\, S_{i-1}) \\
&=\; \frac{1-\varepsilon}{k} \sum_{e \in S^* \setminus S_{i-1}} F(e \,|\, S_{i-1}) \;.
\end{aligned}$$

From monotonicity and submodularity of F, we have

$$\begin{aligned}
F(S^*) - F(S_{i-1}) \;&\leq\; F(S^* \cup S_{i-1}) - F(S_{i-1}) \\
&=\; F((S^* \setminus S_{i-1}) \cup S_{i-1}) - F(S_{i-1}) \\
&=\; F(S^* \setminus S_{i-1} \,|\, S_{i-1}) \\
&\leq \sum_{e \in S^* \setminus S_{i-1}} F(e \,|\, S_{i-1}) \;.
\end{aligned}$$

Combining the above two inequalities yields

$$\mathbb{E}[F(a_i \,|\, S_{i-1}) \,|\, S_{i-1}] \;\geq\; \frac{1-\varepsilon}{k} \left(F(S^*) - F(S_{i-1})\right) \;.$$

Applying the definition of S_i and the tower property of expectation gives the statement of the lemma. □

The overall approximation guarantee of the stochastic greedy algorithm now follows from a similar argument as for the standard greedy algorithm.

2.2 Distributed greedy

Another important direction for speeding up the greedy algorithm is by designing a distributed variant. In the distributed setting, we have a fixed number of machines, each of which contains a subset of the ground set V. A natural approach is to run the standard greedy algorithm independently on each machine in order to determine a local solution of the desired size with a provable approximation guarantee. We can then send the local solutions to a central machine, where we run the greedy algorithm again to find an overall solution with cardinality k.

2.2.1 A counterexample

While this approach is intuitive, it can lead to an approximation ratio that is arbitrarily close to 0 as a function of the number of machines. The main problem is that the local greedy decisions are only good with respect to the local set of elements (the marginal gains are only relative to the local subset of the ground set). To see this, consider the counderexample in Figure 1. Here, our goal is to maximize a cardinality constrained coverage function

$$\max_{S \subseteq V, |S| \leq k} F(S) \qquad \text{where} \qquad F(S) = \left| \bigcup_{e \in S} \text{area}(e) \right| .$$

The area of an element is simply the sum of the numbers in the respective rectangles. The local elements (i.e., subsets of the cross-pattern in this example) at each machine are two or three nonempty sets and a several empty sets (displayed in the top row of Figure 1). The arrows indicate which elements each machine sends to the central machine. Note that the example is set up so that the marginal gains of the "border" elements (subsets in colors green, yellow, light blue, and dark blue) are 0 in the local runs of the greedy algorithm, and hence the local solutions can exclude an empty set instead of the border elements. However, at the central machine it is now impossible to find the optimal solution over the entire ground set because the border elements have already been discarded (for cardinality constraint $k = 6$, the optimal solution is a full cover).

Parametrized by the number of machines ℓ, this counterexample can be constructed so that the number of machines is $\ell^2 + 1$, the cardinality constraint is $k = \ell + \ell^2$, and the distributed greedy solution is $2\ell^2$. In contrast, a full cover with total area $\ell^2 + \ell^3$ is still possible, and hence the approximation ratio becomes arbitrarily small for large values of ℓ.

2.2.2 Randomized distributed greedy

While this counterexample demonstrates that the distributed greedy algorithm does not have a constant factor approximation guarantee, the algorithm usually works well in practice [MKSK13]. We now show that we can explain this behavior by assuming that the elements are distributed randomly across the machines. In particular, we consider the following algorithm for a set of m machines:

1. Randomly distribute the data set: for each element $e \in V$ and each machine i, assign element e to machine i with probability $\frac{1}{m}$. Let V_i be the set of elements assigned to machine i.
2. Run the greedy algorithm locally on each machine.

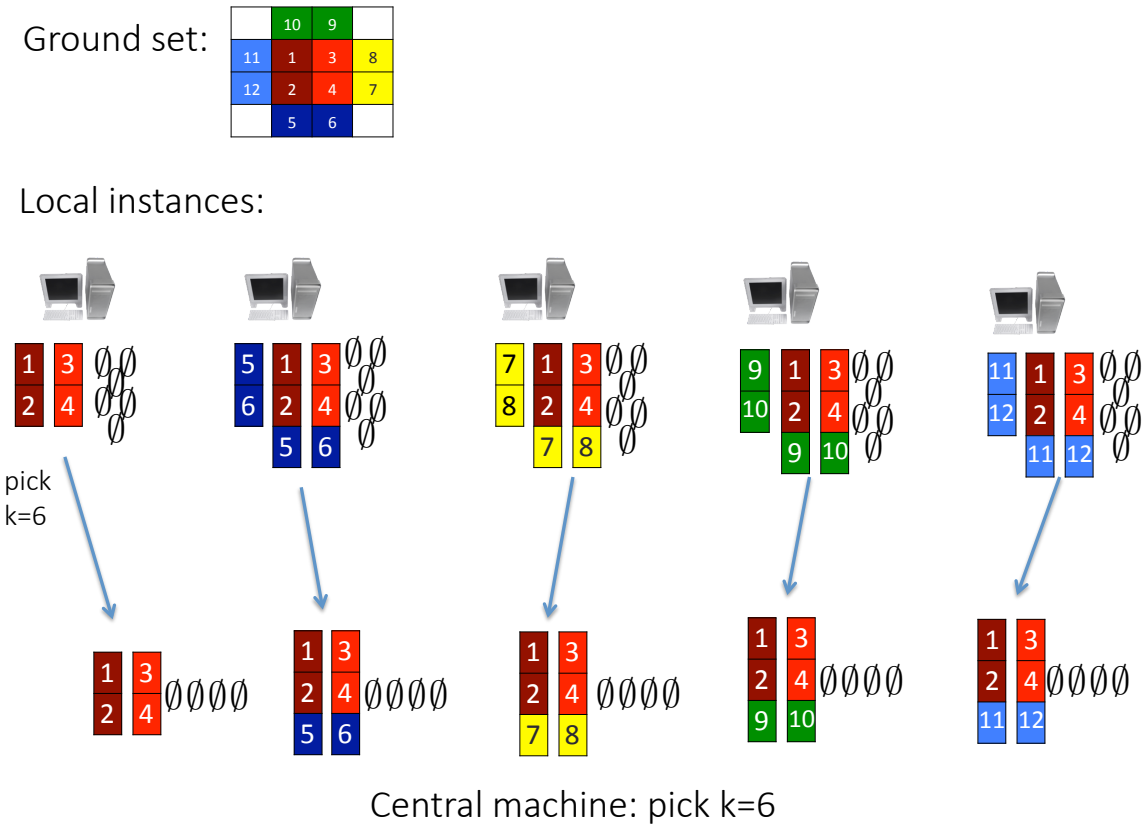

Figure 1: Counterexample for the distributed greedy algorithm.

3. Send the solution sets S_G^i to the central machine. Let $V_C = \bigcup_i S_G^i$ be the set of elements at the central machine.
4. Run the greedy algorithm centrally on V_C and let S_C be the solution..
5. Pick the best solution among the m local solutions and the central solution S_C. We call this solution S_{DG}.

This algorithm gives an approximation guarantee of

$$F(S_{\text{DG}}) \;=\; \frac{1}{2}\left(1-\frac{1}{e}\right)\text{OPT}\;.$$

2.2.3 Analysis

We now give a sketch of the analysis in [BENW15]. An important quantity is the following vector of probabilities $p \in \mathbb{R}^{|V|}$. In the definition of p, we let $A \subseteq V$ be a set of elements such that $\mathbb{P}[e \in A] = \frac{1}{m}$ for all $e \in V$. Moreover, for a set $B \subseteq V$, we let $\textsc{Greedy}(B)$ denote the solution of the greedy algorithm run on the set B.

$$p_e \;=\; \begin{cases} P_A[e \in \textsc{Greedy}(A \cup \{e\})] & \text{if } e \in S^* \\ 0 & \text{otherwise} \end{cases}\;.$$

We first prove the following lemma:

Lemma 2. *If machine i uses a greedy algorithm that guarantees*

$$F(S_G^i) \;\geq\; \alpha \max_{S \subseteq V_i, |S| \leq k} F(S)$$

then

$$\mathbb{E}[F(S_G^i)] \;\geq\; \alpha F(1_{S^*} - p)$$

where $f : \mathbb{R}^{|V|} \to \mathbb{R}$ is the Lovasz extension of F.

Proof. First, we define the following set:

$$O_i \;=\; \{e \in S^* \,|\, e \notin \textsc{Greedy}(V_i \cup O_i)\}\;.$$

From this definition, it follows that

$$S_G^i \;=\; \textsc{Greedy}(V_i) \;=\; \textsc{Greedy}(V_i \cup O_i)\;.$$

Using the assumption in the lemma, we now get

$$\begin{aligned} F(S^i) &\geq \alpha \max_{\substack{S \subseteq V_i \cup O_i \\ |S| \leq k}} F(S) \\ &\geq \alpha \cdot F(O_i) \ . \end{aligned}$$

Since the V_i are chosen randomly, we have

$$\mathbb{P}[e \in O_i] \ = \ 1 - \mathbb{P}[e \in O_i] \ = \ \begin{cases} 1 - p_e & \text{if } e \in S^* \\ 0 & \text{otherwise} \end{cases}$$

from the definition of the probability vector p. It is now easy to see that $\mathbb{E}[\mathbb{1}_{O_i}] = \mathbb{1}_{S^*} - p$. Combined with Jensen's inequality and the definition of the Lovasz extension, this yields

$$\begin{aligned} \mathbb{E}[F(S_G^i)] &\geq \alpha \mathbb{E}[F(O_i)] \\ &= \alpha \mathbb{E}[f(\mathbb{1}_{O_i})] \\ &\geq \alpha f(\mathbb{E}\mathbb{1}_{O_i}) \\ &= \alpha f(\mathbb{1}_{S^*} - p) \end{aligned}$$

which is precisely the statement of the lemma. □

For the proof of the overall approximation guarantee, we use another lemma that we only state.

Lemma 3. *If the central machine uses an algorithm that guarantees*

$$F(S_C) \geq \beta \max_{S \subseteq V_C, |S| \leq k} F(S)$$

then

$$\mathbb{E}[F(S_{\text{DG}})] \geq \beta f(p) \ .$$

We now sketch how the proof proceeds given the above two lemmas. First, note that we have

$$F(S_{\text{DG}}) \geq \max_i F(S_G^i) \qquad \text{and} \qquad F(S_{\text{DG}}) \geq F(S_C)$$

from the definition of the distributed greedy algorithm. Hence Lemma 2 implies $\mathbb{E}[F(S_{\text{DG}})] \geq \alpha f(\mathbb{1}_{S^*} - p)$ and Lemma 3 implies $\mathbb{E}[F(S_{\text{DG}})] \geq \beta f(p)$. Combining these two inequalities yields

$$\begin{aligned} (\alpha + \beta)\mathbb{E}[F(S_{\text{DG}})] &\geq \alpha\beta(f(\mathbb{1}_{S^*} - p) + f(p)) \\ &\geq \alpha\beta f(\mathbb{1}_{S^*} - p + p) \\ &= \alpha\beta F(S^*) \end{aligned}$$

where we used the convexity and positive homogeneity of the Lovasz extension. The above argument gives an overall approximation guarantee of

$$\mathbb{E}[F(S_{\text{DG}})] \;\geq\; \frac{\alpha\beta}{\alpha+\beta} F(S^*) \;.$$

2.2.4 Further remarks

The approach outlined above is fairly general and applies to *any* local (and central) algorithm with a constant approximation ratio. Moreover, the approach generalizes to various down-monotone constraints and non-monotone submodular maximization. Better approximation ratios can be achieved if items are selected with multiplicity. For further work on distributed submodular maximization, see [MKSK13, IMMM14, MZ15, BENW15].

3 Futher submodular problems

We ended the lecture with a high-level discussion of further submodular problems. So far, the class has mostly focused on submodular problems that can be solved fairly well (exact solution or constant factor approximation in polynomial time). However, there are other classes of problems that are significantly harder.

One example is submodular minimization with constraints. In the constrained setting, the Lovasz extension does not give an exact relaxation any more and there are several approximation lower bounds of the form $\Omega(n^c)$ for $c > 0$. These lower bounds are information-theoretic (lower bounds for oracle access to the objective function F) and apply to constraints such as perfect matchings or minimum cuts. In contrast to these lower bounds, maximization is usually more robust and still allows constant-factor approximations for many types of constraints in polynomial time. Solution strategies for constrained submodular minimization problems include convex relaxations combined with rounding, approximating the objective F with an easier function, and sampling-based approaches.

In practice there is often additional structure that allows us to avoid the worst case behavior on many instances. We briefly talked about an example in image segmentation. Many segmentation algorithms are based on min-cut approaches, which favor short boundaries. This leads to problems when segmenting long and thin objects such as tree branches. One way to design a more robust algorithm is via "cooperative cuts" [JB11]. The idea is that the cut boundary should not only be short but also homogeneous, which allows for a trade-off between homogeneity and cut energy. In practice, approximate algorithms work very well on these problem instances. One additional helpful property on these instances

is the notion of curvature, which quantifies the deviation from linearity of a submodular function. By taking curvature into account, it is possible to prove better approximation ratios [IJB13].

References

[BBCL14] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 946–957, 2014.

[BENW15] Rafael Barbosa, Alina Ene, Huy Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1236–1244, 2015.

[BMKK14] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 671–680, 2014.

[BV14] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1497–1514, 2014.

[CDPW14] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pages 629–638, 2014.

[CWY09] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 199–208, 2009.

[DLBS14] Nan Du, Yingyu Liang, Maria-Florina Balcan, and Le Song. Influence function learning in information diffusion networks. In *Proceedings of The 31st International Conference on Machine Learning (ICML)*, pages 2016–2024, 2014.

[IJB13] Rishabh K. Iyer, Stefanie Jegelka, and Jeff A. Bilmes. Curvature and optimal algorithms for learning and minimizing submodular functions. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 2742–2750. 2013.

[IMMM14] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 100–108, 2014.

[JB11] Stefanie Jegelka and Jeff Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1897–1904, 2011.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.

[KMVV13] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–10, 2013.

[LOS15] Brendan Lucier, Joel Oren, and Yaron Singer. Influence at scale: Distributed computation of complex contagion in networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 735–744, 2015.

[MBC+11] Michael Mathioudakis, Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Antti Ukkonen. Sparsification of influence networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 529–537, 2011.

[MBK+15] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1812–1818, 2015.

[Min78] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, pages 234–243. 1978.

[MKSK13] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2049–2057, 2013.

[MR10] Elchanan Mossel and Sebastien Roch. Submodularity of influence in social networks: From local to global. *SIAM Journal on Computing*, 39(6):2176–2188, 2010.

[MZ15] Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, pages 153–162, 2015.

[PJB14] Adarsh Prasad, Stefanie Jegelka, and Dhruv Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2645–2653, 2014.

[PJG+14] Xinghao Pan, Stefanie Jegelka, Joseph E. Gonzalez, Joseph K. Bradley, and Michael I. Jordan. Parallel double greedy submodular maximization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 118–126, 2014.

[TSX15] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1539–1554, 2015.

[TXS14] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 75–86, 2014.

[WIB14] Kai Wei, Rishabh K. Iyer, and Jeff A. Bilmes. Fast multi-stage submodular maximization. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pages 1494–1502, 2014.