Linear-Work Greedy Parallel Approximate Set Cover and Variants

Guy E. Blelloch Richard Peng Kanat Tangwongsan
Carnegie Mellon University
{guyb, yangp, ktangwon}@cs.cmu.edu

ABSTRACT

We present parallel greedy approximation algorithms for set cover and related problems. These algorithms build on an algorithm for solving a graph problem we formulate and study called Maximal Nearly Independent Set (MaNIS)—a graph abstraction of a key component in existing work on parallel set cover.

We derive a randomized algorithm for MaNIS that has $O(m)$ work and $O(\log^2 m)$ depth on input with m edges. Using MaNIS, we obtain RNC algorithms that yield a $(1+\varepsilon)H_n$-approximation for set cover, a $(1-\frac{1}{e}-\varepsilon)$-approximation for max cover and a $(4+\varepsilon)$-approximation for min-sum set cover all in linear work; and an $O(\log^* n)$-approximation for asymmetric k-center for $k \le \log^{O(1)} n$ and a $(1.861+\varepsilon)$-approximation for metric facility location both in essentially the same work bounds as their sequential counterparts.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Theory

Keywords

Parallel algorithms, approximation algorithms, set cover, max cover, facility location

1. INTRODUCTION

Set cover is one of the most fundamental and well-studied problems in optimization and approximation algorithms. This problem and its variants have a wide variety of applications in the real world, including locating warehouses, testing faults, scheduling crews on airlines, and allocating wavelength in wireless communication. Let $\mathcal{U}$ be a set of n ground elements, $\mathcal{F}$ be a collection of subsets of $\mathcal{U}$ covering $\mathcal{U}$ (i.e., $\cup_{S\in\mathcal{F}} S = \mathcal{U}$), and $c\colon \mathcal{F} \to \mathbb{R}_+$ a cost function. The *set cover problem* is to find the cheapest collection of sets $\mathcal{A} \subseteq \mathcal{F}$ such that $\cup_{S\in\mathcal{A}} S = \mathcal{U}$, where the cost of the solution $\mathcal{A}$ is specified by $c(\mathcal{A}) = \sum_{S\in\mathcal{A}} c(S)$. Unweighted set cover (all weights are equal) appeared as one of the 21 problems Karp identified as NP-complete in 1972 [Kar72]. Two years later, Johnson [Joh74] proved that the simple greedy method gives an approximation that is at most a factor $H_n = \sum_{k=1}^{n} \frac{1}{k}$ from optimal. Subsequently, Chvátal [Chv79] proved the same approximation bounds for the weighted case. These results are complemented by a matching hardness result: Feige [Fei98] showed that unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$, set cover cannot be approximated in polynomial time with a ratio better than $(1-o(1))\ln n$. This essentially shows that the greedy algorithm is optimal.

Not only is greedy set cover optimal but it also gives an extremely simple $O(M)$ time algorithm for the unweighted case and $O(M \log M)$ time for the weighted case, where $M \ge n$ is the sum of the sizes of the sets. In addition, ideas similar to greedy set cover have been successfully applied to max k-cover, min-sum set cover, k-center, and facility location, generally leading to optimal or good-quality, yet simple, approximation algorithms.

From a parallelization point of view, however, the greedy method is in general difficult to parallelize, because at each step, only the highest-utility option is chosen and every subsequent step is likely to depend on the preceding one. Berger, Rompel, and Shor [BRS94] (BRS) showed that the greedy set cover algorithm can be "approximately" parallelized by bucketing[1] utility values (in this case, the number of new elements covered per unit cost) by factors of $(1+\varepsilon)$ and processing sets within a bucket in parallel. Furthermore, the number of buckets can be kept to $O(\log n)$ by preprocessing. However, deciding which sets within each bucket to choose requires some care: although at a given time, many sets might have utility values within a factor of $(1+\varepsilon)$ of the current best option, the sets taken together might not cover as many *unique* elements as their utility values imply—shared elements can be counted towards only one of the sets. BRS developed a technique to subselect within a bucket by first further bucketing by cost, then set size, and finally element degree, and then randomly selecting sets with an appropriate probability. This leads to an $O(\log^5 M)$-depth and $O(m \log^4 M)$-work randomized algorithm, giving a $((1+\varepsilon)H_n)$-approximation on a PRAM. Rajagopalan and Vazirani [RV98] improved the

[1]The bucketing approach has also been used for other algorithms such as vertex cover [KVY94] and metric facility location [BT10].

SPAA'11, June 4–6, 2011, San Jose, California, USA.

depth to $O(\log^3(Mn))$ with work $O(M \log^2 M)$ but at the cost of a factor of two in the approximation (essentially a factor-$2(1+\varepsilon)H_n$ approximation).

In comparison to their sequential counterparts, none of these previous set-cover algorithms are work efficient—their work is asymptotically more than the time for the optimal sequential algorithm.[2] Work efficiency is important since it allows an algorithm to be applied efficiently to both a modest number of processors (one being the most modest) and a larger number. Even with a larger number of processors, work-efficient algorithms limit the amount of resources used and hence presumably the cost of the computation.

Our Contributions: In this paper, we abstract out the most important component of the bucketing approach, which we refer to as Maximal Nearly Independent Set (MaNIS), and develop an $O(m)$ work and $O(\log^2 m)$ depth algorithm for an input graph with m edges, on an EREW PRAM (*work* is the total operation count, and *depth* the number of steps on the PRAM). The MaNIS problem is to find a subset of sets such that they are nearly independent (their elements do not overlap too much), and maximal (no set can be added without introducing too much overlap). Since we have to look at the input, which has size $O(m)$, the algorithm is work efficient. The MaNIS abstraction allows us to reasonably easily apply it to several set-cover like problems. In particular, we develop the following work-efficient approximation algorithms:

— *Set cover.* We develop an $O(M)$ work, $O(\log^3 M)$ depth (parallel time) algorithm with approximation ratio $(1+\varepsilon)H_n$. For the unweighted case, the same algorithm gives a $(1+\varepsilon)(1+\ln(n/\mathsf{opt}))$ approximation where opt is the optimal set-cover cost.

— *Max cover.* We develop an $O(M)$ work, $O(\log^3 M)$ depth prefix-optimal algorithm with approximation ratio $(1-1/e-\varepsilon)$. This significantly improves the work bounds over a recent result [CKT10].

— *Min-sum set cover.* We develop an $O(M)$ work, $O(\log^3 M)$ depth algorithm with an approximation ratio of $(4+\varepsilon)H_n$. We know of no other RNC parallel approximation algorithms for this problem.

— *Asymmetric k-center.* We develop an $O(p(k+\log^* n))$ work, $O(k \log n + \log^3 n \log^* n)$ depth algorithm with approximation ratio $O(\log^* n)$, where $p = n(n-1)/2$ is the size of the table of distances between elements. The algorithm is based on the sequential algorithm of Panigrahy and Vishwanathan [PV98] and we know of no other RNC parallel approximation algorithms for this problem.

— *Metric facility location.* We develop an $O(p \log p)$ work, $O(\log^4 p)$ depth algorithm with approximation ratio $(1.861+\varepsilon)$, where $p = |F| \times |C|$ is the size of the distance table. The algorithm is based on the greedy algorithm of Jain et al. [JMM+03] and improves on the approximation ratio of $(3+\varepsilon)$ for the best previous RNC algorithm [BT10].

All these algorithms run on a CRCW PRAM but rely on only a few primitives discussed in the next section, and thus should be easily ported to other models.

2. PRELIMINARIES AND NOTATION

For a graph G, we denote by $\deg_G(v)$ the degree of the vertex v in G and use $N_G(v)$ to denote the neighbor set of the node v. Furthermore, we write $u \sim v$ for u is adjacent to v. Extending this notation, we write $N_G(X)$ to mean the neighbors of the vertex set X, i.e., $N_G(X) = \cup_{w \in X} N_G(w)$. We drop the subscript (i.e., writing $\deg(v)$, $N(v)$, and $N(X)$) when the context is clear. Let $V(G)$ and $E(G)$ denote respectively the set of nodes and the set of edges. We use the notation $\widetilde{O}(f(n))$ to mean $O(f(n)\,\mathrm{polylog}(n))$ and $[k]$ to denote the set $\{1, 2, \ldots, k\}$. An event happens with high probability (w.h.p.) if it happens with probability exceeding $1 - n^{-\Omega(1)}$. We analyze the algorithms in the PRAM model. We use both the EREW (Exclusive Read Exclusive Write) and CRCW (Concurrent Read Concurrent Write) variants of the PRAM, and for the CRCW, assume an arbitrary value is written. For an input of size n, we assume that every memory word has $O(\log n)$ bits. In our analysis, we are primarily concerned with minimizing the work while achieving polylogarithmic depth and less concerned with polylogarithmic factors in the depth since such measures are typically not robust across models. All algorithms we develop are in NC or RNC, so they have polylogarithmic depth.

The algorithms in this paper are all based on a bipartite graph $G = (A \cup B, E), E \subseteq A \times B$. In set cover, for example, we use A to represent the subsets $\mathcal{F}$ and B for the ground elements $\mathcal{U}$. In addition to operating over the vertices and edges of the graph, the algorithms need to copy a value from each vertex (on either side) to its incident edges, need to "sum" values from the incident edges of each vertex using a binary associative operator, and given $A' \subseteq A$ and $B' \subseteq B$ need to *subselect* the graph $G' = (A' \cup B', (A' \times B') \cap E)$.

For analyzing bounds, we assume that G is represented in a form of adjacency array we refer to as the *packed representation* of G. In this representation, the vertices in A and B and the edges in E are each stored contiguously, and each vertex has a pointer to a contiguous array of pointers to its incident edges. With this representation, all the operations mentioned in the previous paragraph can be implemented using standard techniques in $O(|G|)$ work and $O(\log |G|)$ depth on an EREW PRAM, where $|G| = |A| + |B| + |E|$. The set-cover algorithm also needs the following operation for constructing the packed representation.

Lemma 2.1 *Given a bipartite graph $G = (A \cup B, E)$ represented as an array of $a \in A$, each with a pointer to an array of integer identifiers for its neighbors in B, the packed representation of G can be generated with $O(|G|)$ work and $O(\log^2 |G|)$ depth (both w.h.p.) on a* CRCW PRAM.

Proof. We note that the statement of the lemma allows for the integer identifiers to be sparse and possibly much larger than $|B|$. To implement the operation use duplicate elimination over the identifiers for B to get a unique representative for each $b \in B$ and give these representatives contiguous integer labels in the range $[|B|]$. This can be done with hashing in randomized $O(|G|)$-work $O(\log^2 |G|)$-depth [BM98]. Now that the labels for B are bounded by $[|B|]$ we can use a bounded integer sort [RR89] to collect all edges pointing to the same $b \in B$ and generate the adjacency arrays for the vertices in B in randomized $O(n)$ work and $O(\log n)$ depth on a (arbitrary) CRCW PRAM. □

We will also use the following.

Lemma 2.2 *If $y_1, \ldots, y_n \in (0, 1]$ are drawn independently such that $\mathbf{Pr}\left[x_i \in \left(\frac{j-1}{n}, \frac{j}{n}\right]\right] \leq \frac{1}{n}$ for all $i, j = 1, \ldots, n$, then*

[2] We note that the sequential time for a weighted $(1+\varepsilon)H_n$-approximation for set-cover is $O(M)$ when using bucketing.

the keys $y_1, \ldots, y_n$ can be sorted in expected $O(n)$ work and $O(\log n)$ depth on an CRCW PRAM

PROOF. Use parallel radix sort to bucket the keys into $B_1, \ldots, B_n$, where B_j contains keys between $\left(\frac{j-1}{n}, \frac{j}{n}\right]$. This requires $O(n)$ work and $O(\log n)$ depth. Then, for each B_i, in parallel, we can sort the elements in the bucket in $O(|B_i|^2)$ work $O(|B_i|)$ depth using, for example, a parallel implementation of the insertion sort algorithm. The work to sort these buckets is $\mathbf{E}\left[\sum_i |B_i|^2\right] \leq 2n$. Furthermore, balls-and-bins analysis shows that for all i, $|B_i| \leq O(\log n)$ with high probability. Thus, the depth of the sorting part is bounded by $\mathbf{E}[\max_i |B_i|] \leq O(\log n)$. □

3. MANIS

We motivate the study of Maximal Nearly Independent Set (MANIS) by revisiting existing parallel algorithms for set cover. These algorithms define a notion of utility—the number of new elements covered per unit cost—for each available option (set). Each iteration then involves identifying and working on the remaining sets that have utility within a $(1+\varepsilon)$ factor of the current best utility value—and for fast progress, requires that the best option after an iteration has utility at most a $(1+\varepsilon)$ factor smaller than before. Among the sets meeting the criterion, deciding which ones to include in the final solution is non-trivial. Selecting any one of these sets leads to an approximation ratio within $(1+\varepsilon)$ of the strictly greedy algorithm but may not meet the fast progress requirement. Including all of them altogether leads to arbitrarily bad bounds on the approximation ratio (many sets are likely to share ground elements) but does ensure fast progress. To meet both requirements, we would like to select a "maximal" collection of sets that have small, bounded overlap—if a set is left unchosen, its utility must have dropped sufficiently. This leads to the following graph problem formulation, where the input bipartite graph models the interference between sets.

Definition 3.1 ((ε,δ)-MaNIS) *Let $\varepsilon, \delta > 0$. Given a bipartite graph $G = (A \cup B, E)$, we say that a set $J \subseteq A$ is a* (ε,δ) maximal nearly independent set*, or* (ε,δ)-MANIS*, if*

(1) Nearly Independent. *The chosen options do not interfere much with each other, i.e.,*

$$|N(J)| \;\geq\; (1-\delta-\varepsilon)\sum_{a\in J}|N(a)|.$$

(2) Maximal. *The unchosen options have significant overlaps with the chosen options, i.e., for all $a \in A \setminus J$,*

$$|N(a) \setminus N(J)| < (1-\varepsilon)|N(a)|$$

The first condition in this MANIS definition only provides guarantees on average—it ensures that *on average* each chosen option does not overlap much with each other. It is often desirable to have a stronger guarantee that provides assurance on a per-option basis. This motivates the following strengthened definition, which implies the previous definition.

Definition 3.2 (Ranked (ε,δ)-MaNIS) *Let $\varepsilon, \delta > 0$. Given a bipartite graph $G = (A \cup B, E)$, we say that a set $J = \{s_1, s_2, \ldots, s_k\} \subseteq A$ is a* ranked (ε,δ) maximal nearly independent set*, or a ranked* (ε,δ)-MANIS *for short, if*

(1) Nearly Independent. *There is an ordering (not part of the* MANIS *solution) $s_1, s_2, \ldots, s_k$ such that each chosen option s_i is almost completely independent of $s_1, s_2, \ldots, s_{i-1}$, i.e., for all $i = 1, \ldots, k$,*

$$|N(s_i) \setminus N(\{s_1, s_2, \ldots, s_{i-1}\})| \geq (1-\delta-\varepsilon)|N(s_i)|.$$

(2) Maximal. *The unchosen options have significant overlaps with the chosen options, i.e., for all $a \in A \setminus J$,*

$$|N(a) \setminus N(J)| < (1-\varepsilon)|N(a)|.$$

Under this definition, an algorithm for ranked MANIS only has to return a set J but not the ordering. Furthermore, the following fact is easy to verify:

Fact 3.3 *If J is a ranked (ε,δ)-*MANIS*, then every $J' \subseteq J$ satisfies $|N(J')| \geq (1-\delta-\varepsilon)\sum_{j\in J'}|N(j)|$.*

Connection with previous work: Both versions of MANIS can be seen as a generalization of maximal independent set (MIS). Indeed, when $\delta = \varepsilon = 0$, the problem is the maximal set packing problem, which can be solved using a maximal independent set algorithm [KW85, Lub86], albeit with $O(\log n)$ more work than the simple sequential algorithm that solves both versions of MANIS in $O(|E|)$ sequential time.

Embedded in existing parallel set-cover algorithms are steps that can be extracted to compute MANIS. We obtain from Berger et al. [BRS94] (henceforth, the BRS algorithm) an RNC^4 algorithm for computing $(\varepsilon, 8\varepsilon)$-MANIS in $O(|E|\log^3 n)$ work. Similarly, we extract from Rajagopalan and Vazirani [RV98] (henceforth, the RV algorithm) an RNC^2 algorithm for computing *ranked* $(\varepsilon^2, 1-\frac{1}{2(1+\varepsilon)}-\varepsilon^2)$-MANIS in $O(|E|\log|E|)$ work.

Unfortunately, neither of the existing algorithms, as analyzed, is work efficient. In addition, the existing analysis of the RV algorithm places a restriction on δ: even when ε is arbitrarily close to 0, we cannot have δ below $\frac{1}{2}$.

3.1 Linear-Work Ranked MaNIS

We present an algorithm for the ranked MANIS problem. Our algorithm is inspired by the RV algorithm. Not only is the algorithm work efficient but also it removes the $\frac{1}{2}$ restriction on δ, matching and surpassing the guarantees given by previous algorithms. To obtain these bounds, we need a new analysis that differs from that of the RV algorithm. Our algorithm can be modified to compute ranked (ε,δ)-MANIS for any $0 < \varepsilon < \delta$ in essentially the same work and depth bounds (with worse constants); however, for the sake of presentation, we settle for the following theorem:

Theorem 3.4 (Ranked MaNIS) *Fix $\varepsilon > 0$. For a bipartite graph $G = (A \cup B, E)$ in packed representation there exists a randomized* EREW PRAM algorithm $\texttt{MaNIS}_{(\varepsilon,3\varepsilon)}(G)$ that produces a ranked $(\varepsilon, 3\varepsilon)$-MANIS in $O(|E|)$ expected work and $O(\log^2|E|)$ expected depth.

Presented in Algorithm 3.1 is an algorithm for computing ranked MANIS. To understand this algorithm, we will first consider a natural sequential algorithm for $(\varepsilon, 3\varepsilon)$-MANIS—and discuss modifications that have led us to the parallel version. To compute MANIS, we could first pick an ordering of the vertices of A and consider them in this order: for

Algorithm 3.1 $\texttt{MaNIS}_{(\varepsilon,3\varepsilon)}$ — a parallel algorithm for computing ranked $(\varepsilon, 3\varepsilon)$-MaNIS

Input: a bipartite graph $G = (A \cup B, E)$.

Output: $J \subseteq A$ satisfying Definition 3.2.

Initialize $G^{(0)} = (A^{(0)} \cup B^{(0)}, E^{(0)}) = (A \cup B, E)$, and for each $a \in A$, $D(a) = |N_{G^{(0)}}(a)|$.

Set $t = 0$. Repeat the following steps until $A^{(t)}$ is empty:

1. For $a \in A^{(t)}$, randomly pick $x_a \in_R [0, 1]$
2. For $b \in B^{(t)}$, let $\varphi^{(t)}(b)$ be b's neighbor with maximum x_a
3. Pick vertices of $A^{(t)}$ chosen by sufficiently many in $B^{(t)}$:
$$J^{(t)} = \Big\{ a \in A^{(t)} \;\Big|\; \sum_{b \in B^{(t)}} \mathbf{1}_{\{\varphi^{(t)}(b)=a\}} \geq (1-4\varepsilon) D(a) \Big\}.$$
4. Update the graph by removing J and its neighbors, and elements of $A^{(t)}$ with too few remaining neighbors:
$B^{(t+1)} = B^{(t)} \setminus N_{G^{(t)}}(J^{(t)})$
$A^{(t+1)} = \{a \in A^{(t)} \setminus J^{(t)} : |N_{G^{(t)}}(a) \cap B^{(t+1)}| \geq (1-\varepsilon) D(a)\}$
$E^{(t+1)} = E^{(t)} \cap (A^{(t+1)} \times B^{(t+1)})$
5. $t = t + 1$

Finally, return $J = J^{(0)} \cup \cdots \cup J^{(t-1)}$.

each $a \in A$, if a has at least $(1-4\varepsilon)D(a)$ neighbors, we add a to the output and delete its neighbors; otherwise, set it aside. Thus, every vertex added certainly satisfies the nearly-independent requirement. Furthermore, if a vertex is not added, its degree must have dropped below $(1-\varepsilon)D(a)$, which ensures the maximality condition.

Algorithm 3.1 achieves parallelism in two ways. First, we adapt the selection process so that multiple vertices can be chosen together at the same time. Unlike the sequential algorithm, the parallel version can decide whether to include a vertex $a \in A$ without knowing the outcomes of the preceding vertices. This is done by making the inclusion condition more conservative: Assign each $b \in B$ to the first $a \in A$ in the chosen ordering—regardless of whether a will be included in the solution. Then, for each $a \in A$, include it in the solution if enough of its neighbors are assigned to it. This step is highly parallel and ensures that every vertex added satisfies the nearly-independent requirement. Unfortunately, this process by itself may miss vertices that must be included.

Second, we repeat the selection process until no more vertices can be selected but ensure that the number of iterations is small. As the analysis below shows, a random permutation allows the algorithm to remove a constant fraction of the edges, making sure that it will finish in a logarithmic number of iterations. Note that unlike before, the multiple iterations make it necessary to distinguish between the original degree of a vertex, $D(a)$, and its degree in the current iteration (which we denote by $\deg(a)$ in the proof). Furthermore, we need an clean-up step after each iteration to eliminate vertices that are already maximal so that they will not hamper progress in subsequent rounds.

Running Time Analysis: To prove the work and depth bounds, consider the potential function

$$\Phi^{(t)} \stackrel{\text{def}}{=} \sum_{a \in A^{(t)}} |N_{G^{(t)}}(a)|,$$

which counts the number of remaining edges. The following lemma shows that sufficient progress is made in each step:

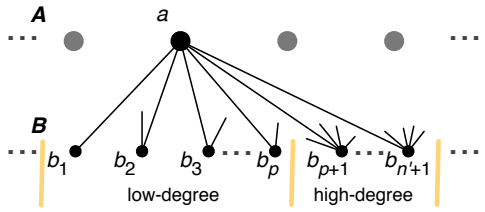


Figure 1: MaNIS analysis: for each $a \in A^{(t)}$, order its neighbors so that $N_{G^{(t)}}(a) = \{b_1, \ldots, b_{n'}\}$ and $\deg(b_1) \leq \deg(b_2) \leq \cdots \leq \deg(b_{n'})$, where $n' = \deg_{G^{(t)}}(a)$.

Lemma 3.5 *For $t \geq 0$, $\mathbf{E}\left[\Phi^{(t+1)}\right] \leq (1-c)\,\Phi^{(t)}$, where $c = \frac{1}{4}\varepsilon^2(1-\varepsilon)$.*

Before proceeding with the proof, we offer a high-level sketch. We say a vertex $a \in A^{(t)}$ deletes an edge (a', b) if $a \in J^{(t)}$ and $\varphi^{(t)}(b) = a$. In essence, the proof shows that for $a \in A^{(t)}$, the expected number of edges a deletes, denoted by Δ_a in the proof, is proportional to the degree of a. If a has few neighbors, it suffices to consider the probability that all neighbors select a. Otherwise, the proof separates the neighbors of a into high- and low- degree groups and analyzes Δ_a by averaging over possible values of x_a. In particular, it considers a y_a (i.e., $1 - \varepsilon/\deg(b_p)$ in the proof) such that for all $x_a \geq y_a$, there are likely sufficiently many low-degree neighbors that select a to ensure with constant probability that a is in $J^{(t)}$. Then, the proof shows that there is sufficient contribution to Δ_a from just the high-degree neighbors and just when $x_a \geq y_a$ (that is when a is likely in $J^{(t)}$). This is formalized in the proof below.

Proof. Consider an iteration t. Let $\deg(x) = \deg_{G^{(t)}}(x)$ and $\Delta_a = \mathbf{1}_{\{a \in J^{(t)}\}} \sum_{b:\varphi^{(t)}(b)=a} \deg(b)$. Thus, when a is included in $J^{(t)}$, Δ_a is the sum of the degrees of all neighbors of a that are assigned to a (by $\varphi^{(t)}$). It is zero otherwise if $a \notin J^{(t)}$. Since $\varphi^{(t)} : B^{(t)} \to A^{(t)}$ maps each $b \in B^{(t)}$ to a unique element in $A^{(t)}$, the sum of Δ_a over a is a lower bound on the number edges we lose in this round. That is,

$$\Phi^{(t)} - \Phi^{(t+1)} \geq \sum_{a \in A^{(t)}} \Delta_a,$$

so it suffices to show that for all $a \in A^{(t)}$, $\mathbf{E}[\Delta_a] \geq c \cdot \deg(a)$.

Let $a \in A^{(t)}$ be given and assume WLOG that $N_{G^{(t)}}(a) = \{b_1, \ldots, b_{n'}\}$ such that $\deg(b_1) \leq \deg(b_2) \leq \cdots \leq \deg(b_{n'})$. (as shown in Figure 1). Now consider the following cases:

— *Case 1. a has only a few neighbors:* Suppose $n' < \frac{2}{\varepsilon}$. Let $\mathcal{E}_1$ be the event that $x_a = \max\{x_{a'} : a' \in N_{G^{(t)}}(N_{G^{(t)}}(A^{(t)}))\}$. Then, $\mathcal{E}_1$ implies that (1) $a \in J^{(t)}$ and (2) $\varphi^{(t)}(b_{n'}) = a$. Therefore,

$$\mathbf{E}[\Delta_a] \geq \mathbf{Pr}[\mathcal{E}_1] \cdot \deg(b_{n'}) \geq \tfrac{1}{n'} \geq c \cdot \deg(a),$$

because $|N_{G^{(t)}}(N_{G^{(t)}}(A^{(t)}))| \leq n' \cdot \deg(b_{n'})$ and $n' = \deg(a) < 2/\varepsilon$.

— *Case 2. a has many neighbors*, i.e., $n' \geq \frac{2}{\varepsilon}$. Partition the neighbors of a into low- and high- degree elements as follows. Let $p = \lfloor (1-\varepsilon)\deg(a) \rfloor$, $L(a) = \{b_1, \ldots, b_p\}$, and $H(a) = \{b_{p+1}, \ldots, b_{n'}\}$. To complete the proof for this case, we rely on the following claim.

Claim 3.6 *Let* $\mathsf{select}_a^{(t)} = \{b \in B^{(t)} : \varphi^{(t)}(b) = a\}$, *and* $\mathcal{E}_2$ *be the event that* $|L(a) \setminus \mathsf{select}_a^{(t)}| \le 2\varepsilon|L(a)|$. *Then, (i) for* $\gamma \le \varepsilon/\deg(b_p)$, $\mathbf{Pr}[\mathcal{E}_2 | x_a = 1-\gamma] \ge \frac{1}{2}$; *and (ii) for* $b \in H(a)$ *and* $\gamma \le \varepsilon/\deg(b)$, $\mathbf{Pr}\left[\varphi^{(t)}(b) = a | \mathcal{E}_2, x_a = 1-\gamma\right] \ge 1-\varepsilon$.

Note that $\mathcal{E}_2$ implies $|\mathsf{select}_a^{(t)}| \ge n' - \varepsilon n' - 2\varepsilon n' \ge (1-4\varepsilon)D(a)$, because $n' \ge (1-\varepsilon)D(a)$. This in turn means that $\mathcal{E}_2$ implies $a \in J^{(t)}$. Applying the claim, we establish

$$\begin{aligned}
\mathbf{E}[\Delta_a] &\ge \sum_{b \in H(a)} \deg(b) \mathbf{Pr}\left[\mathcal{E}_2 \wedge \varphi^{(t)}(b) = a\right] \\
&\ge \sum_{b \in H(a)} \int_{\gamma=0}^{\frac{\varepsilon}{\deg(b)}} \deg(b) \mathbf{Pr}\left[\mathcal{E}_2 \middle| x_a = 1-\gamma\right] \\
&\qquad \mathbf{Pr}\left[\varphi^{(t)}(b) = a \middle| \mathcal{E}_2, x_a = 1-\gamma\right] d\gamma \\
&\ge \sum_{b \in H(a)} \varepsilon \frac{1}{2}(1-\varepsilon) \\
&\ge c \cdot \deg(a),
\end{aligned}$$

where the final step follows because $|H(a)| \ge \varepsilon n' \ge 1$. □

Proof of Claim 3.6: To prove (i), let $X \stackrel{\text{def}}{=} |L(a) \setminus \mathsf{select}_a^{(t)}| = \sum_{j \in L(a)} \mathbf{1}_{\{j \notin \mathsf{select}_a^{(t)}\}}$, so

$$\mathbf{E}[X | x_a = 1-\gamma] = \sum_{j \in L(a)} \mathbf{Pr}\left[j \notin \mathsf{select}_a^{(t)} | x_a = 1-\gamma\right].$$

Then, note that $j \in \mathsf{select}_a^{(t)}$ iff. $x_a = \max\{x_i : i \in N_{G^{(t)}}(j)\}$. Thus, for $j \in L(a)$,

$$\mathbf{Pr}\left[j \notin \mathsf{select}_a^{(t)} | x_a = 1-\gamma\right] \le 1 - \left(1 - \tfrac{\varepsilon}{\deg(b_p)}\right)^{\deg(j)} \le \varepsilon,$$

and so $\mathbf{E}[X | x_a = 1-\gamma] \le \varepsilon |L(a)|$. By Markov's inequality, we have $\mathbf{Pr}[\mathcal{E}_2 | x_a = 1-\gamma] \ge 1 - \frac{\varepsilon}{2\varepsilon} = \frac{1}{2}$.

We will now prove (ii). Consider that for $i \in N_{G^{(t)}}(b) \setminus \{a\}$, $\mathbf{Pr}[x_i > x_a | \mathcal{E}_2, x_a = 1-\gamma] \le \gamma$. Union bounds give

$$\begin{aligned}
\mathbf{Pr}\left[\varphi^{(t)}(b) = a | \mathcal{E}_2, x_a = 1-\gamma\right] &\ge 1 - \sum_{i \in N_{G^{(t)}}(b) \setminus \{a\}} \gamma \\
&\ge 1 - \varepsilon.
\end{aligned}$$

■

Each iteration can be implemented in $O(\Phi^{(t)})$ work and $O(\log \Phi^{(t)})$ depth on an EREW PRAM since beyond trivial parallel application and some summations, the iteration only involves the operations on the packed representation of G discussed in Section 2. Since $\Phi^{(t)}$ decreases geometrically (in expectation), the bounds follow. Algorithm 3.1 as described selects random reals x_a between 0 and 1. But it is sufficient for each a to use a random integer with $O(\log n)$ bits such that w.h.p., there are no collisions. In fact, since the x_a's are only compared, it suffices to use a random permutation over A since the distribution over the ranking would be the same.

4. LINEAR-WORK SET COVER

As our first example, in this section, we apply MaNIS to parallelize a greedy approximation algorithm for weighted set cover. Specifically, we prove the following theorem:

Theorem 4.1 *Fix* $0 < \varepsilon < \frac{1}{2}$. *For a set system* $(\mathcal{U}, \mathcal{F})$, *where* $|\mathcal{U}| = n$, *there is a randomized* $(1+\varepsilon)H_n$*-approximation for (weighted) set cover that runs in* $O(M)$ *expected work and* $O(\log^3 M)$ *expected depth on a* CRCW PRAM, *where* $M = \sum_{S \in \mathcal{F}} |S|$.

Algorithm 4.1 `SetCover` — parallel greedy set cover.

Input: a set cover instance $(\mathcal{U}, \mathcal{F}, c)$.

Output: a collection of sets covering the ground elements.

i. Let $\gamma = \max_{e \in \mathcal{U}} \min_{S \in \mathcal{F}} c(S)$,
$M = \sum_{S \in \mathcal{F}} |S|$,
$T = \log_{1/(1-\varepsilon)}(M^3/\varepsilon)$,
and $\beta = \frac{M^2}{\varepsilon \cdot \gamma}$.

ii. Let $(\mathcal{A}; A_0, \ldots, A_T) = \texttt{Prebucket}(\mathcal{U}, \mathcal{F}, c)$ and $\mathcal{U}_0 = \mathcal{U}$

iii. For $t = 0, \ldots, T$, perform the following steps:

1. Remove deleted elements from sets in this bucket: $A'_t = \{S \cap \mathcal{U}_t : S \in A_t\}$
2. Only keep sets that still belong in this bucket: $A''_t = \{S \in A'_t : c(S)/|S| > \beta \cdot (1-\varepsilon)^{t+1}\}$.
3. Select a maximal nearly independent set from the bucket: $J_t = \texttt{MaNIS}_{(\varepsilon, 3\varepsilon)}(A''_t)$.
4. Remove elements covered by J_t: $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus X_t$ where $X_t = \cup_{S \in J_t} S$
5. Move remaining sets to the next bucket: $A_{t+1} = A_{t+1} \cup (A'_t \setminus J_t)$

iv. Finally, return $\mathcal{A} \cup J_0 \cup \cdots \cup J_T$.

We describe a parallel greedy approximation algorithm for set cover in Algorithm 4.1. To motivate the algorithm, we discuss three ideas crucial for transforming the standard greedy set-cover algorithm into a linear-work algorithm with substantial parallelism: (1) approximate greedy through bucketing, (2) prebucketing and lazy bucket transfer, and (3) subselection via MaNIS. Despite the presence of some these ideas in previous work, it is the combination of our improved MaNIS algorithm and careful lazy bucket transfer that is responsible for better work and approximation bounds.

Like in previous algorithms, bucketing creates opportunities for parallelism at the round level, by grouping together sets by their coverage-per-unit-cost values in powers of $(1-\varepsilon)$. Consequently, there will be at most $O(\log_{1+\varepsilon} \rho)$ buckets (also rounds), where ρ is the ratio between the largest and the smallest coverage-per-unit-cost values. This, however, raises several questions (which are solved by ideas (2) and (3)).

First, *how can we make* ρ *small and keep the contents of the relevant buckets "fresh" in linear work?* As detailed in Lemma 4.2, the algorithm relies on a subroutine `Prebucket` that first preprocesses the input to keep ρ polynomially bounded by setting aside certain "cheap" sets that will be included in the final solution by default and throwing away sets that will never be used in the solution. It then classifies the sets into buckets $A_0, A_1, \ldots A_T$ by their utility; however, this initial bucketing will be stale as the algorithm progresses. While we cannot afford to reclassify the sets in every round, it suffices to maintain an invariant that each set in $S \in A_i$ satisfies $|S \cap \mathcal{U}_t|/c(S) \le \beta \cdot (1-\varepsilon)^i$. Further, we make sure that the bucket that contains the current best option is fresh—and move the sets that do not belong there accordingly.

Second, *what to do with the sets in each bucket to satisfy both the bucket invariant and the desired approximation ratio?* This is where we apply MaNIS. As previously discussed in Section 3, MaNIS allows the algorithm to choose nearly

non-overlapping sets, which helps bound the approximation guarantees and ensures that sets which MaNIS leaves out can be moved to the next bucket and satisfy the bucket invariant.

In the following lemma and proof, we use the definitions for γ, M, T, and β from Algorithm 4.1. Further, let opt denote the optimal cost.

Lemma 4.2 *There is an algorithm* `Prebucket` *that takes as input a set system $(\mathcal{U}, \mathcal{F}, c)$ and produces a set $\mathcal{A}$ such that $c(\mathcal{A}) \leq \varepsilon \cdot \mathsf{opt}$ and buckets $A_0, \ldots, A_T$ such that*

1. *for each $S \in \mathcal{F} \setminus \mathcal{A}$, either S costs more than $M\gamma$ or $S \in A_i$ for which $c(S)/|S| \in (\beta \cdot (1-\varepsilon)^{i+1}, \beta \cdot (1-\varepsilon)^i]$.*
2. *there exists a set cover solution costing at most opt using sets from $\mathcal{A} \cup A_0 \cup A_1 \cup \cdots \cup A_T$;*
3. *the algorithm runs in $O(M)$ work and $O(\log M)$ depth on a* CRCW PRAM.

Proof. We rely on the following bounds on opt [RV98]: $\gamma \leq \mathsf{opt} \leq M\gamma$. Two things are clear as a consequence: (i) if $c(S) \leq \varepsilon \cdot \frac{\gamma}{M}$, S can be included in $\mathcal{A}$, yielding a total cost at most $\varepsilon\gamma \leq \varepsilon \cdot \mathsf{opt}$. (ii) if $c(S) > M\gamma$, then S can be discarded (S is not part of any optimal solution).

Thus, we are left with sets costing between $\varepsilon \cdot \frac{\gamma}{M}$ and $M\gamma$. Compute $|S|/c(S)$ for each remaining set S, in parallel, and store S in A_i such that $c(S)/|S| \in (\beta \cdot (1-\varepsilon)^{i+1}, \beta \cdot (1-\varepsilon)^i]$. We know that $1/(M\gamma) \leq c(S)/|S| \leq M^2/(\varepsilon\gamma) = \beta$, so the buckets are numbered between $i = 0$ and $i = \log_{1/(1-\varepsilon)}(M^3/\varepsilon) = T$.

Computing M, γ, and $c(S)/|S|$ for all sets S can be done in $O(M)$ work and $O(\log M)$ depth using parallel sums. Once each set knows its bucket based on $c(S)/|S|$, a stable integer sort over integers in the range $[O(\log M)]$ can be used to collect them into buckets with the same work and depth bounds [RR89]. □

Approximation Guarantees: We follow a standard proof in Vazirani [Vaz01]. It should be noted that although we do not mention LPs here, the proof given below is similar in spirit to the dual-fitting proof presented by Rajagopalan and Vazirani [RV98]. Let $p_t = \frac{1}{\beta}(1-\varepsilon)^{-(t+1)}$. For each $e \in \mathcal{U}$, if e is covered in iteration t, define the price of this element to be $p(e) = p_t$. That is, every element covered in this iteration has the same price p_t. Now, Step 2 ensures that if $S \in A''_t$ can cover e, then $c(S)/|S| \leq p(e)$, where $|S|$ is the size of S after Step 2 in iteration t. Let $X_t = \cup_{S \in J_t} S$ be the set of elements covered in iteration t. The near independent property of $(\varepsilon, 3\varepsilon)$-MaNIS indicates that $|X_t| = |N(J_t)| \geq (1-4\varepsilon)\sum_{S \in J_t} N(S)$, where $N(\cdot)$ here is the neighborhood set in A''_t. Thus, $c(J_0 \cup \cdots \cup J_T)$ can be written as

$$\sum_t \sum_{S \in J_t} \frac{c(S)}{|S|} \cdot |S| \leq \sum_t p_t \sum_{S \in J_t} |S| \leq \frac{1}{1-4\varepsilon} \sum_{e \in \mathcal{U}} p(e).$$

Let $\mathcal{O}^*$ be any optimal solution. Consider a set $S^* \in \mathcal{O}^*$. Since all buckets $t' < t$ are empty, we know that $p_t \leq \frac{1}{1-\varepsilon} \min_{S \in A''_t} \frac{c(S)}{|S|}$. Furthermore, for each $e \in S^*$, let t_e denote the iteration in which e was covered. By greedy properties (as argued in [RV98, Vaz01]),

$$\sum_{e \in S^*} \min_{S \in G''_{t_e}} \frac{c(S)}{|S|} \leq \left(1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{|S^*|}\right) c(S^*).$$

Hence, $c(J_0 \cup \cdots \cup J_T) \leq \frac{1}{1-5\varepsilon} \sum_{S^* \in \mathcal{O}^*} H_{|S^*|} c(S^*) \leq \frac{1}{1-5\varepsilon} H_n \cdot \mathsf{opt} \leq (1+\varepsilon')H_n \cdot \mathsf{opt}$ (using $\varepsilon = O(\varepsilon')$), as promised.

Implementation and Work and Depth Bounds: To analyze the cost of the algorithm, we need to be more specific about the representation of all structures that are used. We assume the sets $S \in \mathcal{F}$ are given unique integer identifiers $[|\mathcal{F}|]$, and similarly for the elements $e \in \mathcal{U}$. Each set keeps a pointer to an array of identifiers for its elements, and each bucket keeps a pointer to an array of identifiers for its sets. The sets can shrink over time as elements are filtered out in Step 1 of each iteration of the algorithm. We keep a Boolean array indicating which of the elements from $\mathcal{U}$ remain in $\mathcal{U}_\sqcup$. Let $\mathcal{A}^{(t)}$ be the snapshot of A_t at the beginning of iteration t of Algorithm 4.1, and $M_t = \sum_{S \in \mathcal{A}^{(t)}} |S|$.

Claim: Iteration t of Algorithm 4.1 can be accomplished in expected $O(M_t)$ work and $O(\log^2 M_t)$ depth on the randomized CRCW PRAM.

Steps 1 and 2 use simple filtering on arrays of total length $O(M_t)$, which can be done with prefix sums. Step 3 requires converting adjacency arrays for each set in A''_t to the packed representation needed by MaNIS. The indices of the elements might be sparse, but this conversion can be done using Lemma 2.1. The cost of this conversion, as well as the cost of running MaNIS, is within the claimed bounds. Step 4 and 5 just involve setting flags, a filter, and an append, all on arrays of length $O(M_t)$.

Now there are $O(\log M)$ iterations, and `Prebucket` has depth $O(\log M)$, so the overall depth is bounded by $O(\log^3 M)$. To prove the work bound, we will derive a bound on $\sum_t M_t$. We note that every time a set is moved from one bucket to the next its size decreases by a constant factor, and hence the total work attributed to each set is proportional to its original size. More formally, we have the following claim:

Claim: If $S \in \mathcal{A}^{(t)}$, $|S| \leq c(S) \cdot \beta \cdot (1-\varepsilon)^t$.

This claim can be shown inductively: For any set $S \in \mathcal{F}$, `Prebucket` guarantees that the bucket that S went into satisfies the claim. Following that, this set can be shrunk and moved (Steps 1, 2, and 5). It is easy to check that the claim is satisfied (by noting Steps 2 and 5's criteria and that sets not chosen by MaNIS are shrunk by an ε fraction).

By this claim, the total sum $\sum_t M_t$ is at most

$$\sum_t \sum_{S \in \mathcal{A}^{(t)}} c(S) \cdot \beta \cdot (1-\varepsilon)^t \leq \sum_{S \in \mathcal{F}} \frac{1}{\varepsilon} \cdot c(S)\beta \cdot (1-\varepsilon)^{t_S},$$

where t_S is the bucket index of S in the initial bucketing. Furthermore, Lemma 4.2 indicates that $|S| \geq c(S) \cdot \beta \cdot (1-\varepsilon)^{t_S+1}$, showing that $\sum_t M_t = O(\frac{1}{\varepsilon}M)$ as $\varepsilon \leq \frac{1}{2}$. Since the work on each step is proportional to M_t, the overall work is $O(\frac{1}{\varepsilon}M)$.

5. SET COVERING VARIANTS

Using the `SetCover` algorithm from the previous section, we describe simple changes to the algorithm or the analysis that results in solutions to variants of set cover. In this section, *we will be working with unweighted set cover.*

Ordered vs. Unordered. We would like to develop algorithms for prefix-optimal max cover and min-sum set cover, using our set-cover algorithm; however, unlike set cover, these problems want an ordering on the chosen sets—not

just an unordered collection. As we now describe, minimal changes to the `SetCover` algorithm will enable it to output an ordered sequence of sets which closely approximate the greedy behavior. Specially, we will give an algorithm with the following property[3]: Let $\mathcal{T} \subseteq \mathcal{U}$ be given. Suppose there exist ℓ sets covering $\mathcal{T}$, and our parallel algorithm outputs an ordered collection $S_1, \ldots, S_p$ covering $\mathcal{U}$, then

Lemma 5.1 *For any $i \leq p$, the number of elements in $\mathcal{T}$ freshly covered by S_i, i.e., $|S_i \cap R_i|$, is at least $(1-5\varepsilon)|R_i|/\ell$, where $R_i = \mathcal{T} \setminus (\cup_{j<i} S_j)$ contains the elements of $\mathcal{T}$ that remain uncovered after choosing $S_1, \ldots, S_{i-1}$.*

We modify the `SetCover` algorithm as follows. Make `MaNIS` returns a totally ordered sequence, by sorting each $J^{(t)}$ by their x_a's values and stringing together the sorted sequences $J^{(0)}, J^{(1)}, \ldots$; this can be done in the same work-depth bounds (Lemma 2.2) in CRCW. Further, modify `SetCover` so that (1) `Prebucket` only buckets the sets (but will not throw away sets nor eagerly include some of them) and (2) its Step iv. returns a concatenated sequence, instead of a union. Again, this does not change the work-depth bound but outputs an ordered sequence.

Next, we show that the output sequence has the claimed property by proving the following technical claim (variables here refer to those in Algorithm 4.1). Lemma 5.1 is a direct sequence of this claim (note that the sets we output come from $J_0, J_1, \ldots$ in that order).

Claim 5.2 *For all $t \geq 0$, if $\widehat{J}_t \subseteq J_t$ and $\widehat{X}_t = \cup_{S \in \widehat{J}_t} S$, then $|\widehat{X}_t \cap \mathcal{T}| \geq (1-5\varepsilon) \cdot |\widehat{J}_t| \cdot |Q_t|/\ell$, where $Q_t = \mathcal{T} \setminus (\cup_{t'<t} X_{t'})$.*

Proof. Let $t \geq 0$. By our assumption, there exist ℓ sets that fully cover Q_t. An averaging argument shows that there must be a single set, among the remaining sets, with a coverage ratio of at least $|Q_t|/\ell$. Since at the beginning of iteration t, we have $A_{t'} = \emptyset$ for $t' < t$, it follows that $\tau_t \geq |Q_t|/\ell$, where $\tau_t = \beta \cdot (1-\varepsilon)^t$. Furthermore, all sets $S \in A''_t$ have the property that $|S| \geq \tau_t(1-\varepsilon)$. Furthermore, Fact 3.3 guarantees that $\widehat{J}_t$ covers, among $\mathcal{T}$, at least $|N(\widehat{J}_t)| \geq (1-4\varepsilon)\sum_{j \in \widehat{J}_t} |N(j)| \geq (1-4\varepsilon)(1-\varepsilon)\tau_t|\widehat{J}_t| \geq (1-5\varepsilon)|\widehat{J}_t||Q_t|/\ell$, proving the lemma. □

5.1 Max Cover

The max k-cover problem takes as input an integer $k > 0$ and a set system (generally unweighted), and the goal is to find k sets that cover as many elements as possible. The sequential greedy algorithm gives a $(1-1/e)$-approximation, which is tight assuming standard complexity assumptions. In the parallel setting, previous parallel set-covering algorithms do not directly give $(1-\frac{1}{e}-\varepsilon)$-approximation. But in the related MapReduce model, Chierichetti et al. [CKT10] give a $1-1/e-\varepsilon$-approximation, which gives rise to a $O(m \log^3 M)$-work algorithm in PRAM, where $M = \sum_{S \in \mathcal{F}} |S|$. This is not work efficient compared to the greedy algorithm, which runs in at most $O(M \log M)$ time for any k.

In this section, we give a factor-$(1-\frac{1}{e}-\varepsilon)$ prefix optimal algorithm for max cover. As in Chierichetti et al. [CKT10], we say that a sequence of sets $S_1, S_2, \ldots, S_p$ covering the whole ground elements is *factor-σ prefix optimal* if for all $k \leq p$, $|\cup_{i \leq k} S_i| \geq \sigma \cdot \mathsf{opt}_k$, where opt_k denotes the optimal coverage using k sets. More specifically, we prove the following theorem:

Theorem 5.3 *Fix $0 < \varepsilon < \frac{1}{2}$. There is a factor-$(1-\frac{1}{e}-\varepsilon)$ prefix optimal algorithm the max cover problem requiring $O(M)$ work and $O(\log^3 M)$ depth, where $M = \sum_{S \in \mathcal{F}} |S|$.*

Proof. Use the algorithm from Lemma 5.1, so the work-depth bounds follow immediately from set cover. To argue prefix optimality, let k be given and $OPT_k \subseteq \mathcal{F}$ be an optimal max k-cover solution. Applying Lemma 5.1 with $\mathcal{T} = OPT_k$ gives that $|R_{i+1}| \leq |R_i|(1-\frac{1}{k}(1-5\varepsilon))$ and $|R_1| = |OPT_k|$. Also, we know that using $S_1, \ldots, S_k$, we will have covered at least $OPT_k - |R_{k+1}|$ elements of OPT_k. By unfolding the recurrence, we have $OPT_k - |R_{k+1}| \geq OPT_k - OPT_k \cdot \exp\{-(1-5\varepsilon)\}$. Setting $\varepsilon = \frac{e}{5}\varepsilon'$ completes the proof. □

5.2 Special Case: Unweighted Set Cover

When the sets all have the same cost, we can derive a slightly different and stronger form of approximation guarantees for the same algorithm. We apply this bound to derive guarantees for asymmetric k-center in Section 5.4. The following corollary can be derived from Lemma 5.1 in a manner similar to the max-cover proof; we omit the proof in the interest of space.

Corollary 5.4 *Let $0 < \varepsilon \leq \frac{1}{2}$. For an unweighted set cover instance, set cover can be approximated with cost at most $\mathsf{opt}(1+\varepsilon)(1+\ln(n/\mathsf{opt}))$, where opt is the cost of the optimal set cover solution.*

5.3 Min-Sum Set Cover

Another important and well-studied set covering problem is the min-sum set cover problem: given a set system $(\mathcal{U}, \mathcal{F})$, the goal is to find a sequence $S_1, \ldots, S_{n'}$ to minimize the cost $cost(\langle S_1, \ldots, S_{n'} \rangle) \stackrel{\text{def}}{=} \sum_{e \in \mathcal{U}} \tau(e)$, where $\tau(e) \stackrel{\text{def}}{=} \min\{i : e \in S_i\}$. Feige et al. [FLT04] (also implicit in Bar-Noy et al. [BNBH+98]) showed that the standard set cover algorithm gives a 4-approximation, which is optimal unless $\mathsf{P} = \mathsf{NP}$. The following theorem shows that this carries over to our parallel set cover algorithm:

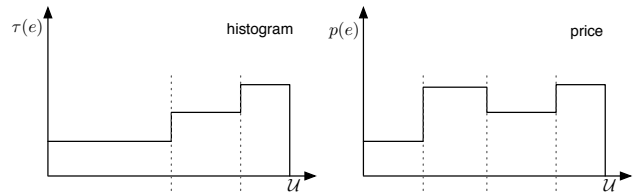
Theorem 5.5 *Fix $0 < \varepsilon \leq \frac{1}{2}$. There is a parallel $(4+\varepsilon)$-approximation algorithm for the min-sum set cover problem that runs in $O(M)$ work and $O(\log^3 M)$ depth.*

Proof. Consider the modified algorithm in Lemma 5.1. Suppose it outputs a sequence of sets $\mathsf{Alg} = \langle S_1, S_2, \ldots, S_{n'} \rangle$ covering $\mathcal{U}$, and an optimal solution is $\mathcal{O}^* = \langle O_1, \ldots, O_q \rangle$.

Let $\alpha_i = S_i \setminus (\cup_{j<i} S_j)$ denote the elements freshly covered by S_i and $\beta_i = \mathcal{U} \setminus (\cup_{j<i} S_j)$ be the elements not covered by $S_1, \ldots, S_{i-1}$. Thus, $|\beta_i| = |\mathcal{U}| - \sum_{j<i} |\alpha_j|$. Following Feige et al. [FLT04], the cost of our solution is $cost(\mathsf{Alg}) = \sum_{i>0} i \cdot |\alpha_i| = \sum_{i>0} |\beta_i|$, which can be rewritten as $\sum_{i>0} \sum_{e \in \alpha_i} \frac{|\beta_i|}{|\alpha_i|} = \sum_{e \in \mathcal{U}} p(e)$, where the price $p(e) = \frac{|\beta_i|}{|\alpha_i|}$ for i such that $e \in \alpha_i$. We will depict and argue about these costs pictorially as follows. First, the "histogram" diagram (below) is made up of $|\mathcal{U}|$ horizontal columns, ordered from left to right in the order the optimal solution covers them.

[3] This is the analog of the following fact from the sequential greedy algorithm [You95, PV98]: if there exist ℓ sets covering $\mathcal{T}$, and greedy picks sets $\chi_1, \ldots, \chi_p$ (in that order) covering $\mathcal{U}$, then for $i \leq p$, the number of elements in $\mathcal{T}$ freshly covered by χ_i is at least $|R_i|/\ell$, where $R_i = \mathcal{T} \setminus (\cup_{j<i} \chi_j)$.

The height of column $e \in \mathcal{U}$ is its $\tau(e)$ in the optimal solution. Additionally, the "price" diagram is also made up of $|\mathcal{U}|$ columns, though ordered from left to right in the order our solution covers them; the height of column e is $p(e)$.



We can easily check that (1) the histogram curve has area $\mathsf{opt} = cost(\mathcal{O}^*)$ and (2) the price curve has area $cost(\mathsf{Alg})$. We will show that shrinking the price diagram by 2 horizontally and θ vertically (θ to be chosen later) allows it to lie completely inside the histogram when they are aligned on the bottom-right corner. Let $p = (x, y)$ be a point inside (or on) the price diagram. Suppose p lies in the column $e \in \alpha_i$, so $y \le p(e) = |\beta_i|/|\alpha_i|$—and p is at most $|\beta_i|$ from the right.

When shrunk, p will have height $h = p(e)/\theta$ and width—the distance from the right end—$r = \frac{1}{2}|\beta_i|$. We estimate how many elements inside β_i are covered by the optimal solution using its first h sets. Of all the sets in $\mathcal{F}$, there exists a set S such that $|S \cap \beta_i| \ge |O_j^* \cap \beta_i|$ for all $j < i$.

Arguing similarly to previous proofs in this section, we have that $|\alpha_i| \ge (1 - 5\varepsilon)|S|$, so at this time, the optimal algorithm could have covered at most $h \cdot \frac{1}{1-5\varepsilon}|\alpha_i|$. Setting $\theta = \frac{2}{1-5\varepsilon}$ gives that the first h sets of $\mathcal{O}^*$ will leave $|\beta_i| - \frac{1}{2}|\beta_i| \ge \frac{1}{2}|\beta_i| = r$ elements of β_i remaining. Therefore, the scaled version of p lies inside the histogram, proving that the algorithm is a 2θ-approximation. By setting $\varepsilon = \frac{1}{40}\varepsilon'$, we have $2\theta = \frac{4}{1-5\varepsilon} \le 4 + \varepsilon'$, which completes the proof. □

5.4 Application: Asymmetric k-Center

Building on the set cover algorithm we just developed, we present an algorithm for asymmetric k-center. The input is an integer $k > 0$ and a distance function $d\colon V \times V \to \mathbb{R}_+$, where V is a set of n vertices; the goal is to find a set $F \subseteq V$ of k centers that minimizes the objective $\max_{j \in V} \min_{i \in F} d(i, j)$, where $d(x, y)$, which needs not be symmetric, denotes the distance from x to y. The distance d, however, is assumed to satisfy the triangle inequality, i.e., $d(x, y) \le d(x, z) + d(z, y)$. For symmetric d, there is a 2-approximation for both the sequential [HS85, Gon85] and parallel settings [BT10]. This result is optimal assuming $\mathsf{P} \neq \mathsf{NP}$. However, when d is not symmetric—hence the name asymmetric k-center, there is a $O(\log^* n)$-approximation in the sequential setting, which is also optimal unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log \log n)})$ [CGH+05], but nothing was previously known for the parallel setting.

In this section, we develop a parallel factor-$O(\log^* n)$ algorithm for this problem, based on the (sequential) algorithm of Panigrahy and Vishwanathan [PV98]. Thier algorithm consists of two phases: recursive cover and find-and-halve.

Recursive Cover for Asymmetric k-Center. The recursive cover algorithm of Panigrahy and Vishwanathan [PV98] (shown below) is easy to parallelize given the set cover routine from Corollary 5.4. Here, V is the input set of vertices.

Set $A_0 = A$ and $i = 0$. While $|A_i| > 2k$, repeat the following:

1. Construct a set cover instance $(\mathcal{U}, \mathcal{F})$, where $\mathcal{U} = A_i$ and $\mathcal{F} = \{S_1, \ldots, S_{|V|}\}$ such that $S_x = \{y \in A_i : d(x, y) \le r\}$.
2. $B = \texttt{SetCover}(\mathcal{U}, \mathcal{F})$.
3. $A_{i+1} = B \cap A$ and $i = i + 1$.

Let $n = |A|$. Assuming d is given as a distance matrix, Step 1 takes $O(n^2)$ work and $O(\log n)$ depth (to generate the packed representation). In $O(n^2)$ work and $O(\log^3 n)$ depth, Steps 2 and 3 can be implemented using the set-cover algorithm (Section 5.2) and standard techniques. Following [PV98]'s analysis, we know the number of iterations is at most $O(\log^* n)$. Therefore, this recursive cover requires $O(n^2 \log^* n)$ work and $O(\log^* n \log^3 n)$ depth.

Next, the find-and-halve phase will be run sequentially beyond trivial parallization. We have the following theorem:

Theorem 5.6 *Let $\varepsilon > 0$. There is a $O(n^2 \cdot (k + \log^* n))$-work $O(k \cdot \log n + \log^3 n \log^3 n)$-depth factor-$O(\log^* n)$ approximation algorithm for the asymmetric k-center problem*

Note that the algorithm performs essentially the same work as the sequential one. Furthermore, for $k \le \log^{O(1)} n$, this is an RNC algorithm. As suggested in [PV98], the recursive cover procedure alone yields a bicriteria approximation, in which the solution consists of $2k$ centers and costs at most $O(\log^* n)$ more than the optimal cost. This bicriteria approximation has $O(\log^{O(1)} n)$ depth for any k.

6. GREEDY FACILITY LOCATION

Metric facility location is a fundamental problem in approximation algorithms. The input consists of a set of *facilities* F and a set of *clients* C, where each facility $i \in F$ has cost f_i, and each client $j \in C$ incurs $d(j, i)$ to use facility i—and the goal is to find a set of facilities $F_S \subseteq F$ that minimizes the objective function $\Phi_{\mathsf{F}}(F_S) = \sum_{i \in F_S} f_i + \sum_{j \in C} d(j, F_S)$. The distance d is assumed to be symmetric and satisfy the triangle inequality. This problem has an exceptionally simple factor-1.861 greedy algorithm due to Jain et al. [JMM+03], which has been parallelized by Blelloch and Tangwongsan [BT10], yielding an RNC $(6+\varepsilon)$-approximation with work $O(p \log^2 p)$, where $p = |F| \times |C|$.

Using ideas from previous sections, we develop an RNC algorithm with an improved approximation guarantee, which essentially matches that of the sequential version.

Theorem 6.1 *Let $\varepsilon > 0$ be a small constant. There is a $O(p \log p)$-work $O(\log^4 p)$-depth factor-$(1.861 + \varepsilon)$ approximation algorithm for the (metric) facility location problem.*

We need the following definition for the algorithm:

Definition 6.2 (Star, Price, and Maximal Star) *A* star *$\mathcal{S} = (i, C')$ consists of a facility i and a subset of clients $C' \subseteq C$. The* price *of $\mathcal{S}$ is $\pi(\mathcal{S}) = (f_i + \sum_{j \in C'} d(j, i))/|C'|$. A star $\mathcal{S}$ is said to be* maximal *if all* strict *super sets of C' have a larger price, i.e., for all $C'' \supsetneq C'$, $\pi((i, C'')) > \pi((i, C'))$. Let $\mathsf{best}(i)$ be the price of the lowest-priced maximal star centered at i (on the current/remaining instance).*

Presented in Algorithm 6.1 is a parallel greedy approximation algorithm for metric facility location. The algorithm closely mimics the behaviors of Jain et al.'s algorithm, except the parallel algorithm is more aggressive in choosing the stars to open. Consider the natural integer-program formulation of facility location for which the relaxation yields the pair of

$$\textbf{Minimize} \quad \sum_{i\in F, j\in C} d(j,i)x_{ij} \;+\; \sum_{i\in F} f_i y_i$$

$$\textbf{Subj. to:} \begin{cases} \sum_{i\in F} x_{ij} & \geq & 1 & \text{for } j \in C \\ y_i - x_{ij} & \geq & 0 & \text{for } i \in F, j \in C \\ x_{ij} \geq 0,\ y_i \geq 0 \end{cases}$$

$$\textbf{Maximize} \quad \sum_{j\in C} \alpha_j$$

$$\textbf{Subj. to:} \begin{cases} \sum_{j\in C} \beta_{ij} & \leq & f_i & \text{for } i \in F \\ \alpha_j - \beta_{ij} & \leq & d(j,i) & \text{for } i \in F, j \in C \\ \beta_{ij} \geq 0,\ \alpha_j \geq 0 \end{cases}$$

Figure 2: The primal (left) and dual (right) programs for metric (uncapacitated) facility location.

Algorithm 6.1 Parallel greedy algorithm for metric facility location.

Set $F_A = \emptyset$. For $t = 1, 2, \ldots$, until $C = \emptyset$,

i. Let $\tau^{(t)} = (1+\varepsilon)\cdot \min\{\mathsf{best}(i) : i \in F\}$ and $F^{(t)} = \{i \in F : \mathsf{best}(i) \leq \tau^{(t)}\}$.

ii. Build a bipartite graph G from $F^{(t)}$ and $N(\cdot)$, where for each $i \in F^{(t)}$, $N(i) = \{j \in C : d(j,i) \leq \tau^{(t)}\}$.

iii. While $(F^{(t)} \neq \emptyset)$, repeat:

1. Compute $\Delta J^{(t)} = \mathtt{MaNIS}_{(\varepsilon,3\varepsilon)}(G)$ and $J^{(t)} = J^{(t)} \cup \Delta J^{(t)}$
2. Remove $\Delta J^{(t)}$ and $N(\Delta J^{(t)})$ from G, remove the clients $N(\Delta J^{(t)})$ from C, and set $f_i = 0$ for all $i \in \Delta J^{(t)}$.
3. Delete any $i \in F^{(t)}$ such that $\pi((i, N(i))) > \tau^{(t)}$.

primal (left) and dual (right) programs shown in Figure 2. We wish to give a dual-fitting analysis similar to that of Jain et al.

Their proof shows that the solution's cost is equal to the sum of α_j's over the clients, where α_j is the price of the star with which client j is connected up. Following this analysis, we set α_j to $\tau^{(t)}/(1+\varepsilon)$ where t is the iteration that the client was removed. Note that stars chosen in $F^{(t)}$ may have overlapping clients. For this reason, we cannot afford to open them all, or we would not be able to bound the solution's cost by the sum of α_j's. This situation, however, is rectified by the use of MaNIS, allowing us to prove Lemma 6.5, which relates the cost of the solution to α_j's. Before we prove this lemma, two easy-to-check facts are in order:

Fact 6.3 *In the graph G constructed in Step 2, for all $i \in F^{(t)}$, $\pi((i, N(i))) \leq \tau^{(t)}$.*

Fact 6.4 *At any point during iteration t, $\mathsf{best}(i) \leq \tau^{(t)}$ if and only if $\pi((i, N(i))) \leq \tau^{(t)}$.*

Lemma 6.5 *The cost of the algorithm's solution $\Phi_{\mathsf{F}}(F_A)$ is upper-bounded by $\frac{1}{1-5\varepsilon}\sum_{j\in C}\alpha_j$.*

Proof. Let $t > 0$ and consider what happens inside the inner loop (Steps iii.1—iii.3). Each iteration of the inner loop runs MaNIS on $F^{(t)}$ with each $i \in F^{(t)}$ satisfying $\tau^{(t)} \geq \pi((i, N(i))) = (f_i + \sum_{j\in N(i)} d(j,i))/|N(i)|$, so

$$|N(i)| \geq \tfrac{1}{\tau^{(t)}}\Big(f_i + \textstyle\sum_{j\in N(i)} d(j,i)\Big).$$

Because of Fact 6.3 and Step iii.3, the relationship $\tau^{(t)} \geq \pi((i, N(i)))$ is maintained throughout iteration t. Running a $(\varepsilon, 3\varepsilon)$-MaNIS on $F^{(t)}$ ensures that each $\Delta J^{(t)}$ satisfies $|N(\Delta J^{(t)})| \geq (1-4\varepsilon)\sum_{i\in\Delta J^{(t)}} |N(i)|$. Thus,

$$\sum_{j\in\Delta J^{(t)}} \alpha_j = \frac{\tau^{(t)}}{1+\varepsilon}|N(\Delta J^{(t)})| \geq \frac{\tau^{(t)}}{1+\varepsilon}(1-4\varepsilon)\sum_{i\in\Delta J^{(t)}} |N(i)|$$
$$\geq (1-5\varepsilon)\sum_{i\in\Delta J^{(t)}}\Big(f_i + \sum_{j\in N(i)} d(j,i)\Big),$$

which is at least

$$(1-5\varepsilon)\Big(\sum_{i\in\Delta J^{(t)}} f_i + \sum_{j\in N(\Delta J^{(t)})} d(j, \Delta J^{(t)})\Big).$$

Since every client has to appear in at least one $J^{(t)}$, summing across the inner loop's iterations and t gives the lemma. □

In the series of claims that follows, we show that when scaled down by a factor of $\gamma = 1.861$, the α setting determined above is a dual feasible solution. We will assume without loss of generality that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{|C|}$. Let $W_i = \{j \in C : \alpha_j \geq \gamma \cdot d(j,i)\}$ for all $i \in F$ and $W = \cup_i W_i$.

Claim 6.6 *For any facility $i \in F$ and client $j_0 \in C$,*

$$\textstyle\sum_{j\in W: j\geq j_0} \max(0, \alpha_{j_0} - d(j,i)) \;\leq\; f_i.$$

Proof. Suppose for a contradiction that there exist client j and facility i such that the inequality in the claim does not hold. Let t be the iteration such that $\alpha_{j_0} = \tau^{(t)}/(1+\varepsilon)$. Let $\widehat{C}^{(t)}$ be the set of clients j's such that $\alpha_{j_0} - d(j,i) > 0$ that remain at the beginning of iteration t. Thus, by our assumption and the fact that $\{j \in W : j \geq j_0 \wedge \alpha_{j_0} > d(j,i)\} \subseteq \widehat{C}^{(t)}$, we establish $\sum_{j\in\widehat{C}^{(t)}} \alpha_{j_0} - d(j,i) = \sum_{j\in\widehat{C}^{(t)}} \max(0, \alpha_{j_0} - d(j,i)) \geq \sum_{j\in W: j\geq j_0} \max(0, \alpha_{j_0} - d(j,i)) > f_i$. It follows that $\alpha_{j_0} > \frac{1}{|\widehat{C}^{(t)}|}(f_i + \sum_{j\in\widehat{C}^{(t)}} d(j,i))$. Hence, $\tau^{(t)}/(1+\varepsilon) = \alpha_{j_0} > \frac{1}{|\widehat{C}^{(t)}|}(f_i + \sum_{j\in\widehat{C}^{(t)}} d(j,i)) \geq \mathsf{best}(i) \geq \tau^{(t)}/(1+\varepsilon)$ since $\tau^{(t)}/(1+\varepsilon)$ is the minimum of the best price in that iteration. This gives a contradiction, proving the claim. □

Claim 6.7 *Let $i \in F$, and $j, j' \in W$ be clients. Then, $\alpha_j \leq \alpha_{j'} + d(i,j') + d(i,j)$.*

Proof. If $\alpha_j \leq \alpha_{j'}$, the proof is trivial, so assume $\alpha_j > \alpha_{j'}$. Let i' be any facility that removed j' (i.e, $i' \in \Delta J^{(t)}$ such that $j \in N(i')$). It suffices to show that $\alpha_j \leq d(i',j)$ and the claim follows from triangle inequality. Since $\alpha_j > \alpha_{j'}$, in the iteration t where $\alpha_j = \tau^{(t)}/(1+\varepsilon)$, we know that $f_{i'}$ has already been set to 0, so $\mathsf{best}(i') \leq d(j,i')$. Furthermore, in this iteration, $\alpha_j \leq \mathsf{best}(i')$ as $\alpha_j = \min\{\mathsf{best}(i)\}$, proving the claim. □

These two claims are sufficient to set up a factor-revealing LP identical to Jain et al.'s. Therefore, the following lemma follows from Jain et al. [JMM+03] (Lemmas 3.4 and 3.6):

Lemma 6.8 *The setting $\alpha'_j = \frac{\alpha_j}{\gamma}$ and $\beta'_{ij} = \max(0, \alpha'_j - d(j,i))$ is a dual feasible solution, where $\gamma = 1.861$.*

Combining this lemma with Lemma 6.5 and weak duality, we have the promised approximation guarantee.

Running time analysis: Fix $\varepsilon > 0$. We argue that the number of rounds is upper bounded by $O(\log p)$. For this, we need a preprocessing step which ensures that the ratio between the largest τ and the smallest τ ever encountered in the algorithm is $O(p^{O(1)})$. This can be done in $O(p)$ work and $O(\log(|F| + |C|))$ depth, adding $\varepsilon \cdot \mathsf{opt}$ to the solution's cost [BT10]. Armed with that, it suffices to show the following claim.

Claim 6.9 $\tau^{(t+1)} \geq (1+\varepsilon) \cdot \tau^{(t)}$.

PROOF. Let $\mathsf{best}^{(t)}(i)$ denote $\mathsf{best}(i)$ at the beginning of iteration t. Let i^* be the facility whose $\mathsf{best}^{(t+1)}(i^*)$ attains $\tau^{(t+1)}/(1+\varepsilon)$. To prove the claim, it suffices to show that $\mathsf{best}^{(t+1)}(i^*) \geq \tau^{(t)}$, as this will imply $\tau^{(t+1)} \geq (1+\varepsilon) \cdot \tau^{(t)}$. Now consider two possibilities.

— *Case 1.* i^* was part of $F^{(t)}$, so then either i^* was opened in this iteration or i^* was removed from $F^{(t)}$ in Step 3.iii. If i^* was opened, all clients at distance at most $\tau^{(t)}$ from it would be connected up, so $\mathsf{best}^{(t+1)}(i^*) \geq \tau^{(t)}$. Otherwise, i^* was removed in Step iii.3, in which case $\mathsf{best}^{(t+1)}(i^*) \geq \tau^{(t)}$ by the removal criteria and Fact 6.4.

— *Case 2.* Otherwise, i^* was not part of $F^{(t)}$. This means that $\mathsf{best}^{(t)}(i^*) > \tau^{(t)}$. As the set of unconnected clients can only become smaller, the price of the best star centered at i^* can only go up. So $\mathsf{best}^{(t+1)}(i^*)$ will be at least $\tau^{(t)}$, which in turn implies the claim. □

Thus, the total number of iterations (outer loop) in the algorithm is $O(\log p)$. We now consider the work and depth of each iteration. Step 1 involves computing $\mathsf{best}(i)$ for all $i \in F$. This can be done in $O(p)$ work and $O(\log p)$ depth using a prefix computation and standard techniques (see [BT10] for details). Step 2 can be done in the same work-depth bounds. Inside the inner loop, each MANIS call requires $O(p')$ work and $O(\log^2 p')$ depth, where m' is the number of edges in G. Steps iii.2–3 do not require more than $O(p')$ work and $O(\log p')$ depth. Furthermore, note that if $i \in F^{(t)}$ is not chosen by MANIS, it loses at least an ε fraction of its neighbors. Therefore, the total work of in the inner loop (for each t) is $O(\varepsilon^{-1}p)$, and depth $O(\log^3 p)$. Combining these gives the theorem.

7. CONCLUSION

We formulated and studied MANIS—a graph abstraction of a problem at the crux of many (set) covering-type problem. We gave a linear-work RNC solution to this problem and applied it to derive parallel approximation algorithms for several problems, yielding RNC algorithms for set cover, (prefix-optimal) max cover, min-sum set cover, asymmetric k-center, and metric facility location.

Acknowledgments. This work is partially supported by the National Science Foundation under grant number CCF-1018188 and by generous gifts from IBM, Intel, and Microsoft. We thank Anupam Gupta for valuable suggestions and conversations.

References

[BM98] Guy E. Blelloch and Bruce M. Maggs. *Handbook of Algorithms and Theory of Computation*, chapter Parallel Algorithms. CRC Press, Boca Raton, FL, 1998.

[BNBH+98] Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, and Tami Tamir. On chromatic sums and distributed resource allocation. *Inform. and Comput.*, 140(2):183–202, 1998.

[BRS94] Bonnie Berger, John Rompel, and Peter W. Shor. Efficient *NC* algorithms for set cover with applications to learning and geometry. *J. Comput. Syst. Sci.*, 49(3):454–477, 1994.

[BT10] Guy E. Blelloch and Kanat Tangwongsan. Parallel approximation algorithms for facility-location problems. In *SPAA*, pages 315–324, 2010.

[CGH+05] Julia Chuzhoy, Sudipto Guha, Eran Halperin, Sanjeev Khanna, Guy Kortsarz, Robert Krauthgamer, and Joseph Naor. Asymmetric k-center is $\log^* n$-hard to approximate. *J. ACM*, 52(4):538–551, 2005.

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):pp. 233–235, 1979.

[CKT10] Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Max-cover in map-reduce. In *WWW*, pages 231–240, 2010.

[Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[FLT04] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[Gon85] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38(2-3):293–306, 1985.

[HS85] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[JMM+03] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.

[Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[Kar72] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KVY94] Samir Khuller, Uzi Vishkin, and Neal E. Young. A primal-dual parallel approximation technique applied to weighted set and vertex covers. *J. Algorithms*, 17(2):280–289, 1994.

[KW85] Richard M. Karp and Avi Wigderson. A fast parallel algorithm for the maximal independent set problem. *Journal of the ACM*, 32(4):762–773, 1985.

[Lub86] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[PV98] Rina Panigrahy and Sundar Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *J. Algorithms*, 27(2):259–268, 1998.

[RR89] Sanguthevar Rajasekaran and John H. Reif. Optimal and sublogarithmic time randomized parallel sorting algorithms. *SIAM J. Comput.*, 18(3):594–607, 1989.

[RV98] Sridhar Rajagopalan and Vijay V. Vazirani. Primal-dual *RNC* approximation algorithms for set cover and covering integer programs. *SIAM J. Comput.*, 28(2):525–540, 1998.

[Vaz01] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001.

[You95] Neal E. Young. Randomized rounding without solving the linear program. In *SODA*, pages 170–178, 1995.