# Fast Distributed Approximation for Max-Cut

Keren Censor-Hillel(✉), Rina Levy, and Hadas Shachnai

Computer Science Department, Technion, 3200003 Haifa, Israel
{ckeren,rinalevy,hadas}@cs.technion.ac.il

**Abstract.** Finding a maximum cut is a fundamental task in many computational settings, with a central application in wireless networks. Surprisingly, Max-Cut has been insufficiently studied in the classic distributed settings, where vertices communicate by synchronously sending messages to their neighbors according to the underlying graph, known as the $\mathcal{LOCAL}$ or $\mathcal{CONGEST}$ models. We amend this by obtaining almost optimal algorithms for Max-Cut on a wide class of graphs in these models. In particular, for any $\epsilon > 0$, we develop randomized approximation algorithms achieving a ratio of $(1 - \varepsilon)$ to the optimum for Max-Cut on bipartite graphs in the $\mathcal{CONGEST}$ model, and on *general* graphs in the $\mathcal{LOCAL}$ model.

We further present efficient *deterministic* algorithms, including a 1/3-approximation for Max-Dicut in our models, thus improving the best known (randomized) ratio of 1/4. Our algorithms make non-trivial use of the greedy approach of Buchbinder et al. (SIAM Journal Computing 44:1384–1402, 2015) for maximizing an unconstrained (non-monotone) submodular function, which may be of independent interest.

## 1 Introduction

Max-Cut is one of the fundamental problems in theoretical computer science. A *cut* in an undirected graph is a bipartition of the vertices, whose size is the number of edges crossing the bipartition. Finding cuts of maximum size in a given graph is among Karp's famous 21 NP-complete problems [25]. Since then, Max-Cut has received considerable attention, in approximation algorithms [17,19,42,45], parallel computation [44], parameterized complexity [43], and streaming algorithms (see, e.g., [24]).

Max-Cut has a central application in *wireless mesh networks (WMNs)*. The capacity of WMNs that operate over a single frequency can be increased significantly by enhancing each router with multiple *transmit (Tx)* or *receive (Rx)* (*MTR*) capability. Thus, a node will not experience collision when two or more neighbors transmit to it. Yet, interference occurs if a node transmits and receives simultaneously. This is known as the *no mix-tx-rx* constraint. The set of links activated in each time slot, defining the capacity of an MTR WMN, is governed

K. Censor-Hillel—The research is supported in part by the Israel Science Foundation (grant 1696/14).

by a link scheduler. As shown in [10], link scheduling is equivalent to finding Max-Cut in each time slot. A maximum cut contains the set of non-conflicting links that can be activated at the same time, i.e., they adhere to the no mix-tx-rx constraint. The induced bipartition of the vertices at each time slot defines a set of transmitters and a set of receivers in this slot. Link scheduling algorithms based on approximating Max-Cut, and other applications in wireless networks, can be found in [27,48–51].[1]

Surprisingly, Max-Cut has been insufficiently studied in the classic distributed settings, where vertices communicate by synchronously sending messages to their neighbors according to the underlying graph, known as the $\mathcal{LOCAL}$ or $\mathcal{CONGEST}$ models. Indeed, there are known distributed algorithms for Max-Cut using MapReduce techniques [5,35,36]. In this setting, the algorithms partition the ground set among $m$ machines and obtain a solution using all the outputs. However, despite a seemingly similar title, our distributed setting is completely different.

In this paper we address Max-Cut in the classic distributed network models, where the graph represents a synchronous communication network. At the end of the computation, each vertex decides locally whether it joins the subset $S$ or $\bar{S}$, and outputs 1 or 0, respectively, so as to obtain a cut of largest possible size.

It is well known that choosing a random cut, i.e., assigning each vertex to $S$ or $\bar{S}$ with probability 1/2, yields a $\frac{1}{2}$-approximation for Max-Cut, and a $\frac{1}{4}$-approximation for Max-Dicut, defined on directed graphs (see, e.g., [37,38]).[2] Thus, a local algorithm, where each vertex outputs 0 or 1 with probability 1/2, yields the above approximation factors with no communication required. On the other hand, we note that a single vertex can find an optimal solution, once it has learned the underlying graph. However, this requires a number of communication rounds that depends *linearly* on global network parameters (depending on the exact model considered). This defines a tradeoff between time complexity and the approximation ratio obtained by distributed Max-Cut algorithms. The huge gap between the above results raises the following natural questions: How well can Max-Cut be approximated in the distributed setting, using a bounded number of communication rounds? Or, more precisely: How many communication rounds are required for obtaining an approximation ratio strictly larger than half, or even a *deterministic* $\frac{1}{2}$-approximation for Max-Cut?

To the best of our knowledge, these questions have been studied in our distributed network models only for a restricted graph class. Specifically, the paper [22] suggests a distributed algorithm for Max-Cut on $d$-regular triangle-free graphs, that requires a single communication round and provides a $(1/2 + 0.28125/\sqrt{d})$-approximation.

The key contribution of this paper is in developing two main techniques for approximating Max-Cut and Max-Dicut in distributed networks, with *any*

[1] Max-Cut naturally arises also in VLSI [9], statistical physics [4] and machine learning [47].

[2] In Max-Dicut we seek the maximum size edge-set crossing from $S$ to $\bar{S}$.

communication graph. Below we detail the challenges we face, and our methods for overcoming them.

## 1.1 The Challenge

In the $\mathcal{LOCAL}$ model, where message sizes and the local computation power are unlimited, every standard graph problem can be solved in $O(n)$ communication rounds. For Max-Cut it also holds that finding an optimal solution requires $\Omega(n)$ communication rounds. This lower bound follows from Linial's seminal lower bound [31, Theorem 2.2] for finding a 2-coloring of an even-sized cycle. In an even cycle, the maximum cut contains all edges. Therefore, finding a Max-Cut is equivalent to finding a 2-coloring of the graph.

An approach that proved successful in many computational settings – in tackling hard problems – is to relax the optimality requirement and settle for approximate solutions. Indeed, in the distributed setting, many approximation algorithms have been devised to overcome the costs of finding exact solutions (see, e.g., [1–3,16,21,28–30,32,39], and the survey of Elkin [11]). Our work can be viewed as part of this general approach. However, we face crucial hurdles attempting to use the known sequential toolbox for approximating Max-Cut in the distributed setting.

As mentioned above, a $\frac{1}{2}$-approximation for Max-Cut can be obtained easily with no communication. While this holds in all of the above models, improving the ratio of 1/2 is much more complicated. In the sequential setting, an approximation factor strictly larger than 1/2 was obtained in the mid-1990's using semidefinite programming [17] (see Sect. 1.3). Almost two decades later, the technique was applied by [44] to obtain a parallel randomized algorithm for Max-Cut, achieving a ratio of $(1-\epsilon)0.878$ to the optimum, for any $\varepsilon > 0$. Adapting this algorithm to our distributed setting seems non-trivial, as it relies heavily on global computation. Trying to apply other techniques, such as local search, unfortunately leads to linear running time, because of the need to compare values of global solutions.

Another obstacle that lies ahead is the lack of *locality* in Max-Cut, due to strong dependence between the vertices. The existence of an edge in the cut depends on the assignment of both of its endpoints. This results in a chain of dependencies and raises the question whether cutting the chain can still guarantee a good approximation ratio.

## 1.2 Our Contribution

We develop two main techniques for approximating Max-Cut, as well as Max-Dicut. Our first technique relies on the crucial observation that the cut value is additive for edge-disjoint sets of vertices. Exploiting this property, we design *clustering-based* algorithms, in which we decompose the graph into small-diameter clusters, find an optimal solution within each cluster, and prove that the remaining edges still allow the final solution to meet the desired approximation ratio. An essential component in our algorithms is efficient graph decomposition to such small-diameter clusters connected by few edges (also known as a *padded partition*), inspired by a parallel algorithm of [34] (see also [12,13]).

For general graphs, this gives $(1-\epsilon)$-approximation algorithms for Max-Cut and Max-Dicut, requiring $O(\frac{\log n}{\epsilon})$ communication rounds in the $\mathcal{LOCAL}$ model. For the special case of a bipartite graph, we take advantage of the graph structure to obtain an improved clustering-based algorithm, which does not require large messages. The algorithm achieves a $(1-\epsilon)$-approximation for Max-Cut in $O(\frac{\log n}{\epsilon})$ rounds, in the more restricted $\mathcal{CONGEST}$ model.

For our second technique, we observe that the contribution of a specific vertex to the cut depends only on the vertex itself and its immediate neighbors. We leverage this fact to make multiple decisions in parallel by independent sets of vertices. We find such sets using distributed coloring algorithms. Our *coloring-based* technique, which makes non-trivial use of the greedy approach of [7] for maximizing an unconstrained submodular function, yields deterministic $\frac{1}{2}$-approximation and $\frac{1}{3}$-approximation algorithms for Max-Cut and Max-Dicut, respectively, and a randomized $\frac{1}{2}$-approximation algorithm for Max-Dicut. Each of these algorithms requires $\tilde{O}(\Delta + \log^* n)$ communication rounds in the $\mathcal{CONGEST}$ model, where $\Delta$ is the maximal degree of the graph, and $\tilde{O}$ ignores polylogarithmic factors in $\Delta$.

Finally, we present $\mathcal{LOCAL}$ algorithms which combine both of our techniques. Applying the coloring-based technique to low-degree vertices, and the clustering-based technique to high-degree vertices, allows as to design faster deterministic algorithms with approximation ratios of $\frac{1}{2}$ and $\frac{1}{3}$ for Max-Cut and Max-Dicut, respectively, requiring $\min\{\tilde{O}(\Delta + \log^* n), O(\sqrt{n})\}$ communication rounds (Table 1).[3]

**Table 1.** A summary of our results.

| Algorithm Properties | | | | Approximation Ratio | | |
|---|---|---|---|---|---|---|
| Rounds | Deterministic | Model | Graph | Max-Cut | Max-Dicut | |
| no communication | ✗ | $\mathcal{CONGEST}$ | any | 1/2 | 1/4 | folklore |
| $O(\log n/\epsilon)$ | ✗ | $\mathcal{CONGEST}$ | bipartite | $1-\epsilon$ | – | new |
| $O(\log n/\epsilon)$ | ✗ | $\mathcal{LOCAL}$ | any | $1-\epsilon$ | $1-\epsilon$ | new |
| $\tilde{O}(\Delta + \log^* n)$ | ✓ | $\mathcal{CONGEST}$ | any | 1/2 | 1/3 | new |
| $\tilde{O}(\Delta + \log^* n)$ | ✗ | $\mathcal{CONGEST}$ | any | 1/2 | 1/2 | new |
| $\min\{\tilde{O}(\Delta + \log^* n), O(\sqrt{n})\}$ | ✓ | $\mathcal{LOCAL}$ | any | 1/2 | 1/3 | new |

## 1.3 Background and Related Work

The weighted version of Max-Cut is one of Karp's NP-complete problems [25]. The unweighted version that we study here is also known to be NP-complete [15].

While there are graph families, such as planar and bipartite graphs, in which a maximum cut can be found in polynomial time [18,19], in general graphs, even

[3] Due to space constraints, some of the results are omitted. A detailed version of this paper can be found in [8].

approximating the problem is NP-hard. In the sequential setting, one cannot obtain an approximation ratio better than $\frac{16}{17}$ for Max-Cut, or an approximation ratio better than $\frac{12}{13}$ for Max-Dicut, unless $P = NP$ [20,46].

Choosing a random cut, i.e., assigning each vertex to $S$ or $\bar{S}$ with probability 1/2, yields a $\frac{1}{2}$-approximation for Max-Cut, and $\frac{1}{4}$-approximation for Max-Dicut. In the sequential setting there are also deterministic algorithms yielding the above approximation ratios [40,42]. For 20 years there was no progress in improving the 1/2 approximation ratio for Max-Cut, until (in 1995) Goemans and Williamson [17] achieved the currently best known ratio, using semidefinite programming. They present a 0.878-approximation algorithm, which is optimal assuming the Unique Games Conjecture holds [26]. In the same paper, Goemans and Williamson also give a 0.796-approximation algorithm for Max-Dicut. This ratio was improved later by Matuura et al. [33], to 0.863. Using spectral techniques, a 0.53-approximation algorithm for Max-Cut was given by Trevisan [45]. In [23] Kale and Seshadhri present a combinatorial approximation algorithm for Max-Cut using random walks, which gives a $(0.5 + \delta)$-approximation, where $\delta$ is some positive constant that appears also in the running time of the algorithm. In particular, for $\tilde{O}(n^{1.6}), \tilde{O}(n^2)$ and $\tilde{O}(n^3)$ times, the algorithm achieves approximation factors of $0.5051, 0.5155$ and $0.5727$, respectively.

Max-Cut and Max-Dicut can also be viewed as special cases of submodular maximization, which has been widely studied. It is known that choosing a solution set $S$ uniformly at random yields a $\frac{1}{4}$-approximation, and a $\frac{1}{2}$-approximation for a general and for symmetric submodular function, respectively [14]. These results imply the known random approximation ratios for Max-Cut and Max-Dicut. Buchbinder et al. [7] present determinstic $\frac{1}{2}$-approximation algorithms for both symmetric and asymmetric submodular functions. These algorithms assume that the submodular function is accessible through a black box returning $f(S)$ for any given set $S$ (known as the *value oracle* model).

In recent years, there is an ongoing effort to develop distributed algorithms for submodular maximization problems, using MapReduce techniques [5,35,36]. Often, the inputs consist of large data sets, for which a sequential algorithm may be inefficient. The main idea behind these algorithms is to partition the ground set among $m$ machines, and have each machine solve the problem optimally independently of others. After all machines have completed their computations, they share their solutions. A final solution is obtained by solving the problem once again over a union of the partial solutions. The algorithms achieve performance guarantees close to the sequential algorithms while decreasing the running time, where the running time is the number of communication rounds among the machines. As mentioned above, these algorithms do not apply to our classic distributed settings.

## 2 Preliminaries

The Max-Cut problem is defined as follows. Given an undirected graph $G = (V, E)$, one needs to divide the vertices into two subsets, $S \subset V$ and $\bar{S} = V \setminus S$, such that the size of the cut, i.e., the number of edges between $S$ and the complementary subset $\bar{S}$, is as large as possible. In the Max-Dicut problem, the given graph $G = (V, E)$ is directed, and the cut is defined only as the edges which are directed from $S$ to $\bar{S}$. As in the Max-Cut problem, the goal is to obtain the largest cut.

Max-Cut and Max-Dicut can be described as the problem of maximizing the submodular function $f(S) = |E(S, \bar{S})|$, where for Max-Dicut $f(S)$ counts only the edges directed from $S$ to $\bar{S}$. Given a finite set $X$, let $2^X$ denote the power set of $X$. A function $f : 2^X \to \mathbb{R}$ is *submodular* if it satisfies the following equivalent conditions:

(i) For any $S, T \subseteq X$: $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$.
(ii) For any $A \subseteq B \subseteq X$ and $x \in X \setminus B$: $f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A)$.

For Max-Cut and Max-Dicut, the submodular function also satisfies: for any pair of disjoint sets $S, T \subseteq X$ such that $E_{S \times T} = \{(u, v) | u \in S, v \in T\} = \emptyset$, $f(S) + f(T) = f(S \cup T)$. Note that for Max-Cut, the function is also symmetric, i.e., $f(S) = f(\bar{S})$.

**Model:** We consider a distributed system, modeled by a graph $G = (V, E)$, in which the vertices represent the computational entities, and the edges represent the communication channels between them. We assume that each vertex $v$ has a unique identifier $id(v)$ of size $O(\log n)$, where $n = |V|$.

The communication between the entities is synchronous, i.e., the time is divided into rounds. In each round, the vertices send messages simultaneously to all of their neighbors and make a local computation based on the information gained so far. This is the classic $\mathcal{LOCAL}$ model [41], which focuses on analyzing how locality affects the distributed computation. Therefore, message sizes and local computation power are unlimited, and the complexity is measured by the number of communication rounds needed to obtain a solution. It is also important to study what can be done in the more restricted $\mathcal{CONGEST}$ model [41], in which message sizes are $O(\log n)$.

We assume that each vertex has preliminary information including the size of the network $n = |V|$, its neighbors, and the maximal degree of the graph $\Delta$.[4]

Each vertex runs a local algorithm to solve the Max-Cut problem. Along the algorithm, each vertex decides locally whether it joins $S$ or $\bar{S}$, and outputs 1 or 0, respectively. We define the *solution* of the algorithm as the set of all outputs. Note that individual vertices do not hold the entire solution, but only their local information. The solution *value* is defined as the size of the cut induced by the solution. We show that this value approximates the size of the maximum cut.

## 3 Clustering-Based Algorithms

In this section we present clustering-based algorithms for Max-Cut and Max-Dicut. Our technique relies on the observation that Max-Cut is a collection of

[4] This assumption is needed only for the $(\Delta+1)$-coloring algorithm [6] used in Sect. 4; it can be omitted (see [6]), increasing the running time by a constant factor.

edges having their endpoints in different sets; therefore, it can be viewed as the union of cuts in the disjoint parts of the graph.

Given a graph $G = (V, E)$, we first eliminate a small fraction of edges to obtain small-diameter connected components. Then, the problem is solved optimally within each connected component. For general graphs, this is done by gathering the topology of the component at a single vertex. For the special case of a bipartite graph, we can use the graph structure to propagate less information. Since the final solution, consisting of the local decisions of all vertices, is at least as good as the sum of the optimal solutions in the components, and since the fraction of eliminated edges is small, we prove that the technique yields a $(1 - \epsilon)$-approximation.

### 3.1 A Randomized Distributed Graph Decomposition

We start by presenting the randomized distributed graph decomposition algorithm. The algorithm is inspired by a parallel graph decomposition by Miller et al. [34] that we adapt to the distributed.[5] The PRAM algorithm of [34] generates a *strong padded partition* of a given graph, namely, a partition into connected components with strong diameter $O(\frac{\log n}{\beta})$, for some $\beta \leq 1/2$, such that the fraction of edges that cross between different clusters of the partition is at most $\beta$. As we show below, the distributed version guarantees the same properties with high probability and requires only $O(\frac{\log n}{\beta})$ communication rounds in the $\mathcal{CONGEST}$ model.

The distributed version of the graph decomposition algorithm proceeds as follows: Let $\delta_v$ be a random value chosen by vertex $v$ from an exponential distribution with parameter $\beta$. Define the *shifted distance* from vertex $v$ to vertex $u$ as $dist_\delta(u, v) = dist(u, v) - \delta_u$. Along the algorithm, each vertex $v$ finds a vertex $u$ within its $\frac{k \log n}{\beta}$-neighborhood that minimizes $dist_\delta(u, v)$, where $k$ is a constant. We define this vertex as $v$'s *center*. This step implies the difference between the parallel and the distributed decomposition. Indeed, in the parallel algorithm, each vertex chooses its center from the *entire* ground set $V$. We show that our modified process still generates a decomposition with the desired properties. Furthermore, w.h.p. the distributed algorithm outputs a decomposition identical to the one created by the parallel algorithm. A pseudocode of the algorithm is given in Algorithm 1.

We prove that the fraction of edges between different components is small. In order to do so, we bound the probability of an edge to be between components, i.e., the probability that the endpoints of the edge choose different centers. We consider two cases for an edge $e = (u, v)$. In the first case, we assume that both $u$ and $v$ choose the center that minimizes their shifted distance, $dist_\delta$, over all the vertices in the graph. In other words, if the algorithm allowed each vertex to learn the entire graph, they would choose the same center as they did in our

[5] Our algorithm can be viewed as one phase of the distributed algorithm presented by Elkin et al. in [12] with some necessary changes.

**Algorithm 1.** Distributed Decomposition, *code for vertex v*

```
1: $0 < \beta < 1, k > 2$.
2: choose $\delta_v$ at random from $Exp(\beta)$
3: $center = id(v)$
4: $dist_{\delta_{min}} = -\delta_v$
5: for $\frac{k \log n}{\beta}$ iterations do
6:     send $(dist_{\delta_{min}}, center)$
7:     for every $(dist_{\delta_{min}}^{'}, center^{'})$ received from $u \in N(v)$ do
8:         if $\left(dist_{\delta_{min}}^{'} + 1 < dist_{\delta_{min}}\right)$ OR $\left((dist_{\delta_{min}}^{'} + 1 = dist_{\delta_{min}})\right.$ AND $\left.(center^{'} < center)\right)$ then
9:             $center \leftarrow center^{'}$
10:            $dist_{\delta_{min}} \leftarrow dist_{\delta_{min}}^{'} + 1$
11:        end if
12:    end for
13: end for
14: output $center$
```

algorithm. In the second case, we assume that at least one of $u$ and $v$ chooses differently if given a larger neighborhood.

Define the *ideal* center of a vertex $v$ as $argmin_{w \in V} dist_\delta(w, v)$. In the next lemma, we upper bound the probability that a vertex does not choose its ideal center.

**Lemma 3.1.** *Let $v'$ be the ideal center of vertex $v$, then the probability that $dist(v', v) > \frac{k \log n}{\beta}$, i.e., vertex $v$ does not join its ideal center, is at most $\frac{1}{n^k}$.*

*Proof.* Since $v'$ is the ideal center of vertex $v$, we have that $dist_\delta(v', v) \leq dist_\delta(v, v)$. Therefore, $dist(v', v) - \delta_{v'} \leq dist(v, v) - \delta_v = -\delta_v \leq 0$, which implies that $dist(v', v) \leq \delta_{v'}$. That is, the distance between each vertex $v$ to its ideal center $v'$ is upper bounded by $\delta_{v'}$, and hence $\Pr\left[dist(v', v) > \frac{k \log n}{\beta}\right] \leq \Pr\left[\delta_{v'} > \frac{k \log n}{\beta}\right]$. Using the cumulative exponential distribution, we have that $\Pr\left[\delta_{v'} > \frac{k \log n}{\beta}\right] = \exp\left(-\frac{k \cdot \beta \log n}{\beta}\right) = \exp\left(-k \log n\right) \leq \frac{1}{n^k}$. □

**Corollary 3.2.** *The Distributed Decomposition algorithm generates a decomposition identical to the decomposition generated by the parallel decomposition algorithm with probability at least $1 - \frac{1}{n^{k-1}}$*

Define an *exterior* edge as an edge connecting different vertex components, and let $F$ denote the set of exterior edges. Let $A_{u,v}$ denote the event that both $u$ and $v$ choose their ideal centers.

**Lemma 3.3.** *The probability that an edge $e = (u, v)$ is an exterior edge, given that $u$ and $v$ choose their ideal centers, is at most $\beta$.*

The lemma follows directly from [34], where indeed the algorithm assigns to each vertex its ideal center. We can now bound the probability of any edge to be an exterior edge.

**Lemma 3.4.** *The probability that an edge $e = (u, v)$ is in $F$ is at most $\beta + \frac{2}{n^k}$.*

We can now prove the performance guarantees of the Distributed Decomposition algorithm. Recall that the *weak diameter* of a set $S = \{u_1, u_2, ...u_l\}$ is defined as $\max_{(u_i,u_j)\in S} dist(u_i, u_j)$.

**Theorem 3.5.** *The Distributed Decomposition algorithm requires $O(\frac{\log n}{\beta})$ communication rounds in the $\mathcal{CONGEST}$ model, and partitions the graph into components, such that in expectation there are $O(\beta m)$ exterior edges. Each of the components is of weak diameter $O(\frac{\log n}{\beta})$, and with high probability also of strong diameter $O(\frac{\log n}{\beta})$.*

*Proof.* Clearly, as each vertex chooses a center from its $\frac{k \log n}{\beta}$-neighborhood, the distance between two vertices that choose the same center, i.e., belong to the same component, over the graph $G$, is at most $O(\frac{\log n}{\beta})$. Therefore, the weak diameter of every component is at most $O(\frac{\log n}{\beta})$. By Corollary 3.2, with probability at least $1 - \frac{1}{n^{k-1}}$, the algorithm outputs a partition identical to the one output by the parallel algorithm, and therefore with the same properties, which implies that the strong diameter of every component is at most $O(\frac{\log n}{\beta})$ as well.

Using the linearity of expectation and Lemma 3.4, we have that $\mathbb{E}[|F|] \leq \sum_{e\in E} \left(\beta + \frac{2}{n^k}\right) = \beta m + \frac{2m}{n^k}$. Since $m \leq n^2$, for any $k > 2$, $\mathbb{E}[|F|] \leq O(\beta m)$. Finally, as can be seen from the code, the algorithm requires $O(\frac{\log n}{\beta})$ communication rounds. □

## 3.2 A Randomized $(1 - \epsilon)$-Approximation Algorithm for Max-Cut on a Bipartite Graph

Clearly, in a bipartite graph, the maximum cut contains all of the edges. Such a cut can be found by selecting arbitrarily a root vertex, and then simply putting all the vertices of odd depth in one set and all the vertices of even depth in the complementary set. However, this would require a large computational time in our model, that depends on the diameter of the graph. We overcome this by using the above decomposition, and finding an optimal solution within each connected component. In each component $C$, we find an optimal solution in $O(D_c)$ communication rounds, where $D_c$ is the diameter of $C$. First, the vertices in each component search for the vertex with the lowest id.[6] Then, the vertex with the lowest id joins $S$ or $\bar{S}$ with equal probability and sends its decision to its neighbors. When a vertex receives a message from one of its neighbors, it joins the opposite set, outputs its decision, and sends it to its neighbors. Since finding the optimal solution within each component does not require learning the entire component topology, the algorithm is applicable in the more restricted

[6] This can be done by running a BFS in parallel from all vertices. Each vertex propagates the information from the root with lowest id it knows so far, and joins its tree. Thus, at the end of the process, we have a BFS tree rooted at the vertex with the lowest id.

$\mathcal{CONGEST}$ model. The algorithm yields a $(1-\epsilon)$-approximation for the Max-Cut problem on a bipartite graph in $O(\frac{\log n}{\epsilon})$ communication rounds with high probability.

**Theorem 3.6.** *Bipartite Max-Cut is a randomized* $(1-\epsilon)$*-approximation for Max-Cut, requiring* $O(\frac{\log n}{\epsilon})$ *communication rounds in the* $\mathcal{CONGEST}$ *model w.h.p.*

**Algorithm 2.** Bipartite Max-Cut

```
1: G=(V,E)
2: apply Distributed Decomposition to G, with $\beta = \epsilon, k > 2$
3: for each component $C$ obtained by the decomposition do
4:     build a BFS tree from the vertex $v$ with the lowest id
5:     assign $v$ to $S$ or $\bar{S}$ with equal probability, assign the rest of the vertices to alternating sides
6: end for
```

## 3.3 A Randomized $(1-\epsilon)$-Approximation Algorithm for General Graphs

We present below a $(1-\epsilon)$-approximation algorithm for Max-Cut in general graphs, using $O(\frac{\log n}{\epsilon})$ communication rounds. As before, the algorithm consists of a decomposition phase and a solution phase. While the decomposition phase works in the $\mathcal{CONGEST}$ model, the algorithm suits for the $\mathcal{LOCAL}$ model, since for general graphs, the generated components are not necessarily sparse, and learning the components topology is expensive in the $\mathcal{CONGEST}$ model.

**Algorithm 3.** Decomposition-Based Max-Cut

```
1: G=(V,E)
2: apply Distributed Decomposition on G, with $\beta = \epsilon/2, k > 2$
3: for each component $C$ obtained by the decomposition do
4:     gather the component topology at the vertex $v \in C$ with the lowest id.
5:     let $v$ find an optimal solution and determine the value output by the component's vertices.
6: end for
```

**Theorem 3.7.** *Decomposition-Based Max-Cut is a randomized* $(1-\epsilon)$*-approximation for Max-Cut, requiring* $O(\frac{\log n}{\epsilon})$ *communication rounds in the* $\mathcal{LOCAL}$ *model.*

*Proof.* Let $OPT(G)$ be the set of edges that belong to some maximum cut in $G$, and let $ALG(G)$ be the set of edges in the cut obtained by Decomposition-Based Max-Cut. Let $S_u$ be the component induced by the vertices which choose $u$ as their center, and denote by $S$ the set of components that algorithm Distributed Decomposition constructs. Then $\mathbb{E}\left[|ALG(G)|\right] \geq \mathbb{E}\left[\sum_{S_u \in S} |OPT(S_u)|\right] \geq |OPT(G)| - \beta m \geq |OPT(G)| - 2\beta|OPT(G)| = (1-\epsilon)|OPT(G)|$. The last inequality follows from the fact that for every graph $G$ it holds that $|OPT(G)| \geq \frac{m}{2}$.

The graph decomposition requires $O(\frac{\log n}{\epsilon})$ communication rounds, and outputs components with weak diameter at most $O(\frac{\log n}{\epsilon})$. Therefore, finding the optimal solution within each component takes $O(\frac{\log n}{\epsilon})$ as well. The time bound follows. $\square$

By taking $\beta = \epsilon/4$, one can now obtain a $(1-\epsilon)$-approximation algorithm for Max-Dicut. The difference comes from the fact that for Max-Dicut it holds that $|OPT(G)| \geq \frac{m}{4}$ for every graph $G$. The rest of the analysis is similar to the analysis for Max-Cut. Hence, we have

**Theorem 3.8.** *Decomposition-Based Max-Dicut is a randomized $(1-\epsilon)$-approximation for Max-Dicut, requiring $O(\frac{\log n}{\epsilon})$ communication rounds in the $\mathcal{LOCAL}$ model.*

## 4 Coloring-Based Algorithms

Many of the sequential approximation algorithms for Max-Cut perform $n$ iterations. Each vertex, in its turn, makes a greedy decision so as to maximize the solution value. We present distributed greedy algorithms that achieve the approximation ratios of the sequential algorithms much faster. We first prove that the greedy decisions of vertices can be done locally, depending only on their immediate neighbors. Then we show how to parallelize the decision process, such that in each iteration an independent set of vertices completes. The independent sets are generated using $(\Delta+1)$-coloring; then, for $(\Delta+1)$ iterations, all vertices of the relevant color make their parallel independent decisions. All algorithms run in the $\mathcal{CONGEST}$ model (see [8]).

## 5 A Deterministic $\mathcal{LOCAL}$ Algorithm

Our coloring-based algorithms may become inefficient for high degree graphs, due to the strong dependence on $\Delta$. Consider a clique in this model. The above algorithms require a linear number of communication rounds, while learning the entire graph and finding an optimal solution requires only $O(1)$ communication rounds in the $\mathcal{LOCAL}$ model. Indeed, there is a tradeoff between the graph diameter and the average degree of its vertices. Based on this tradeoff, we propose a faster, two-step, deterministic algorithm for Max-Cut that requires $min\{\tilde{O}(\Delta + \log^* n), O(\sqrt{n})\}$ communication rounds in the $\mathcal{LOCAL}$ model. The pseudocode is given in Algorithm 4.

We call a vertex $v$ a *low-degree* vertex, if $deg(v) < \sqrt{n}$, and a *high-degree* vertex, if $deg(v) \geq \sqrt{n}$. Define $G_{low}$, and $G_{high}$ as the graphs induced by the low-degree vertices and the high-degree vertices, respectively. The idea is to solve the problem separately for $G_{low}$ and for $G_{high}$.

In the first step, the algorithm deletes every high-degree vertex, if there are any, and its adjacent edges, creating $G_{low}$. The deletion means that the low-degree vertices ignore the edges that connect them to high-degree vertices and do not communicate over them. Then, the algorithm approximates the Max-Cut on $G_{low}$, using one of the coloring-based algorithms described in Sect. 4.

In the second step, the problem is solved optimally within each connected component in $G_{high}$. However, the high-degree vertices are allowed to communicate over edges which are not in $G_{high}$. As we prove next, the distance in the original graph $G$ between any two vertices which are connected in $G_{high}$ is upper bounded by $O(\sqrt{n})$. Hence, the number of rounds needed for this part of the algorithm is $O(\sqrt{n})$.

**Algorithm 4.** Fast Distributed Greedy Max-Cut

```
1: run Distributed Greedy Max-Cut on $G_{low}$
2: **for** each connected component in $G_{high}$ **do**
3:     learn the component topology in $G$, including all its adjacent edges
4:     let the vertex with the lowest id find an optimal solution, and determine the
       output for each vertex in its component
5: **end for**
6: output the vertices decisions
```

**Lemma 5.1.** *Assume $u, v$ are connected in $G_{high}$, then the distance between $u$ and $v$ in the original graph $G$ is at most $3\sqrt{n}$.*

**Theorem 5.2.** *Fast Distributed Greedy Max-Cut yields a $\frac{1}{2}$-approximation to Max-Cut, using $min\{\tilde{O}(\Delta + \log^* n), O(\sqrt{n})\}$ communication rounds in the $\mathcal{LOCAL}$ model.*

*Proof.* We first prove the approximation ratio. Since Distributed Greedy Max-Cut is applied on $G_{low}$, at least half of the edges of $G_{low}$ are in the cut. Given the decisions of vertices in $G_{low}$, the algorithm finds an optimal solution for all vertices in $G_{high}$. Note that running Distributed Greedy Max-Cut on the high-degree vertices of $G$, would give at least half of the remaining edges. This is due to the fact that the algorithm makes sequential greedy decisions. Therefore, an optimal solution for the high-degree vertices guarantees at least half of the edges in $G \setminus G_{low}$, implying the approximation ratio.

Applying Distributed Greedy Max-Cut on $G_{low}$ requires $\tilde{O}(\Delta_{low} + \log^* n)$ communication rounds, where $\Delta_{low} = min\{\Delta, \sqrt{n}\}$. Using Lemma 5.1 we have that each high degree vertex can communicate with every high-degree vertex connected to it in $G_{high}$, using at most $O(\sqrt{n})$ communication rounds. Hence, Steps 2. – 4. of the algorithm take $O(\sqrt{n})$ communication rounds. We note that when $\Delta < \sqrt{n}$, the algorithm terminates after the first step. Thus, the algorithm requires $min\{\tilde{O}(\Delta + \log^* n), O(\sqrt{n})\}$ communication rounds. □

Using the above technique, we obtain a fast, deterministic algorithm for the Max-Dicut problem, by replacing the call to Distributed Greedy Max-Cut in Step 1 with a call to Distributed Greedy Max-Dicut. Using the same arguments as in the analysis for the Max-Cut algorithm, we have:

**Theorem 5.3.** *Fast Distributed Greedy Max-Dicut yields a $\frac{1}{3}$-approximation to Max-Dicut, using $min\{\tilde{O}(\Delta + \log^* n), O(\sqrt{n})\}$ communication rounds in the $\mathcal{LOCAL}$ model.*

**Acknowledgements.** We thank Roy Schwartz and Shay Kutten for stimulating discussions and for helpful comments on the paper.

## References

1. Åstrand, M., Floréen, P., Polishchuk, V., Rybicki, J., Suomela, J., Uitto, J.: A local 2-approximation algorithm for the vertex cover problem. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 191–205. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04355-0_21
2. Åstrand, M., Suomela, J.: Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In: Proceedings of the Twenty-Second Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 294–302. ACM (2010)
3. Bar-Yehuda, R., Censor-Hillel, K., Schwartzman, G.: A distributed $(2+\epsilon)$-approximation for vertex cover in $O(\log\Delta/\epsilon \log\log \Delta)$ rounds. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, 25–28 July 2016, pp. 3–8 (2016)
4. Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An application of combinatorial optimization to statistical physics and circuit layout design. Oper. Res. **36**(3), 493–513 (1988)
5. da Ponte Barbosa, R., Ene, A., Nguyen, H.L., Ward, J.: A new framework for distributed submodular maximization. arXiv preprint http://arxiv.org/abs/1507.03719 (2015)
6. Barenboim, L.: Deterministic $(\delta+ 1)$-coloring in sublinear (in $\delta$) time in static, dynamic and faulty networks. In: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, pp. 345–354. ACM (2015)
7. Buchbinder, N., Feldman, M., Naor, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. SIAM J. Comput. **44**(5), 1384–1402 (2015)
8. Censor-Hillel, K., Levy, R., Shachnai, H.: Fast distributed approximation for max-cut. arXiv preprint http://arxiv.org/abs/1707.08496 (2017)
9. Chang, K., Du, D.C.: Efficient algorithms for layer assignment problem. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **6**(1), 67–78 (1987)
10. Chin, K.W., Soh, S., Meng, C.: Novel scheduling algorithms for concurrent transmit/receive wireless mesh networks. Comput. Netw. **56**(4), 1200–1214 (2012)

11. Elkin, M.: Distributed approximation: a survey. ACM SIGACT News **35**(4), 40–57 (2004)
12. Elkin, M., Neiman, O.: Distributed strong diameter network decomposition. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, pp. 211–216. ACM (2016)
13. Elkin, M., Neiman, O.: Efficient algorithms for constructing very sparse spanners and emulators. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, 16–19 January, pp. 652–669 (2017)
14. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. SIAM J. Comput. **40**(4), 1133–1153 (2011)
15. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theoret. Comput. Sci. **1**(3), 237–267 (1976)
16. Ghaffari, M., Kuhn, F.: Distributed minimum cut approximation. In: Afek, Y. (ed.) DISC 2013. LNCS, vol. 8205, pp. 1–15. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41527-2_1
17. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
18. Grötschel, M., Pulleyblank, W.R.: Weakly bipartite graphs and the max-cut problem. Oper. Res. Lett. **1**(1), 23–27 (1981)
19. Hadlock, F.: Finding a maximum cut of a planar graph in polynomial time. SIAM J. Comput. **4**(3), 221–225 (1975)
20. Håstad, J.: Some optimal inapproximability results. J. ACM (JACM) **48**(4), 798–859 (2001)
21. Henzinger, M., Krinninger, S., Nanongkai, D.: A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In: Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, pp. 489–498. ACM (2016)
22. Hirvonen, J., Rybicki, J., Schmid, S., Suomela, J.: Large cuts with local algorithms on triangle-free graphs. arXiv preprint arXiv:1402.2543 (2014)
23. Kale, S., Seshadhri, C.: Combinatorial approximation algorithms for maxcut using random walks. arXiv preprint arXiv:1008.3938 (2010)
24. Kapralov, M., Khanna, S., Sudan, M.: Streaming lower bounds for approximating max-cut. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1263–1282. SIAM (2015)
25. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations, pp. 85–103. Springer, Heidelberg (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
26. Khot, S., Kindler, G., Mossel, E., O'Donnell, R.: Optimal inapproximability results for max-cut and other 2-variable CSPs? SIAM J. Comput. **37**(1), 319–357 (2007)
27. Komurlu, C., Bilgic, M.: Active inference and dynamic Gaussian Bayesian networks for battery optimization in wireless sensor networks. In: AI for Smart Grids and Smart Buildings, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA (2016)
28. Kuhn, F., Moscibroda, T.: Distributed approximation of capacitated dominating sets. Theory Comput. Syst. **47**(4), 811–836 (2010)
29. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local computation: lower and upper bounds. J. ACM (JACM) **63**(2), 17 (2016)

30. Lenzen, C., Pignolet, Y.A., Wattenhofer, R.: Distributed minimum dominating set approximations in restricted families of graphs. Distrib. Comput. **26**(2), 119–137 (2013)
31. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992)
32. Lotker, Z., Patt-Shamir, B., Pettie, S.: Improved distributed approximate matching. In: Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures, pp. 129–136. ACM (2008)
33. Matuura, S., Matsui, T.: 0.863-approximation algorithm for MAX DICUT. In: Goemans, M., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX/RANDOM -2001. LNCS, vol. 2129, pp. 138–146. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44666-4_17
34. Miller, G.L., Peng, R., Xu, S.C.: Parallel graph decompositions using random shifts. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 196–203. ACM (2013)
35. Mirrokni, V., Zadimoghaddam, M.: Randomized composable core-sets for distributed submodular maximization. In: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, pp. 153–162. ACM (2015)
36. Mirzasoleiman, B., Karbasi, A., Sarkar, R., Krause, A.: Distributed submodular maximization: identifying representative elements in massive data. In: Advances in Neural Information Processing Systems, pp. 2049–2057 (2013)
37. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, Cambridge (2005)
38. Motwani, R., Raghavan, P.: Randomized Algorithms. Chapman & Hall/CRC, London (2010)
39. Nanongkai, D.: Distributed approximation algorithms for weighted shortest paths. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pp. 565–573. ACM (2014)
40. Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 229–234. ACM (1988)
41. Peleg, D.: Distributed Computing. SIAM Monographs on Discrete Mathematics and Applications, vol. 5 (2000)
42. Sahni, S., Gonzalez, T.: P-complete approximation problems. J. ACM (JACM) **23**(3), 555–565 (1976)
43. Saurabh, S., Zehavi, M.: $(k, n-k)$-Max-Cut: an $\mathcal{O}^*(2^p)$-time algorithm and a polynomial kernel. In: Kranakis, E., Navarro, G., Chávez, E. (eds.) LATIN 2016. LNCS, vol. 9644, pp. 686–699. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49529-2_51
44. Tangwongsan, K.: Efficient parallel approximation algorithms. Ph.D. thesis, School of Computer Science, Carnegie Mellon University (2011)
45. Trevisan, L.: Max cut and the smallest eigenvalue. SIAM J. Comput. **41**(6), 1769–1786 (2012)
46. Trevisan, L., Sorkin, G.B., Sudan, M., Williamson, D.P.: Gadgets, approximation, and linear programming. SIAM J. Comput. **29**(6), 2074–2097 (2000)
47. Wang, J., Jebara, T., Chang, S.F.: Semi-supervised learning using greedy max-cut. J. Mach. Learn. Res. **14**(Mar), 771–800 (2013)
48. Wang, L., Chin, K., Soh, S.: Joint routing and scheduling in multi-Tx/Rx wireless mesh networks with random demands. Comput. Netw. **98**, 44–56 (2016)

49. Wang, W., Liu, B., Yang, M., Luo, J., Shen, X.: Max-cut based overlapping channel assignment for 802.11 multi-radio wireless mesh networks. In: 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 662–667 (2013)
50. Xu, Y., Chin, K., Raad, R., Soh, S.: A novel distributed max-weight link scheduler for multi-transmit/receive wireless mesh networks. IEEE Trans. Veh. Technol. **65**(11), 9345–9357 (2016)
51. Xue, G., He, Q., Zhu, H., He, T., Liu, Y.: Sociality-aware access point selection in enterprise wireless LANs. IEEE Trans. Parallel Distrib. Syst. **24**(10), 2069–2078 (2013)