Adapting Local Sequential Algorithms to the Distributed Setting

Ken-ichi Kawarabayashi
National Institute of Informatics, Tokyo, Japan
k_keniti@nii.ac.jp

Gregory Schwartzman
National Institute of Informatics, Tokyo, Japan
greg@nii.ac.jp

Abstract

It is a well known fact that sequential algorithms which exhibit a strong "local" nature can be adapted to the distributed setting given a legal graph coloring. The running time of the distributed algorithm will then be at least the number of colors. Surprisingly, this well known idea was never formally stated as a unified framework. In this paper we aim to define a robust family of local sequential algorithms which can be easily adapted to the distributed setting. We then develop new tools to further enhance these algorithms, achieving state of the art results for fundamental problems.

We define a simple class of greedy-like algorithms which we call *orderless-local* algorithms. We show that given a legal c-coloring of the graph, every algorithm in this family can be converted into a distributed algorithm running in $O(c)$ communication rounds in the CONGEST model. We show that this family is indeed robust as both the method of conditional expectations and the unconstrained submodular maximization algorithm of Buchbinder *et al.* [10] can be expressed as orderless-local algorithms for *local utility functions* – Utility functions which have a strong local nature to them.

We use the above algorithms as a base for new distributed approximation algorithms for the weighted variants of some fundamental problems: Max k-Cut, Max-DiCut, Max 2-SAT and correlation clustering. We develop algorithms which have the same approximation guarantees as their sequential counterparts, up to a constant additive ϵ factor, while achieving an $O(\log^* n)$ running time for deterministic algorithms and $O(\epsilon^{-1})$ running time for randomized ones. This improves exponentially upon the currently best known algorithms.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed, Approximation Algorithms, Derandomization, Max-Cut

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.35

Related Version A full version of the paper is available at `https://arxiv.org/abs/1711.10155`.

Funding This work was supported by JST ERATO Grant Number JPMJER1201, Japan

Acknowledgements We would like to thank Ami Paz and Seri Khoury for many fruitful discussions and useful advice.

Table 1 Summary of our results for the CONGEST model ($\tilde{O}$ hides factors polylogarithmic in Δ). (*) General graphs, max-agree (†) Unweighted graphs, only Max-Cut ($k = 2$).

Problem	Our Approx.	Our Time	Prev Approx.	Prev Time	Notes
Weighted Correlation-Clustering*	$1/2 - \epsilon$	$O(\log^* n)$	-	-	det.
Weighted Max k-Cut	$1 - 1/k - \epsilon$	$O(\log^* n)$	$1/2$ [12]†	$\tilde{O}(\Delta + \log^* n)$	det.
Weighted Max-Dicut	$1/3 - \epsilon$	$O(\log^* n)$	$1/3$ [12]†	$\tilde{O}(\Delta + \log^* n)$	det.
Weighted Max-Dicut	$1/2 - \epsilon$	$O(\epsilon^{-1})$	$1/2$ [12]†	$\tilde{O}(\Delta + \log^* n)$	rand.
Weighted Max 2-SAT	$3/4 - \epsilon$	$O(\epsilon^{-1})$	-	-	rand.

1 Introduction

A large part of research in the distributed environment aims to develop fast distributed algorithms for problems which have already been studied in the sequential setting. Ideally, we would like to use the power of the distributed environment to achieve a substantial improvement in the running time over the sequential algorithm, and indeed, for many problems distributed algorithms achieve an exponential improvement over the sequential case. One approach to designing distributed algorithms is using the sequential algorithm as natural staring point [5–7, 12, 18], then certain adjustments are made for the distributed environment in order to achieve a faster running time.

There is a well known folklore in distributed computing, which roughly says that if a sequential graph algorithm works by traversing nodes in any order (perhaps adversarial), and for every node makes a local decision, then given a legal c-coloring of the graph, the algorithm can be adapted to the distributed setting by going over all color classes, and for each executing all nodes in the class simultaneously. Surprisingly, there is no formal framework describing the above. In this paper we provide such a framework for a specific class of algorithms (defined later).

We note that for general graphs a legal coloring may require at least $\Delta + 1$ colors, where Δ is the maximal degree of the graph. Using the above framework we aim to answer the following question: Are there certain classes of algorithms where using the above can result in a running time sublinear in Δ? We show that for certain approximation problems the answer is quite surprising, as we are able to achieve an almost constant running time!

More precisely, we show that for the problems of Max k-Cut, Max-DiCut, Max 2-SAT and correlation clustering we can adapt the sequential algorithm to these problems in such a way that the running time is $O(\log^* n)$ rounds for deterministic algorithms and $O(\epsilon^2)$ for randomized ones, while losing only an additive ϵ-factor in the approximation ratio. For the problems of Max-Cut and Max-DiCut this greatly improves upon the previous best known results, which required a number of rounds linear in Δ. A summary of our results appears in Table 1.

1.1 Tools and results

In this paper we focus our attention on approximation algorithms for unconstrained optimization problems on graphs. We are given some graph $G(V, E)$, where each vertex v is assigned a variable X_v taking values in some set A. We aim to maximize some utility function f over these variables (For a formal definition see Section 2). Our distributed model is the CONGEST model of distributed computation, where the network is represented by a graph, s.t nodes are computational units and edges are communication links. Nodes communicate in synchronous communication rounds, where at each round a node sends and receives messages from all of its neighbors. In the CONGEST model the size of messages sent between nodes is limited to $O(\log n)$ bits, where $|V| = n$. This is more restrictive than the LOCAL model, where message size is unbounded. Our complexity measure is the number of communication rounds of the algorithm.

Adapting a sequential algorithm of the type we describe above to the distributed setting, means we wish each node v in the communication graph to output an assignment to X_v such that the approximation guarantee is close to that of the sequential algorithm, while minimizing the number of communication rounds of the distributed algorithm. Our goal is to formally define a family of sequential algorithms which can be easily converted to distributed algorithms, and then develop tools to allow these algorithms to run exponentially faster, while achieving almost the same approximation ratio. To achieve this we focus our attention on a family of sequential algorithms which exhibit a very strong local nature.

We define a family of utility functions, which we call *local utility functions* (Formally defined in Section 2). We say that a utility function f is a local utility function, if the change to the value of the function upon setting one variable X_v can be computed locally. Intuitively, while optimizing a general utility function in the distributed setting might be difficult for global functions, the local nature of the family of local utility functions makes it a perfect candidate.

We focus on adapting a large family of, potentially randomized, local algorithms to the distributed setting. We consider *orderless-local* algorithms - algorithms that can traverse the variables in any order and in each iteration apply some local function to decide the value of the variable. By local we mean that the decision only depends on the local environment of the node in the graph, the variables of nodes adjacent to that variable and some randomness only used by that node. This is similar to the family of Priority algorithms first defined in [9]. The goal of [9] was to formally define the notion of a greedy algorithm, and then to explore the limits of these algorithms. Our definition is similar (and can be expressed as a special case of priority algorithms), but the goal is different. While [9] aims to prove lower bounds, we provide some sufficient conditions that allow us to easily transform local sequential algorithms into fast distributed algorithms.

Our definitions are also similar to the SLOCAL model [21], which also shows that sequential algorithms which traverse the graph vertices in any order and make local decisions can be adapted to the distributed LOCAL model in poly logarithmic rounds using randomization. While the results of [21] are much more broad, our transformation does not require any randomization and works in the CONGEST model. Finally, we should also mention the field of local computation algorithms [35] whose aim is developing efficient local *sequential* algorithms. We refer the reader to an excellent survey by Levi and Medina [30].

One might expect that due to the locality of this family of algorithms it can be distributed if the graph is provided with a legal coloring. The distributed algorithm goes over the color classes one after another and executes all nodes in the color class simultaneously. This solves any conflicts that may occur form executing two neighboring nodes, while the orderless property guarantees that this execution is valid. In a sense this argument was already used for specific algorithm (Coloring to MIS [32], MaxIS of [5], Max-Cut of [12]). We provide a more general result, using this classical argument. Specifically, we show that given a legal c-coloring, any orderless-local algorithm can be distributed in $O(c)$-communication rounds in the CONGEST model.

To show that this definition is indeed robust, we show two general applications. The first is adapting the method of conditional expectations (Formally defined in Section 2) to the distributed setting. This method is inherently sequential, but we show that if the utility function optimized is a local utility function, then the algorithm is an orderless-local algorithm. A classical application of this technique is for Max k-cut, where an $(1-1/k)$-approximation is achieved when every node chooses a cut side at random. This can be derandomized using the method of conditional expectations, and adapted to the distributed setting, as the cut function is a local utility function. We note that the same exact approach results in a $(1/2-\epsilon)$-approximation for max-agree correlation clustering on general graphs (see Section 2 for a definition). Because the tools used for Max-Cut directly translate to correlation clustering, we focus on Max-Cut for the rest of the paper, and only mention correlation clustering at the very end.

The second application is the unconstrained submodular maximization algorithms of [10], where a deterministic 1/3-approximation and a randomized expected 1/2-approximation algorithms are presented. We show that both are orderless-local algorithms when provided with a local utility function. This can be applied to the problem of Max-DiCut, as it is an unconstrained submodular function, and also a local utility function. The algorithms of [10] were already adapted to the distributed setting for the specific problem of Max-DiCut by [12] using similar ideas. The main benefit of our definition is the convenience and generality of adapting these algorithms without the need to consider their analysis or reprove correctness. We conclude that the family of orderless-local algorithms indeed contains robust algorithms for fundamental problems, and especially the method of conditional expectations.

At the time this paper was first made public, there was no distributed equivalent for the method of conditional expectations. We have since learned that, independently and simultaneously, an adaptation of the method of conditional expectations to the distributed setting was also presented in [20]. Their results show how the method of conditional expectations combined with a *legal* coloring can be used to convert any randomized LOCAL r-round algorithm for a locally checkable problem to a deterministic one, running in $O(\Delta^{O(r)}+O(r\log^* n))$.[1] This is done via a transformation to an SLOCAL algorithm, where the derandomization is applied and then transforming back to a LOCAL algorithm.

Although not stated for the CONGEST model, we believe it to be the case that when $r=1$ their application of the method of conditional expectations works in the CONGEST, and is equivalent to our results. Another difference apart from the different model of communication, is that they focus on derandomizing locally checkable problems, while we focus on local utility functions. These two families of problems are different, as the approximation guaranteed for a certain local utility function need not be locally checkable. This last point highlights the different goal of the two papers. While [20] skillfully show that a large family of LOCAL algorithm can be derandomized, we aim to adapt sequential algorithm to the distributed setting while achieving as *fast* of a running time as possible in the more restrictive CONGEST model – hence we focus on local utility function which capture the locality of the optimization process.

Next, we wish to consider the running time of these algorithms. Recall that we expressed the running time of orderless local algorithms in terms of the colors of some legal coloring for the graph. For a general graph, we cannot hope for a legal coloring using less than $\Delta+1$, where Δ is the maximum degree in the graph. This means that using the distributed version of an orderless-local algorithm unchanged will have a running time linear in Δ. We show how

[1] They actually show that the running time is either $O(\Delta^{O(r)}+O(r\log^* n))$ or $r\cdot 2^{O(\sqrt{logn})}$, achieving the latter via network decomposition. We focus on the first bound, as the second is less relevant for the comparison which follows.

to overcome this obstacle for Max k-Cut and Max-DiCut. The general idea is to compute a *defective* coloring of the graph which uses few colors, drop all monochromatic edges, and call the algorithm for the new graph which now has a legal coloring.

A key tool in our algorithms is a new type of defective coloring we call a *weighted ϵ-defective coloring*. The classical defective coloring allows each vertex to have at most d monochromatic edges, for some defect parameter d. We consider positively edge weighted graphs and require a weighted fractional defect - for every vertex the total weight of monochromatic edges is at most an ϵ-fraction of the total weight of all edges of that vertex. We show that a weighted ϵ-defective coloring using $O(\epsilon^{-2})$ colors can be computed *deterministically* in $O(\log^* n)$ rounds using the defective coloring algorithm of [29]. The classical algorithm of Kuhn was found useful in the adaptation of sequential algorithms to the distributed setting [17, 21], thus its effectiveness for weighted ϵ-defective coloring might be of further use.

Although we cannot guarantee a legal coloring with a small number of colors for any graph $G(V, E, w)$, we may remove some subset of E which will result in a new graph G' with a low chromatic number. We wish to do so while not decreasing the total sum of edge weights in G', which we prove guarantees the approximation will only be mildly affected for our cut problems. Formally, we show that if we only decrease the total edge weight by an ϵ-fraction, we will incur an additive ϵ-loss in the approximation ratio of the cut algorithms for G. For the randomized algorithm this is easy, simply color each vertex randomly with a color in $[\lceil\epsilon^{-1}\rceil]$ and drop all monochromatic edges. For the deterministic case, we execute our weighted ϵ-defective coloring algorithm, and then remove all monochromatic edges. We then execute the relevant cut algorithm on the resulting graph G' which now has a legal coloring, using a small number of colors. The above results in extremely fast approximation algorithms for weighted Max k-Cut and weighted Max-DiCut, while having almost the same approximation ratio as their sequential counterpart.

Finally, our techniques can also be applied to the problem of weighted Max 2-SAT. To do so we may use the randomized expected 3/4-approximation algorithm presented in [34]. It is based on the algorithm of [10], and thus is almost identical to the unconstrained submodular maximization algorithm. Because the techniques we use are very similar to the above, we defer the entire proof to the full version of the paper.[2]

1.2 Previous research

Cut problems: An excellent overview of the Max-Cut and Max-DiCut problems appears in [12], which we follow in this section. Computing Max-Cut exactly is NP-hard as shown by Karp [27] for the weighted version, and by [19] for the unweighted case. As for approximations, it is impossible to improve upon a 16/17-approximation for Max-Cut and a 12/13-approximation for Max-DiCut unless $P = NP$ [24, 38]. If every node chooses a cut side randomly, an expected 1/2-approximation for Max-Cut, a 1/4-approximation for Max-DiCut and a $(1 - 1/k)$-approximation is achieved. This can be derandomized using the method of conditional expectations. In the breakthrough paper of Goemans and Williamson [23] a 0.878-approximation is achieved using semidefinite programming. This is optimal under the unique games conjecture [28]. In the same paper a 0.796-approximation for Max-DiCut was presented. This was later improved to 0.863 in [MatuuraM01]. Other results using different techniques are presented in [26, 37].

[2] The full version can be found here: `https://arxiv.org/abs/1711.10155`.

In the distributed setting the problem has not received much attention. A node may choose a cut side at random, achieving the same guarantees as above in constant time. In [25] a distributed algorithm for d-regular triangle free graphs which achieves a $(1/2 + 0.28125/\sqrt{d})$-approximation ratio in a single communication round is presented. The only results for general graphs in the distributed setting is due to [12]. In the CONGEST model they present a deterministic 1/2-approximation for Max-Cut, a deterministic 1/3-approximation for Max-DiCut, and a randomized expected 1/2 approximation for Max-DiCut running in $\tilde{O}(\Delta + \log^* n)$ communication rounds. The results for Max-DiCut follow from adapting the unconstrained submodular maximization algorithm of [10] to the distributed setting. Better results are presented for the LOCAL model; we refer the reader to [12] for the full details. Recently, a lower bound of $O(1/\epsilon)$-rounds in the LOCAL model for any (even randomized) $(1-\epsilon)$-approximation algorithm for Max-Cut and Max-DiCut was presented in [8].

Max 2-SAT: The decision version of Max 2-SAT is NP-complete [19], and there exist several approximation algorithms [16, 23, 31, 33], of which currently the best known approximation ratio is 0.9401 [31]. In [3] it is shown that assuming the unique games conjecture, the approximation factor of [31] cannot be improved. Assuming only that $P \neq NP$ it cannot be approximated to within a 21/22-factor [24]. To the best of our knowledge the problem of Max 2-SAT (or Max-SAT) was not studied in the distributed model.

Correlation clustering: An excellent overview of correlation clustering (see Section 2 for a definition) appears in [1], which we follow in this section. Correlation clustering was first defined by [4]. Solving the problem exactly is NP-Hard, thus we are left with designing approximation algorithms for the problem, here one can try to approximate *max-agree* or *min-disagree*. If the graph is a clique, there exists a PTAS for max-agree [4, 22], and a 2.06-approximation for max-disagree [14]. For general (even weighted) graphs there exists a 0.7666-approximation for max-agree [13, 36], and a $O(\log n)$-approximation for min-disagree [15]. A trivial 1/2-approximation for max-agree on general graphs can be achieved by considering putting every node in a separate cluster, then considering putting all nodes in a single cluster, and taking the more profitable of the two.

In the distributed setting little is known about correlation clustering. In [11] a dynamic distributed MIS algorithm is provided, it is stated that this achieves a 3-approximation for min-disagree correlation clustering as it simulates the classical algorithm of Ailon *et al.* [2]. We note that the algorithm of Ailon *et al.* assumes the graph to be a clique, thus the above result is limited to complete graphs where the edges of the communication graph are taken to be the positive edges, and the non-edges are taken as the negative edges (as indeed for general graphs, the problem is APX-Hard, and difficult to approximate better than $\Theta(\log n)$ [15]). We also note that using only two clusters, where each node chooses a cluster at random, guarantees an expected 1/2-approximation for max-agree on weighted general graphs. We derandomize this approach in this paper.

2 Preliminaries

Sequential algorithms: The main goal of this paper is converting (local) sequential graph algorithms for unconstrained maximization (or minimization) to distributed graph algorithms. Let us first define formally this family of algorithms. The sequential algorithm receives as input a graph $G = (V, E)$, we associate each vertex $v \in V$ with a variable X_v taking values in some finite set A. The algorithm outputs a set of assignments $\overline{X} = \{X_v = \alpha_v\}$. The goal of the algorithms is to maximize some utility function $f(G, \overline{X})$ taking in a graph and the set of assignments and outputting some value in $\mathbb{R}$. For simplicity we assume that the order of the variables in $\overline{X}$ does not affect f, so we use a set notation instead of a vector notation. We somewhat abuse notation, and when assigning a variable we write $\overline{X} \cup \{X_v = \alpha\}$, meaning that any other assignment to X_v is removed from the set $\overline{X}$. We also omit G as a parameter when it is clear from context.

When considering randomized algorithms we assume the algorithm takes in a vector of random bits denoted by $\vec{r}$. This way of representing random algorithms is identical to having the algorithm generate random coins, and we use these two definitions interchangeably. The randomized algorithm aims to maximize the expectation of f, where the expectation is taken over the random bits of the algorithm.

Max k-Cut, Max-DiCut: In this paper we provide fast distributed approximation algorithms to some fundamental problems, which we now define formally. In the Max k-Cut problem we wish to divide the vertices into k disjoint sets, such that the weight of edges between different sets is maximized. In the Max-DiCut problem the edges are directed and we wish to divide the nodes into two disjoint sets, denoted A, B, such that the weight of edges directed from A to B is maximized.

Max 2-SAT: In the Max 2-SAT problem we are given a set of unique weighted clauses over some set of variables, where each clause contains *at most* two literals. Our goal is to maximize the weight of satisfied clauses. This problem is more general than the cut problems, so we must define what it means in the distributed context. First, the variables will be node variables as defined before. Second, each node knows all of the clauses it appears in as a literal.

Correlation clustering: We are given an edge weighted graph $G(V, E, w)$, such that each edge is also assigned a value from $\{+, -\}$ (referred to positive and negative edges). Given some partition, C, of the graph into disjoint clusters, we say that an edge *agrees* with C if it is positive and both endpoints are in the same cluster, or it is negative, and its endpoints are in different clusters. Otherwise we say it *disagrees* with C. We aim to find a partition C, using any number of clusters, such that the weight of edges that agree with C (agreements) is maximized (max-agree), or equivalently the weight of edges that disagree with C is minimized (min-disagree).

The problem is usually expressed as an LP using edge variables, where each variable indicates whether the nodes are in the same cluster. This allows a solution to use any number of clusters. In this paper we only aim to achieve a $(1/2 - \epsilon)$-approximation for the problem. This can be done rather simply without employing the full power of correlation clustering. Specifically, two clusters are enough for our case as we show that we can deterministically achieve $(1/2 - \epsilon)|E|$ agreements which results in the desired approximation ratio.

Local utility functions: We are interested in a type of utility function which we call a *local utility function.* Before we continue with the definition let us define an operator on assignments $\overline{X}$, we define $L_v[\overline{X}] = \big\{\{X_u = \alpha_u\} \in \overline{X} \mid u \in N(v)\big\}$. For convenience, when we pass $L_v[\overline{X}]$ as parameter to a function, we assume that the function also receives the 1-hop neighborhood of v which we do not write explicitly. We say that a utility function f, as defined above, is a local utility function if for every v there exists a function g_v s.t $f(\overline{X} \cup \{X_v = \alpha\}) - f(\overline{X} \cup \{X_v = \alpha'\}) = g_v(L_v[X], \alpha, \alpha')$. That is, to compute the change in the utility function which is caused by changing X_v from α' to α, we only need to know the immediate neighborhood of v, and the assignment to neighboring node variables. We note that for the cut problems considered in this paper the utility functions are indeed local utility functions. This is proven in the following Lemma:

▶ **Lemma 1.** *The utility functions for Max k-Cut, Max-DiCut and max-agree correlation clustering with 2 clusters are local utility functions.*

Proof. The utility functions for Max k-Cut is given by $f(\overline{X}) = \sum_{e=(w,u)\in E} w(e) \cdot X_w \oplus X_u$ where $X_w \oplus X_u = 0$ if $X_w = X_u$ and 1 otherwise. Thus, if we fix some v it holds that

$$\begin{aligned}
&f(\overline{X} \cup \{X_v = \alpha'\}) - f(\overline{X} \cup \{X_v = \alpha\}) \\
&= \sum_{e=(v,u)\in E} w(e) \cdot \alpha' \oplus X_u - \sum_{e=(v,u)\in E} w(e) \cdot \alpha \oplus X_u \\
&= \sum_{e=(v,u)\in E} w(e) \cdot (\alpha' \oplus X_u - \alpha \oplus X_u) \triangleq g_v(L_v[\overline{X}], \alpha', \alpha)
\end{aligned}$$

Because the final sum only depends on vertices $u \in N(v)$, the last equality defines the local function equivalent to the difference, and we are done.

For the problem of Max-DiCut the utility functions is given by $f(\overline{X}) = \sum_{e=(v\to u)\in E} w(e) \cdot X_v \wedge (1 - X_u)$, and for max-agree correlation clustering with 2 clusters the utility function is given by $f(\overline{X}) = \sum_{e=(v,u)\in E^+} w(e) \cdot (1 - X_v \oplus X_u) + \sum_{e=(v,u)\in E^-} w(e) \cdot X_v \oplus X_u$ (E^+, E^- are the positive and negative edges, respectively), and the proof is exactly the same. ◀

Submodular functions: A family of functions that will be of interest in this paper is the family of *submodular functions.* A function $f : \{0,1\}^\Omega \to \mathbb{R}$ is called a set function, with ground set Ω. It is said to be submodular if for every $S, T \subseteq \Omega$ it holds that $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. The functions we are interested in have V as their ground set, thus we remain with our original notation, setting $A = \{0,1\}$ and having f take in a set of binary assignments $\overline{X}$ as a parameter.

The method of conditional expectations: Next, we consider the method of conditional expectations. Let A be some set and $f : A^n \to \mathbb{R}$, next let $\overline{X} = (X_1, ..., X_n)$ be a vector of random variables taking values in A. We wish to be consistent with the previous notation, thus we treat $\overline{X}$ as a set of assignments. If $E[f(\overline{X})] \geq \beta$, then there is an assignment of values $\overline{Z} = \{X_i = \alpha_i\}_{i=1}^n$ such that $f(\overline{Z}) \geq \beta$. We describe how to *find* the vector Z. We first note that from the law of total expectation it holds that $E[f(\overline{X})] = \sum_{\alpha\in A} E[f(\overline{X}) \mid X_1 = \alpha] Pr[X_1 = \alpha]$, and therefore for at least some $\alpha \in A$ it holds that $E[f(\overline{X}) \mid X_1 = \alpha] \geq \beta$. We set this value to be α_1. We then repeat this process for the rest of the values in $\overline{X}$, which results in the set $\overline{Z}$. In order for this method to work we need it to be possible to *compute*[3] the conditional expectation of $f(\overline{X})$.

Graph coloring: A c-coloring for $G(V,E)$ is defined as a function $\varphi : V \to \mathcal{C}$. For simplicity we treat any set $\mathcal{C}$ of size c with some ordering as the set of integers $[c]$. This simplifies things as we can always consider $\varphi(v) \pm 1$, which is very convenient. We say that a coloring

[3] This point is critical, and this computation is not simple in many cases. In our case we also need this computation to be done *locally* at every nodes. We apply this technique to Max-Cut, which meets all of these demands.

```
Algorithm 1: OL($G, \vec{r}, \pi$).
1 $\forall v \in V, X_v = init(L_v[\overline{X}])$
2 Order the variables according to $\pi$: $v_1, v_2 ..., v_n$
3 for $i$ from 1 to $n$ do  $X_{v_i} = decide(L_v[\overline{X}], r_i)$
4 Return $\overline{X}$
```

is a legal coloring if $\forall v, u$ s.t $(v, u) \in E$ it holds that $\varphi(v) \neq \varphi(u)$. An important tool in this paper is *defective coloring*. Let us fix some c-coloring function $\varphi : V \to [c]$. We define the defect of a vertex to be the number of monochromatic edges it has. Formally, $defect(v) = size\{u \in N(v) \mid \varphi(v) = \varphi(u)\}$. We call φ a c-coloring with defect d if it holds that $\forall v \in V, defect(v) \leq d$. A classic result by Kuhn [29] states that for all $d \in \{1, 2, ..., \Delta\}$ an $O(\Delta^2/d^2)$-coloring with defect d can be computed deterministically in $O(\log^* n)$ rounds in the CONGEST model.

In this paper we define a new kind of defective coloring which we call a *weighted ϵ-defective coloring*. Given a positively edge weighted graph and any coloring, for every vertex we denote by $E_m(v)$ its monochromatic edges. Define its weighted defect as $defect_w(v) = \sum_{e=(u,v) \in E_m(v)} w(e)$. We aim to find a coloring s.t the defect for every v is below $\epsilon w(v) = \epsilon \sum_{v \in e} w(e)$. We show that the algorithm of Kuhn actually computes a weighted ϵ-defective $O(\epsilon^{-2})$-coloring. We state the following theorem (As the analysis is rather similar to the original analysis of Kuhn, the proof is deferred to the full version):

▶ **Theorem 2.** *For any constant $\epsilon \in (0, 1/e)$ a weighted ϵ-defective $O(\epsilon^{-2})$-coloring can be computed deterministically in $O(\log^* n)$ rounds in the CONGEST model.*

3 Orderless-local algorithms

Next we turn our attention to a large family of (potentially randomized) greedy algorithms. We limit ourselves to graph algorithms s.t every node v has a variable X_v taking values in some set A. We aim to maximize some global utility function $f(\overline{X})$. We focus on a class of algorithms we call *orderless-local* algorithms. These are greedy-like algorithms which may traverse the vertices in any order, and at each step decide upon a value for X_v. This decision is local, meaning that it only depends on the 1-hop topology of v and the values of neighboring variables. The decision may be random, but each variable has its own random bits, keeping the decision process local.

The code for a generic algorithm of this family is given in Algorithm 1. The algorithm first initiates the vertex variables. Next it traverses the variables in some order $\pi : V \to [n]$. Each X_{v_i} is assigned a value according to some function *decide*, which only depends on $L_v[\overline{X}]$ at the time of the assignment and some random bits $\vec{r}_i$ which are only used to set the value for that variable. Finally the assignment to the variables is returned. We are guaranteed that the expected value of f is at least $\beta(G)$ for any, potentially adversarial, ordering π of the variables. Formally, $E_{\vec{r}}[f(OL(G, \vec{r}, \pi))] \geq \beta(G)$.

We show that this family of algorithms can be easily distributed using coloring, s.t the running time of the distributed version depends on the number of colors. The distributed version, `OLDist`, is presented as Algorithm 2. The variables are all initiated as in the sequential version, and then the color classes are executed sequentially, while in each color class the nodes execute *decide* simultaneously, and send the newly assigned value to all neighbors. Decide does not communicate with the neighbors, so the algorithm finishes in $O(c)$ rounds.

Algorithm 2: OLDist($G, \vec{r}, \varphi$).

```
1 $\forall v \in V, X_v = init(L_v[\overline{X}])$
2 for $i$ from 1 to $c$ do
3     foreach $v$ s.t $\varphi(v) = i$ simultaneously do
4         $X_{v_i} = decide(L_v[\overline{X}], r_i)$
5         Send $X_{v_i}$ to neighbors
6     end
7 end
8 return $\overline{X}$
```

It is easy to see that given the same randomness both the sequential and distributed algorithms output the same result, this is because all decisions of the distributed algorithm only depend on the 1-hop environment of a vertex, and we are provided with a *legal* coloring. Thus, one round of the distributed algorithm is equivalent to many steps of the sequential algorithm. We prove the following lemma:

▶ **Lemma 3.** *For any graph G with a legal coloring φ, there exists an order π on the variables s.t it holds that $OL(G, \vec{r}, \pi) = OLDist(G, \vec{r}, \pi)$ for any $\vec{r}$.*

Proof. We prove the claim by induction on the executions of color classes by the distributed algorithm. We note that the execution of the distributed algorithm defines an order on the variables. Let us consider the i-th color class. Let us denote these variables as $\{X_{v_j}\}_{j=1}^{k}$, assigning some arbitrary order within the class. The ordering we analyze for the sequential algorithm would be $\pi(v_j) = (\varphi(v), j)$. Now both the distributed and sequential algorithms follow the same order of color classes, thus we allow ourselves to talk about the sequential algorithm finishing an execution of a color class.

Let Y_i be the assignments to all variables of the distributed algorithm after the i-th color class finishes execution. And let Y_i' be the assignments made by the sequential algorithm following π until all variable in the i-th color class are assigned. Both algorithms initiate the variables identically, so it holds that $Y_0 = Y_0'$. Assume that it holds that $Y_{i-1} = Y_{i-1}'$. The coloring is legal, so for any X_u, X_v, s.t $\varphi(u) = \varphi(v) = i$ it holds that $N(v) \cap u = \emptyset$. Thus, when assigning v, its neighborhood is not affected by any other assignments done in the color class, so the randomness is identical for both algorithms, and using the induction hypothesis all assignments up until this color class were identical. Thus, for all variables in this color class *decide* will be executed with the same parameters for both the distributed and sequential algorithms, and all assignments will be identical. ◀

Finally we show that for any graph G with a legal coloring φ, it holds that

$$E_{\vec{r}}[f(OLDist(G, \vec{r}, \varphi))] \geq \beta(G).$$

We know from Lemma 3 that for any coloring φ there exists an ordering π s.t $OL(G, \vec{r}, \pi) = OLDist(G, \vec{r}, \varphi)$ for any $\vec{r}$. The proof is direct from here:

$$E_{\vec{r}}[f(OLDist(G, \vec{r}, \varphi))] = \sum_{\vec{r}} Pr[\vec{r}] f(OLDist(G, \vec{r}, \varphi))$$

$$= \sum_{\vec{r}} Pr[\vec{r}] f(OL(G, \vec{r}, \pi)) = E_{\vec{r}}[f(OL(G, \vec{r}, \pi))] \geq \beta(G)$$

Algorithm 3: `CondExpSeq(`G`)`.

```
1 $\forall v \in V, X_v = \emptyset$
2 Order the variables according to any order: $v_1, v_2 ..., v_n$
3 for $i$ from 1 to $n$ do  $X_{v_i} = \text{argmax}_\alpha E[f(\overline{X}) \mid Y, X_{v_i} = \alpha_v] - E[f(\overline{X}) \mid Y]$
4 Return $\overline{X}$
```

We conclude that any orderless-local algorithm can be distributed, achieving the same performance guarantee on f, and requiring $O(c)$ communication rounds to finish, given a legal c-coloring. We state the following theorem:

▶ **Theorem 4.** *Given some utility function f, any sequential orderless-local algorithm for which it holds that $E_{\vec{r}}[f(OL(G, \vec{r}, \pi))] \geq \beta(G)$, can be converted into a distributed algorithm for which it holds that $E_{\vec{r}}[f(OLDist(G, \vec{r}, \varphi))] \geq \beta(G)$, where φ is a legal c-coloring of the graph. The running time of the distributed algorithm is $O(c)$ communication rounds.*

3.1 Distributed derandomization

We consider the method of conditional expectations in the distributed case for some local utility function $f(G, \overline{X})$, as defined in the preliminaries. Assume that the value of every X_v is set independently at random according to some distribution on A which depends only on the 1-hop neighborhood of v. We are guaranteed that $\forall G, E[f(G, \overline{X})] \geq \beta(G)$. Thus in the sequential setting we may use the method of conditional expectations to compute a deterministic assignment to the variables with the same guarantee. We show that because f is a local utility function, the method of conditional expectations applied on f is an orderless-local algorithm, and thus can be distributed.

Initially all variables are initiated to some value $\emptyset \notin A$, meaning the variable is unassigned. Let $Y = \{X_u = \alpha_u \mid u \in U \subseteq V\}$ be some partial assignment to the variables. The method of conditional expectations goes over the variables in any order, and in each iteration sets $X_{v_i} = \text{argmax}_\alpha E[f(\overline{X}) \mid Y, X_{v_i} = \alpha]$. This is equivalent to $\text{argmax}_\alpha \left\{E[f(\overline{X}) \mid Y, X_{v_i} = \alpha] - E[f(\overline{X}) \mid Y]\right\}$, as the subtracted term is just a constant. With this in mind, we present the pseudo code for the method of conditional expectations in Algorithm 3.

To show that Algorithm 3 is an orderless-local algorithm we only need to show that $\text{argmax}_\alpha E[f(\overline{X}) \mid Y, X_v = \alpha_v] - E[f(\overline{X}) \mid Y]$ can be computed locally for any v. We state the following lemma, followed by the main theorem for this section.

▶ **Lemma 5.** *The value* $\text{argmax}_\alpha E[f(\overline{X}) \mid Y, X_v = \alpha_v] - E[f(\overline{X}) \mid Y]$ *can be computed locally.*

Proof. It holds that:

$$\begin{aligned}
&E[f(\overline{X}) \mid Y, X_v = \alpha_v] - E[f(\overline{X}) \mid Y] \\
&= \sum_{\alpha \in A} E[f(\overline{X}) \mid Y, X_v = \alpha_v] Pr[X_v = \alpha] - \sum_{\alpha \in A} E[f(\overline{X}) \mid Y, X_v = \alpha] Pr[X_v = \alpha] \\
&= \sum_{\alpha \in A} Pr[X_v = \alpha](E[f(\overline{X}) \mid Y, X_v = \alpha_v] - E[f(\overline{X}) \mid Y, X_v = \alpha])
\end{aligned}$$

Where the first equality is due to the law of total expectation and the fact that $\sum_{\alpha \in A} Pr[X_v = \alpha] = 1$. The probability of assigning X_v to some value can be computed locally, so we are only left with the difference between the expectations. To show that this is indeed a local quantity we use the definition of expectation as a weighted summation over all possible assignments to unassigned variables. Let U_v be the set of all possible assignments to unassigned variables in $N(v)$ and let U be the set of all possible assignments to the rest of the unassigned variables. It holds that:

$$\begin{aligned}
&E[f(\overline{X}) \mid Y, X_v = \alpha_v] - E[f(\overline{X}) \mid Y, X_v = \alpha] \\
&= \sum_{Z_v \in U_v} \sum_{Z \in U} Pr[Z_v]Pr[Z]f(\overline{X} \cup Z_v \cup Z \cup \{X_v = \alpha_v\}) - f(\overline{X} \cup Z_v \cup Z \cup \{X_v = \alpha\}) \\
&= \sum_{Z_v \in U_v} \sum_{Z \in U} Pr[Z_v]Pr[Z]g_v(L_v[\overline{X} \cup Z_v \cup Z], \alpha, \alpha_v) \\
&= \sum_{Z_v \in U_v} \sum_{Z \in U} Pr[Z_v]Pr[Z]g_v(L_v[\overline{X} \cup Z_v], \alpha, \alpha_v) \\
&= \sum_{Z_v \in U_v} Pr[Z_v]g_v(L_v[\overline{X} \cup Z_v], \alpha, \alpha_v),
\end{aligned}$$

where in the first equality we use the definition of expectations and the fact that the variables are set independently of each other. Then we use the definition of a local utility function, and finally the dependence on U disappears due to the law of total probability. The final sum can be computed locally, as the probabilities for assigning variables in Z_v are known and g_v is local. ◀

▶ **Theorem 6.** *Let G be any graph and f a local utility function for which it holds that $E[f(\overline{X})] \geq \beta$, where the random assignments to the variables are independent of each other, and depend only on the immediate neighborhood of the node. There exists a distributed algorithm achieving the same value as the expected value for f, running in $O(c)$ communication rounds in the CONGEST model, given a legal c-coloring.*

3.2 Submodular Maximization

In this section we consider the problem of unconstrained submodular function maximization. Given an submodular function f (as defined in Section 2), we aim to find an input s.t the function is maximized. There are no constraints on the input set we pass to the function, hence it is 'unconstrained'. We are interested in finding an approximate solution to the problem, to this end, we consider both the deterministic and randomized algorithms of [10], achieving 1/3 and 1/2 approximation ratios for unconstrained submodular maximization. We show that both can be expressed as orderless-local algorithms for any local utility function. As the deterministic and randomized algorithms of [10] are almost identical, we focus on the randomized algorithm achieving a 1/2-approximation in expectation (Algorithm 5), as it is a bit more involved (The deterministic algorithm appears as Algorithm 4). The algorithms of [10] are defined for any submodular function, but as we are interested only in the case where the ground set is V, we will present it as such.

The algorithm maintains two variable assignment Z_i, Y_i, initially $Z_0 = \{X_v = 0 \mid v \in V\}$, $Y_0 = \{X_v = 1 \mid v \in V\}$. It iterates over the variables in any order, at each iteration it considers two nonnegative quantities a_i, b_i. These quantities represent the gain of either setting $X_{v_i} = 1$ in Z_{i-1} or setting $X_{v_i} = 0$ in Y_{i-1}. Next a coin is flipped with probability $p = a_i/(a_i + b_i)$, if $a_i = b_i = 0$ we set $p = 1$. If we get heads we set $X_{v_i} = 1$ in Z_i and otherwise we set it to 0 in Y_i. When the algorithm ends it holds that $Z_n = Y_n$, and this is our solution. The deterministic algorithm is almost identical, only that it allows a_i, b_i to take

Algorithm 4: `det-usm(`f`)`.

```
$Z_0 = \{X_v = 0 \mid v \in V\}$, $Y_0 = \{X_v = 1 \mid v \in V\}$
for $i$ from 1 to $n$ do
    $a_i = f(Z_{i-1} \cup \{X_{v_i} = 1\}) - f(Z_{i-1})$
    $b_i = f(Y_{i-1} \cup \{X_{v_i} = 0\}) - f(Y_{i-1})$
    if $a_i \geq b_i$ then
        $Z_i = Z_{i-1} \cup \{X_{v_i} = 1\}$
        $Y_i = Y_{i-1}$
    end
    else
        $Z_i = Z_{i-1}$
        $Y_i = Y_{i-1} \cup \{X_{v_i} = 0\}$
    end
end
return $Z_n$
```

Algorithm 5: `rand-usm(`f`)`.

```
$Z_0 = \{X_v = 0 \mid v \in V\}$, $Y_0 = \{X_v = 1 \mid v \in V\}$
Order the variables in any order $v_1, ..., v_n$
for $i$ from 1 to $n$ do
    $a_i = \max\{f(Z_{i-1} \cup \{X_{v_i} = 1\}) - f(Z_{i-1}), 0\}$
    $b_i = \max\{f(Y_{i-1} \cup \{X_{v_i} = 0\}) - f(Y_{i-1}), 0\}$
    if $a_i + b_i = 0$ then $p = 1$ else $p = a_i/(a_i + b_i)$
    $Y_i = Y_{i-1}, Z_i = Z_{i-1}$
    Flip a coin with probability $p$, if heads $Z_i = Z_i \cup \{X_{v_i} = 1\}$, else
     $Y_i = Y_i \cup \{X_{v_i} = 0\}$
end
return $Z_n$
```

negative values, and instead of flipping a coin it makes the decision greedily by comparing a_i, b_i.

We first note that the algorithm does not directly fit into our mold, as each vertex has two variables. We can overcome this, by taking X_v to be a binary tuple, the first coordinate stores its value for Z_i, and the other for Y_i. Initially it holds that $\forall v \in V, X_v = (0, 1)$, and our final goal function will only take the first coordinate of the variable. We note that because f is a local utility function the values a_i, b_i can be computed locally, this results directly from the definition of a local utility function, as we are interested in the change in f caused by flipping a single variable. Now we may rewrite the algorithm as an orderless-local algorithm, the pseudocode as Algorithm 6.

Using Theorem 4 we state our main result:

▶ **Theorem 7.** *For any graph G and a local unconstrained submodular function f with V as its ground set, there exists a randomized distributed 1/2-approximation, and a deterministic 1/3-approximation algorithms running in $O(c)$ communication rounds in the CONGEST model, given a legal c-coloring.*

Algorithm 6: `rand-usm(`$G, \vec{r}, \pi$`)`.

```
1 $\forall v \in V, X_v = (0,1)$
2 Order the vertices according to $\pi$
3 for $i$ from 1 to $n$ do
4     $X_u = decide(L_v[\overline{X}], \vec{r}_i)$
5 end
6 return $X_n$
```

Algorithm 7: `decide(`$L_v[\overline{X}], r$`)`.

```
1 $Z = \{X_u = \alpha_{u,1} \mid \{X_u = (\alpha_{u,1}, \alpha_{u,2})\} \in L_v[\overline{X}]\}$
2 $Y = \{X_u = \alpha_{u,2} \mid \{X_u = (\alpha_{u,1}, \alpha_{u,2})\} \in L_v[\overline{X}]\}$
3 $a = \max\{g_v(Z, 0, 1), 0\}$
4 $b = \max\{g_v(Z, 1, 0), 0\}$
5 if $a + b = 0$ then $p = 1$
6 else $p = a/(a + b)$
7 Flip coin with probability $p$
8 if heads then return (1,1)
9 else return (0,0)
```

3.3 Fast approximations for cut functions

Using the results of the previous sections we can provide fast and simple approximation algorithms for Max-DiCut and Max k-Cut. Lemma 1 guarantees that the utility functions for these problems are indeed local utility functions. For Max-DiCut we use the algorithms of Buchbinder *et al.*, as this is an unconstrained submodular function. For Max k-Cut each node choosing a side uniformly at random achieves a $(1 - 1/k)$ approximation, thus we use the results of Section 3.1. Theorem 7 and Theorem 6 immediately guarantee distributed algorithms, running in $O(c)$ communication rounds given a legal c-coloring.

Denote by $Cut(G, \varphi)$ one of the cut algorithms guaranteed by Theorem 7 or Theorem 6. We present two algorithms, `approxCutDet`, a deterministic algorithm to be used when $Cut(G, \varphi)$ is deterministic (Algorithm 8), and, `approxCutRand`, a randomized algorithm (Algorithm 9) for the case when $Cut(G, \varphi)$ is randomized. `approxCutDet` works by coloring the graph G using a weighted ϵ-defective coloring and then defining a new graph G' by dropping all of the monochromatic edges. This means that the coloring is a legal coloring for G'. Finally we call one of the deterministic cut functions. `approxCutRand` is identical, apart from the fact that nodes choose a color uniformly at random from $[\lceil \epsilon^{-1} \rceil]$.

For `approxCutDet`, the running time of the coloring is $O(\log^* n)$ rounds, returning a weighted ϵ-defective $O(\epsilon^{-2})$-coloring. The running time of the cut algorithms is the number of colors, thus the total running time of the algorithm is $O(\epsilon^{-2} + \log^* n)$ rounds. Using the same reasoning, the running time of `approxCutRand` is $O(\epsilon^{-1})$. It is only left to prove the approximation ratio. We prove the following lemma:

▶ **Lemma 8.** *Let $G(V, E, w)$ be any graph, and let $G'(V, E', w)$ be a graph resulting from removing any subset of edges from G of total weight at most $\epsilon \sum_{e \in E} w(e)$. Then for any constant p, any p-approximation for Max-DiCut or Max k-Cut for G' is a $p(1-4\epsilon)$-approximation for G.*

Algorithm 8: `approxCutDet`(G, ϵ).

```
1 $\varphi$ = epsilonColor$(G, \epsilon)$
2 Let $G' = (V, E' = \{(v,u) \in E \mid \varphi(v) \neq \varphi(u)\})$
3 Cut$(G', \varphi)$
```

Algorithm 9: `approxCutRand`(G, ϵ).

```
1 Each vertex $v$ chooses $\varphi(v)$ uniformly at random from $[\lceil \epsilon^{-1} \rceil]$
2 Let $G' = (V, E' = \{(v,u) \in E \mid \varphi(v) \neq \varphi(u)\})$
3 Cut$(G', \varphi)$
```

Proof. Let OPT, OPT' be the size of optimal solutions for G, G'. It holds that $OPT' \geq OPT - \epsilon \sum_{e \in E} w(e)$, as any solution for G is also a solution for G' whose value differs by at most $\epsilon \sum_{e \in E} w(e)$ (the weight of discarded edges). Assigning every node a cut side uniformly at random the expected cut weight is at least $\sum_{e \in E} w(e)/4$ for Max-DiCut and Max k-Cut. Using the probabilistic method this implies that $OPT \geq \sum_{e \in E} w(e)/4$. Using all of the above we can say that given a p-approximate solution for OPT' it holds that: $p \cdot OPT' \geq p(OPT - \epsilon \sum_{e \in E} w(e)) \geq p(OPT - 4\epsilon OPT) = p(1 - 4\epsilon)OPT$ ◀

Lemma 8 immediately guarantees the approximation ratio for the deterministic algorithm. As for the randomized algorithm, let the random variable δ be the fraction of edges removed, let p be the approximation ratio guaranteed by one of the cut algorithms and let ρ be the approximation ratio achieved by `approxCutRand`. We know that $E_\rho[\rho \mid \delta] = p(1 - 4\delta)$. Applying the law of total expectations we get that $E[\rho] = E_\delta[E_\rho[\rho \mid \delta]] = E_\delta[p(1 - 4\delta)] = p(1 - 4\epsilon)$. We state our main theorems for this section.

▶ **Theorem 9.** *There exists a deterministic $(1 - 1/k - \epsilon)$-approximation algorithms for Weighted Max k-Cut running in $O(\log^* n)$ communication rounds in the CONGEST model.*

▶ **Theorem 10.** *There exists a deterministic $(1/3 - \epsilon)$-approximation algorithm for Weighted Max-DiCut running in $O(\log^* n)$ communication rounds in the CONGEST model.*

▶ **Theorem 11.** *There exists a randomized distributed expected $(1/2 - \epsilon)$-approximation for Weighted Max-DiCut running in $O(\epsilon^{-1})$ communication rounds in the CONGEST model.*

Correlation clustering

We note the same techniques used for Max-Cut work directly for max-agree correlation clustering on general graphs. Specifically, if we divide the nodes into two clusters, s.t each node selectes a cluster uniformly at random, each edge has exactly probability $1/2$ to agree with the clustering, thus the expected value of the clustering is $\sum_{e \in E} w(e)/2$, which is a $1/2$-approximation. The above can be derandomized exactly in the same manner as Max-Cut, meaning this is an orderless local algorithm. Finally, we apply the weighted ϵ-defective coloring algorithm twice (note that we ignore the sign of the edge), discard all monochromatic edges and execute the deterministic algorithm guaranteed from Theorem 6 with a legal coloring. Because there must exists a clustering which has a value at least $\sum_{e \in E} w(e)/2$, a lemma identical to Lemma 8 can be proved and hence we are done. We state the following theorem:

▶ **Theorem 12.** *There exists a deterministic $(1/2 - \epsilon)$-approximation algorithms for weighted max-agree correlation clustering on general graphs, running in $O(\log^* n)$ communication rounds in the CONGEST model.*

References

1 Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2237–2246. JMLR.org, 2015.

2 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008.

3 Per Austrin. Balanced max 2-sat might not be the hardest. In *STOC*, pages 189–197. ACM, 2007.

4 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *FOCS*, page 238. IEEE Computer Society, 2002.

5 Reuven Bar-Yehuda, Keren Censor-Hillel, Mohsen Ghaffari, and Gregory Schwartzman. Distributed approximation of maximum independent set and maximum matching. In *PODC*, pages 165–174. ACM, 2017.

6 Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A distributed $(2 + \epsilon)$-approximation for vertex cover in o(log Δ / ϵ log log Δ) rounds. *J. ACM*, 64(3):23:1–23:11, 2017.

7 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007.

8 Ran Ben-Basat, Ken-ichi Kawarabayashi, and Gregory Schwartzman. Parameterized distributed algorithms. *CoRR*, abs/1807.04900, 2018. `arXiv:1807.04900`.

9 Allan Borodin, Morten N. Nielsen, and Charles Rackoff. (incremental) priority algorithms. In *SODA*, pages 752–761. ACM/SIAM, 2002.

10 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM J. Comput.*, 44(5):1384–1402, 2015. `doi:10.1137/130929205`.

11 Keren Censor-Hillel, Elad Haramaty, and Zohar S. Karnin. Optimal dynamic distributed MIS. In *PODC*, pages 217–226. ACM, 2016.

12 Keren Censor-Hillel, Rina Levy, and Hadas Shachnai. Fast distributed approximation for max-cut. In *Algorithms for Sensor Systems, 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Vienna, Austria, September 7–8, 2017, Revised Selected Papers*, volume 10718 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2017.

13 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *J. Comput. Syst. Sci.*, 71(3):360–383, 2005.

14 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlationclustering on complete and complete k-partite graphs. In *STOC*, pages 219–228. ACM, 2015.

15 Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theor. Comput. Sci.*, 361(2-3):172–187, 2006.

16 Uriel Feige and Michel X. Goemans. Aproximating the value of two prover proof systems, with applications to MAX 2sat and MAX DICUT. In *ISTCS*, pages 182–189. IEEE Computer Society, 1995.

17 Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *FOCS*, pages 180–191. IEEE Computer Society, 2017.

18 Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, 1983.

19 M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976. `doi:10.1016/0304-3975(76)90059-1`.

20 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. *CoRR*, abs/1711.02194, 2017.

21 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *STOC*, pages 784–797. ACM, 2017.

22 Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(13):249–266, 2006.

23 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. `doi:10.1145/227683.227684`.

24 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

25 Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large cuts with local algorithms on triangle-free graphs. *CoRR*, abs/1402.2543, 2014.

26 Satyen Kale and C. Seshadhri. Combinatorial approximation algorithms for maxcut using random walks. In *ICS*, pages 367–388. Tsinghua University Press, 2011.

27 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

28 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, 2007.

29 Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *SPAA*, pages 138–144. ACM, 2009.

30 Reut Levi and Moti Medina. A (centralized) local guide. *Bulletin of EATCS*, 2(122), 2017.

31 Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-sat and MAX DI-CUT problems. In *IPCO*, volume 2337 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2002.

32 Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

33 Shiro Matuura and Tomomi Matsui. 0.863-approximation algorithm for MAX DICUT. In *RANDOM-APPROX*, volume 2129 of *Lecture Notes in Computer Science*, pages 138–146. Springer, 2001.

34 Matthias Poloczek, Georg Schnitger, David P. Williamson, and Anke van Zuylen. Greedy algorithms for the maximum satisfiability problem: Simple algorithms and inapproximability bounds. *SIAM J. Comput.*, 46(3):1029–1061, 2017.

35 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *ICS*, pages 223–238. Tsinghua University Press, 2011.

36 Chaitanya Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *SODA*, pages 526–527. SIAM, 2004.

37 Luca Trevisan. Max cut and the smallest eigenvalue. *SIAM J. Comput.*, 41(6):1769–1786, 2012.

38 Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comput.*, 29(6):2074–2097, 2000.