ACM SIGACT News Distributed Computing Column 16

Sergio Rajsbaum*

Abstract

The Distributed Computing Column covers the theory of systems that are composed of a number of interacting computing elements. These include problems of communication and networking, databases, distributed shared memory, multiprocessor architectures, operating systems, verification, Internet, and the Web.

This issue contains two items:

- An eulogy in memory of Larry Stockmeyer, who passed away on July 31, 2004. Larry did pioneering work in various areas, such as complexity theory, logic, interactive proofs, distributed computing, parallel computing, secure architectures, database privacy, and storage systems. A celebration of Larry's work will be held May 21–22, 2005, in conjunction with STOC 2005 (May 22–24). On May 21, a tutorial by Nick Pippenger (Princeton) on some of Stockmeyer's fundamental results in complexity theory will be followed by lectures by Miki Ajtai (IBM), Anne Condon (UBC), Cynthia Dwork (Microsoft), Richard Karp (UC Berkeley), Albert Meyer (MIT), and Chris Umans (CalTech). Some time will be reserved for personal remarks. If you wish to speak in this part of Larry's commemoration contact Cynthia Dwork (dwork@microsoft.com). The celebration of Stockmeyer's work will conclude on May 22, with Lance Fortnow giving a keynote address to STOC.
- "Distributed Approximation," a survey by Michael Elkin. Considerable progress has been recently achieved in designing efficient distributed approximation algorithms, and in demonstrating hardness of distributed approximation for various problems. An overview of the research in this area is presented, including several directions for future research.

Many thanks to Michael for his contribution to this issue.

Request for Collaborations: Please send me any suggestions for material I should be including in this column, including news and communications, open problems, and authors willing to write a guest column or to review an event related to theory of distributed computing.

*Instituto de Matemáticas, UNAM. Ciudad Universitaria, Mexico City, D.F. 04510 `rajsbaum@math.unam.mx`.

Larry Stockmeyer 1948–2004

Larry was born in Evansville, Indiana, in 1948. He attended college at MIT, receiving a B.S. in mathematics and an M.S. in electrical engineering in 1972, and a Ph.D. in computer science in 1974. He spent all of his professional career at IBM. He started in 1974 at the Thomas J. Watson Research Center, and in 1982 moved to what is now the Almaden Research Center. He retired from IBM in November 2003. In 1996 he was named an ACM Fellow. Larry passed away on July 31, 2004, at age 55.

Larry made many important contributions to the theory of computation. One – extremely important– example is his discovery (with Albert Meyer in FOCS 1972) of the polynomial time hierarchy. He also did important work in parallel and distributed computing. He had three influential papers about consensus in partial synchrony models. After the FLP result that proved consensus to be impossible in an asynchronous system where one process can crash, finding realistic systems where consensus could be solved became an active research area. Larry's paper "On the Minimal Synchronism Needed for Distributed Consensus" (with Danny Dolev and Cynthia Dwork in FOCS 1983 and JACM 1987) helped explaining why consensus is possible in certain models but not possible in others. It considered various synchrony conditions, examined how varying these affects the number of faults that can be tolerated by a consensus algorithm. In "Consensus in the presence of partial synchrony" (with Dwork and Nancy Lynch, JACM 1988), the concept of partial synchrony in a distributed system is studied, and consensus protocols are given for various cases of partial synchrony and various fault models. Lower bounds that show in most cases that the protocols are optimal with respect to the number of faults tolerated are also given. Also, they introduced fault-tolerant clock synchronization protocols, and used them in their protocols. In "Bounds on the time to reach agreement in the presence of timing uncertainty" (with Hagit Attiya, Dwork and Lynch in JACM 1994) upper and lower bounds are proved for the time complexity of solving consensus in a crash failure model, where time for message delivery is at most d, and the amount of time between any two consecutive steps of any nonfaulty process is at least c_1 and at most c_2; thus, $C = c_2/c_1$ is a measure of the timing uncertainty. On a different subject, "Lower bounds on the competitive ratio for mobile user tracking and distributed job scheduling" (with Noga Alon, Gil Kalai and Moty Ricklin, TCS 1994) proves a lower bound of $\Omega(\log n/\log\log n)$ on the competitive ratio for solving a mobile user problem and a Distributed Job Scheduling problem. His paper "What can be computed locally?" (with Moni Naor, SICOMP 1995) studies computation that can be done within time independent of the size of the network.

These distributed computing papers are consistent with others by Larry – serious, deep work. They further illustrate how broad his research was. He will be missed by many people.

Distributed Approximation - A Survey[1]

Michael Elkin [2]

Abstract

Recently considerable progress was achieved in designing efficient *distributed approximation algorithms*, and in demonstrating *hardness of distributed approximation* for various problems. In this survey we overview the research in this area and propose several directions for future research.

1 Introduction

The area of *distributed approximation* is a new rapidly developing discipline that lies on the boundary between two well-established areas, specifically *distributed computing* and *approximability.*

1.1 Distributed Computing

Distributed computing is a very broad area that encompasses a plethora of disciplines. The particular subarea of distributed computing that is relevant to this survey focuses on the so-called *message-passing model* of computation [5, 13, 23]. In this model we are given a network of processors modeled by an unweighted undirected N-vertex graph $G = (V, E)$. The processors reside in the vertices of the graph, and the edges of E model the links of the network. The processors (henceforth, vertices) have *infinite computational power*, but their *initial knowledge is limited* to their immediate neighborhood. The network is to solve (cooperatively) some *global* problem, whose resolution requires communication between the vertices. The vertices are allowed to communicate via the links of the network (henceforth, *edges*). The communication is *synchronous*, that is, it occurs in discrete *rounds.* On each round at most B bits can be sent through each edge in each direction, where B is the *bandwidth* parameter of the network. A particularly interesting case emerges when B is equal to infinity [24].

The most commonly used measure of efficiency of distributed algorithms is the (worst-case) *number of rounds of communication.* (Note that that the local computation is completely free under this measure.) This efficiency measure naturally gives rise to the *time complexity* measure of problems.

The message-passing model was subject to extensive research for the last two decades (see [28] for an excellent comprehensive survey).

1.2 Approximability

The study of *approximate* variants of combinatorial optimization problems in the Turing machine model of computation is an extremely active mainstream area of research. In the quest for better

[1]This work was done in Yale University, and was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research under Grant N00014-01-1-0795.

[2]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel, `elkinm@cs.bgu.ac.il`

approximation algorithms researchers developed a great variety of elegant and sophisticated mathematical techniques. The study of lower bounds, in turn, gave rise to the algebraically deep PCP theory, which is as great a contribution to Complexity Theory as to Approximability Theory. In other words, the study of the approximability of problems in a given computational model turns out to be instrumental for achieving a better understanding of the capabilities and the limitations of the model. Additionally, such a study enables to expose important structural properties of the studied problems; properties that can be left unexploited when one analizes the *exact variants* of the same problems.

1.3 Distributed Approximation

With this motivation in mind, it is not surprising that much of the recent research efforts in the study of the message-passing model were funneled to *distributed approximation*. Fortunately, those efforts stood up to the expectations in shedding a new light also on the *exact* variants of some of the most important problems in the area, such as the maximal independent set, the minimum spanning tree (henceforth, *MST*), and the maximal matching problems [18, 9]. In fact, the state-of-the-art lower bounds for (the exact versions of) these problems are mere corollaries of more general results concerning the *hardness of distributed approximation*.

1.3.1 Upper Bounds

Distributed approximation algorithms are currently available for a number of problems. One of them is the *minimum dominating set* (henceforth, *MDS*) problem [16, 6, 19]. Another relatively well-studied problem is the *minimum edge-coloring* [26, 4, 14, 2]. Approximation algorithms for the *maximum matching* problem were developed in [3, 32]. Algorithms for distributed *distance estimation* were developed in [8, 10]. Kuhn et al. [17] devised an approximation algorithm for the *minimum vertex cover* problem. Furthermore, in what appears to be the most general upper bound in this area so far, they were able to extend their algorithm to solve *positive linear programs* [20]. This result, in turn, gave rise to improved approximation algorithms for the minimum dominating set, minimum vertex cover, and maximum matching problems. (The study of distributed algorithms for positive linear programs dates back to the papers of Papadimitriou and Yannakakis [27], and of Bartal et al. [1].)

1.3.2 Lower Bounds

Until very recently all the existing results on hardness of distributed approximation could have been divided to two categories.

First, there were inapproximability results that are based on lower bounds on the time required for *exact* solution of certain problems, and on integrality of the objective functions of these problems. For example, there is a classical result due to Linial [24] saying that 3-coloring an N-vertex ring requires $\Omega(\log^* N)$ time. In particular, it implies that any $3/2$-approximation protocol for the vertex-coloring problem requires $\Omega(\log^* N)$ time.

Second, there were inapproximability results that assume that the vertices are computationally limited, e.g., are allowed to perform at most polynomial in N number of operations. Obviously, under this assumption any NP-hardness inapproximability result immediately gives rise to an analogous result in the distributed model.

Note, however, that neither of these inapproximability results sheds a new light on our understanding of the limitations of distributed computing. Specifically, the results of this sort are just somewhat different *semantic interpretations* of already known lower bounds. Additionally, we believe that imposing restrictions on the computational power of the vertices is as unnatural as limiting the computational power of the parties in the two-party communication complexity model (see [21]). In both cases the abstraction of computationally unbounded vertices or parties is necessary to make possible the study of the role that *communication* plays in computation (see also [24, 28]).

Recently strong lower bounds on the hardness of distributed approximation for the MST problem were established by the author in [9]. In another even more recent development Kuhn et al. [17] have shown a series of strikingly elegant lower bounds on the approximability of the minimum vertex cover, the minimum dominating set, and maximum matching problems. As was already mentioned, their technique enabled them to improve drastically the classical lower bounds of Linial [24] for the complexity of the maximal independent set (henceforth, *MIS*) and maximal matching problems.

1.3.3 The Structure of the Survey

The rest of the survey is organized as follows. Section 2 is devoted to the upper bounds, and is based on the work of Kuhn et al. [17, 19, 20]. Section 3 is devoted to the lower bounds. In Section 3.1 we describe the lower bound of [9] for the minimum spanning tree problem, and in Section 3.2 we turn to the lower bounds of [17, 18] for the minimum vertex cover and minimum dominating set problems. We conclude (Section 4) with some open problems.

2 Upper Bounds

Linear Programming (henceforth, LP) is one of the most general and useful tools in algorithmic design. Particularly, there is a great multitude of different approximation algorithms that solve an LP and employ one of the rounding techniques (see [15, 31] for the systematic overview of such algorithms).

However, it is a-priori not clear that this powerful methodology can be applied in the distributed setting. Indeed, all the known algorithms for solving general linear programs (LPs) need to process the entire matrix that defined the LP. It seems that none of these algorithms (the Ellipsoid method, the Simplex, the Interior points method) can possibly succeed in a scenario when the matrix is distributed among processors that *share no common memory*, each keeping only a tiny fraction of the matrix, and with only a scarce communication between them.

Though understanding the distributed complexity of solving general LPs is an outstanding open problem, there are currently available distributed algorithms for solving LPs of certain particularly important type, so-called *positive LPs*. Positive LPs are LPs that are defined by matrices and vectors that satisfy that all their entries are non-negative. LPs of this type are also often called *fractional covering* or *packing* (depending on the form of the particular LP, see below) LP. Positive LPs were intensively studied [25, 30, 7, 11, 12] in the course of the last decade. Many important combinatorial problems, such as the minimum vertex cover, the minimum dominating set, the maximum matching, can be modeled by a positive LP.

Papadimitriou and Yannakakis [27] were the first to show that positive LPs can be approximated rather well even if the matrix is distributed among multiple processors that *are not allowed to communicate*. Further, Bartal et al. [1] have shown that in the distributed setting (that is, when the processors are allowed to communicate), this approximation can be achieved rather efficiently. More recently, Kuhn and Wattenhofer [19, 20] have improved and generalized the results of [1], and have used these LP-solving algorithms for providing efficient distributed approximation algorithms for the minimum vertex cover, the minimum dominating set, the maximum matching problems.

In the rest of this section we will describe in detail the $O(k\Delta^{2/k}\log\Delta)$-approximation algorithm of [19] for the *MDS* problem. Δ is the maximum degree of the input graph, and $k = 1, 2, \ldots$ is a parameter that determines the running time of the algorithm. (Specifically, the running time is $O(k^2)$).

For a graph $G = (V, E)$, and a vertex v, let $\Gamma(v) = \{v\} \cup \{u \mid \{u, v\} \in E\}$. A subset $S \subseteq V$ of vertices is a *dominating set* if it satisfies that $V = \bigcup_{v \in S} \Gamma(v)$. The *MDS* problem asks to compute a dominating set of minimum size.

Let $V = \{1, 2, \ldots, N\}$ (we also use $[N]$ to denote $\{1, 2, \ldots, N\}$), and let A denote the adjacency matrix of the graph G. The *MDS* problem can now be modeled by the integer program

$$\begin{array}{ll} \min \sum_{i=1}^{N} x_i & \text{(IP-MDS)} \\ \text{s.t. } \; A\underline{x} \geq \underline{1}, \;\; \underline{x} \in \{0,1\}^N & , \end{array}$$

and relaxed by the same program (denoted (LP-MDS)) with the condition $\underline{x} \in \{0,1\}^N$ replaced by $\underline{x} \geq \underline{0}$.

LPs of this form are called *covering LPs*. Since $A = A^T$, the dual LP (denoted (DLP-MDS)) is given by

$$\max \sum_{i=1}^{N} y_i \;\; \text{s.t.} \;\; A\underline{y} \leq \underline{1}, \; \underline{y} \geq \underline{0} \; .$$

LPs of this form are called *packing LPs*.

For a vertex $i \in V$, let $\delta^{(1)}(i)$ denote the maximum degree in $\Gamma(i)$, and $\delta^{(2)}(i)$ denote the maximum degree in the 2-neighborhood of i, that is, $\delta^{(2)}(i) = \max\{\delta^{(1)}(i) \mid j \in \Gamma(i)\}$. We need the following standard fact.

Fact 2.1 *For any dominating set $S \subseteq V$,*

$$\sum_{i=1}^{N} \frac{1}{\delta^{(1)}(i) + 1} \;\leq\; |S| \; .$$

(Note that the assignment $y_i = \frac{1}{\delta^{(1)}(i)+1}$ for every $i \in V$ satisfies the (DLP-MDS). The fact now follows from the LP-duality theorem.)

2.1 Rounding Algorithm

We next show that a solution for (LP-MDS) can be efficiently distributively rounded to a solution for (IP-MDS).

Let $\underline{x}$ be a feasible solution for (LP-MDS), and suppose that every vertex $i \in V$ knows x_i. The vertex i also maintains a boolean indicator variable S_i which will be eventually set to 1 if the vertex i will end up in the returned dominating set S, and will be set as 0 otherwise.

The vertex i starts the algorithm by setting S_i as 1 with probability $p_i = \min\{1, x_i \cdot \ln(\delta_i^{(2)} + 1)\}$, and sets it as 0 otherwise. Then the vertex i sends S_i to each of its neighbors. Then it checks whether one of his neighbors j set his S_j to 1, and if it is not the case, the vertex i sets its S_i to 1 (independently of the random coin that was tossed with probability p_i). This concludes the rounding algorithm.

Note that the last step of the algorithm ensures that S is a valid dominating set. Note also that the algorithm requires only a constant number of rounds of communication. To provide an upper bound on the expected size of S, note that $|S| \le X + Y$, where X is the number of vertices i who set their variable S_i as 1 as result of the coin toss, and Y is the number of vertices i that discovered that none of their neighbors set S_i to 1.

Observe that

$$\mathbb{E}(X) \;=\; \sum_{i=1}^{N} p_i \;\le\; \sum_{i=1}^{N} x_i \cdot \ln(\delta_i^{(2)} + 1) \;\le\; (\ln(\Delta + 1)) \sum_{i=1}^{N} x_i \; .$$

Let X_i be the indicator random variable that reflects the result of the coin toss of the vertex i, and Y_i be the indicator random variable that is set to 1 only if for every $j \in \Gamma(i)$ the variable X_j is equal to 0. Then

$$\begin{aligned}
\mathbb{E}(Y_i) &= \Pi_{j\in\Gamma(i)}(1 - \mathbb{P}(X_j = 1)) \;=\; \Pi_{j\in\Gamma(i)}(1 - p_j) \\
&\le \Pi_{j\in\Gamma(i)}(1 - x_j \ln(\delta_j^{(2)} + 1)) \;\le\; \Pi_{j\in\Gamma(i)}(1 - x_j \ln(\delta_i^{(1)} + 1)) \\
&\le e^{-\sum_{j\in\Gamma(i)} x_j \ln(\delta_i^{(1)}+1)} \;\le\; e^{-\ln(\delta_i^{(i)}+1)} \;=\; \frac{1}{\delta_i^{(1)} + 1} \; .
\end{aligned}$$

(The first inequality is because if for some $j \in \Gamma(i)$, $p_j = 1$, then $\mathbb{E}(Y_i) = 0$. The last inequality is because $\underline{x}$ is a feasible solution for (LP-MDS).)

Consequently, $\mathbb{E}(Y) \le \sum_{i=1}^{N} \frac{1}{\delta_i^{(1)}+1} \le |S'|$, for any dominating set S' (by Fact 2.1). Particularly, $\mathbb{E}(Y) \le |S^*|$, where S^* is the minimum dominating set. Hence $\mathbb{E}(|S|) \le \mathbb{E}(X) + \mathbb{E}(Y) \le (\ln(\Delta + 1))(\sum_{i=1}^{N} x_i) + |S^*|$.

If $\underline{x}$ is an optimum solution for (LP-MDS) then obviously $\sum_{i=1}^{N} x_i \le |S^*|$, and consequently $\mathbb{E}(|S|) \le (\ln(\Delta + 1) + 1)|S^*|$. Note also that if $\underline{x}$ is an α-approximate solution for (LP-MDS) for some $\alpha > 1$, then the expected size of S is at most $(\alpha \ln(\Delta + 1) + 1)|S^*|$.

Consequently, any distributed $O(k\Delta^{2/k})$-approximation algorithm for (LP-MDS) can be converted into an $O((\log \Delta)k\Delta^{2/k})$-approximation algorithm for the minimum dominating set problem with essentially the same running time.

2.2 Approximation Algorithm for (LP-MDS)

We next describe the approximation algorithm of [19] for the (LP-MDS).

During the algorithm every vertex i maintains a few local variables, specifically, the eventual output x, the color *color*, the dynamic degree $\tilde{\delta}$, and two additional variables A and z. The variable x is initialized as 0, and is eventually returned as the ith coordinate of the solution of (LP-MDS) produced by the algorithm. The value of x never decreases, and never becomes greater than 1 (from obvious reasons). *color* is a boolean variable that is initially set as "white", and it is changed to "gray" essentially right after the condition $\sum_{j\in\Gamma(i)} x_j \ge 1$ is satisfied. The variable $\tilde{\delta}$ reflects the number of white-colored neighbors that the vertex has.

The general structure of the algorithm is a double loop. On each iteration ℓ of the outer loop the value of the variable x may increase. All vertices whose variables x grow on iteration ℓ are called *active with respect to iteration ℓ*. The variable A of the vertex i counts the *number of active neighbors* that the vertex has on the current iteration. This variable is used only for the convenience of the analysis, and an actual implementation of the algorithm can do without it. The same is true for the variable z. Every increase in the value of the variable $x = x(i)$ of the vertex i is distributed equally between the variables z of the vertices of $\Gamma(i)$ that had white color prior to the increase of $x(i)$. This way the increase of the value of $x(i)$ is charged to its neighbors. It turns out that the variable z changes "more smoothly" than the variable x, making the analysis of $\sum_{i \in V} z(i)$ technically easier than that of $\sum_{i \in V} x(i)$.

The formal description of the algorithm follows.

Algorithm 1 The algorithm of [19] for approximating (LP-MDS).

$x := 0$; $\tilde{\delta} := deg(v)$;
for $\ell := k-1, \ldots, 0$ **do**
  $z := 0$; $A := 0$;
  for $h := k-1, \ldots, 0$ **do**
    send *color* to all neighbors;
    update $\tilde{\delta}$;
    if $\tilde{\delta} \geq (\Delta+1)^{\ell/k}$ **then**
      $x := \max\{x, \frac{1}{(\Delta+1)^{h/k}}\}$;
    end if
    send x to all neighbors;
    update *color*, z and A;
  end for
end for

Note that the algorithm assumes that every vertex knows the maximum degree Δ of the graph; it is, however, shown in [19] that the algorithm can be adapted to work with essentially the same running time and provide the same approximation guarantee even when the vertices do not know Δ in advance. (This adaptation is, however, not trivial. An interested reader should consult [19].)

The algorithm is designed to ensure that

1. The dynamic degree $\tilde{\delta}$ is at most $(\Delta+1)^{\frac{\ell+1}{k}}$ in the beginning of iteration ℓ of the outer loop.

2. The variable A that counts the number of active vertices in the neighborhood is at most $(\Delta+1)^{\frac{h+1}{k}}$ at the beginning of iteration h of the inner loop.

3. At the end of iteration ℓ of the outer loop $z \leq (\Delta+1)^{-\frac{\ell-1}{k}}$.

The first two properties follow in a rather simple way from the algorithm. The third property requires a more delicate argument, which we next describe.

Proof: (of (3)) Let $t \in \{0, 1, \ldots, k-1\}$ be the index of the iteration of the inner loop on which the vertex i becomes gray. (If the vertex i does not become gray on iteration ℓ of the outer loop, then the proof is simpler; see below.) Observe that z may grow only on iterations $h \in \{k-1, k-2, \ldots, t\}$. We analyze separately the growth of z on iterations $h \in \{k-1, k-2, \ldots, t+1\}$, and its growth on iteration t (of the inner loop).

By definition, for $h \in \{k-1, \ldots, t+1\}$, $\sum_{j \in \Gamma(i)} x(j) < 1$, and for every *active* vertex j, $\tilde{\delta}(j) \geq (\Delta+1)^{\ell/k}$. Since the value of the variable $z = z(i)$ of the vertex i grows only when $x(j)$ grows for some neighbor $j \in \Gamma(i)$, it follows that

$$z(i) \;<\; \sum_{j \in \Gamma(i)} \frac{x(j)}{\tilde{\delta}(j)} \;\leq\; \frac{\sum_{j \in \Gamma(i)} x(j)}{(\Delta+1)^{\ell/k}} \;<\; \frac{1}{(\Delta+1)^{\ell/k}}\;. \tag{1}$$

The last inequality holds for the value of the variable $z(i)$ at the end of iteration $t+1$ of the inner loop. If the vertex i does not become gray on iteration ℓ of the outer loop, then this completes the proof.

We next analyze the increase of $z(i)$ on iteration t. Recall that a vertex is active if its dynamic degree is at least $(\Delta+1)^{\ell/k}$. The quantity $(\Delta+1)^{\ell/k}$ is constant during iteration ℓ of the outer loop, and the dynamic degrees may only decrease. Hence all the vertices j that were active on iteration $t+1$ of the inner loop, are active on iteration t as well. Hence step 8 of the algorithm ensures that for every vertex j that is active on iteration t, $x(j) \geq \frac{1}{(\Delta+1)^{\frac{t+1}{k}}}$. On step 8 of iteration t the value of $x(j)$ either stays unchanged, or grows to at most $\frac{1}{(\Delta+1)^{t/k}}$. Hence the increase in $x(j)$, denoted $d(x(j))$, is at most

$$d(x(j)) \;\leq\; \frac{1}{(\Delta+1)^{t/k}} - \frac{1}{(\Delta+1)^{(t+1)/k}} \;=\; \frac{\Delta}{(\Delta+1)^{(t+1)/k}} \; .$$

Since $\tilde{\delta}(j) \geq (\Delta+1)^{\ell/k}$, and since the increase $d(x(j))$ is distributed equally among the active neighbors of j, the vertex j may contribute an additional value of at most $\frac{d(x(j))}{(\Delta+1)^{\ell/k}} = \frac{\Delta}{(\Delta+1)^{\frac{t+1+\ell}{k}}}$ to the value of the variable $z = z(i)$ of the vertex i. The overall increase in the value of $z(i)$ (that comes from all the active neighbors of i) is consequently at most

$$\frac{\Delta}{(\Delta+1)^{\frac{t+1+\ell}{k}}} \cdot A \;\leq\; \frac{\Delta}{(\Delta+1)^{\ell/k}} \; , \tag{2}$$

where A is the value the variable $A(i)$ of the vertex i in the beginning of iteration t of the inner loop. (By (2), $A(i) \leq (\Delta+1)^{\frac{t+1}{k}}$.)

Combining the inequalities (1) and (2) we get the desired bound of $\frac{1}{(\Delta+1)^{\frac{\ell-1}{k}}}$. ▮

Direct inspection shows that the running time of the algorithm is $O(k^2)$. It is also easy to see that it returns a feasible solution for (LP-MDS). It remains to argue that its approximation guarantee is at most $k(\Delta+1)^{2/k}$.

Theorem 2.2 *[19] The algorithm provides a $(k(\Delta+1)^{2/k})$-approximation guarantee for (LP-MDS).*

Proof: By properties (1) and (3), at the beginning of iteration ℓ of the outer loop, $\tilde{\delta} \leq (\Delta+1)^{\frac{\ell+1}{k}}$ and at the end of this iteration $z \leq (\Delta+1)^{-\frac{\ell-1}{k}}$. Since z can only increase as the algorithm proceeds, it follows that $z \leq (\Delta+1)^{-\frac{\ell-1}{k}}$ during the iteration as well. Hence

$$\sum_{j\in\Gamma(i)} z(j) \leq \tilde{\delta} \cdot \max\{z(j) \mid j \text{ is an active neighbor of } i\} \;\leq (\Delta+1)^{2/k} \; .$$

Set $y = \frac{z}{(\Delta+1)^{2/k}}$, and consider the vector $\underline{y} = (y_1, y_2, \ldots, y_N)$. Note that $\underline{y}$ is feasible for (DLP-MDS) because $\sum_{j\in\Gamma(i)} \frac{z_j}{(\Delta+1)^{2/k}} \leq \frac{1}{(\Delta+1)^{2/k}} \sum_{j\in\Gamma(i)} z_j \leq 1$, for every $i \in V$.

Let OPT be the value of optimal solution for (LP-MDS). Then, by duality of LP,

$$\sum_{i=1}^{N} z_i \;=\; (\Delta+1)^{2/k} \sum_{i=1}^{N} y_i \;\leq\; (\Delta+1)^{2/k} \cdot OPT \; .$$

For $\ell \in \{0, 1, \ldots, k-1\}$, let $Z(\ell) = \sum_{i=1}^{N} z_i$ be the sum of the values of all the variables z at the end of the ℓth iteration of the outer loop. Note that

$$\sum_{i=1}^{N} x_i \;=\; \sum_{\ell=0}^{k-1} Z(\ell) \;\leq\; k \cdot (\Delta+1)^{2/k} \cdot OPT \;. \qquad \blacksquare$$

In [17] Kuhn et al. use a similar technique to devise an $O(\Delta^{1/k})$-approximation algorithm for the minimum vertex cover and maximum (unweighted) matching problems. The running time of their algorithm is $O(k)$. In [20] Kuhn and Wattenhofer generalize all the upper bounds of [19, 17] and devise an efficient distributed approximation algorithm for solving positive LPs. Another distributed approximation algorithm (with somewhat inferior performance) for solving positive LPs was devised by Bartal et al. [1].

3 Lower Bounds

3.1 The *MST* Problem

We start with describing the lower bound of [9] on the approximability of the minimum spanning tree (*MST*) problem. This lower bound is based on the lower bound of Peleg and Rubinovich [29] for the *exact MST* problem.

The lower bound is proven in two stages. The first stage is the lower bound on the complexity of the so-called $CorruptedMail$ problem, which we will define shortly. The second stage is the reduction from the $CorruptedMail$ problem to the approximate *MST* problem. This reduction shows that any algorithm that constructs a spanning tree of weight at most H times greater than the weight of the *MST* requires $T = \Omega\left(\sqrt{\frac{N}{H \cdot B}}\right)$ rounds in the worst case, where B is the bandwidth parameter of the model. The result applies even to graphs of diameter Λ that satisfy $\Lambda \ll T$, and it applies to randomized (both Las-Vegas and Monte-Carlo) algorithms as well. Note that the results can be expressed as a *lower bound on the time-approximation tradeoff*, specifically, $T^2 \cdot H = \Omega(N \cdot B)$.

We next describe the $CorruptedMail$ problem. The problem is defined on graphs $G = (V, E)$ of the following form. Let Γ, m and p be positive integer parameters that satisfy $p \geq \log N$ and $(m+1)\Gamma + \frac{(m+1)^{1+\frac{1}{p}}-1}{(m+1)^{\frac{1}{p}}-1} = N$. Let $d = (m+1)^{1/p}$. (For simplicity, we assume that all the algebraic expressions such as $(m+1)^{\frac{1}{p}}$ and $\frac{(m+1)^{1+\frac{1}{p}}-1}{(m+1)^{\frac{1}{p}}-1}$ are integer.)

The graph G has the following structure. It contains a d-regular tree τ of depth p, with $m+1$ leaves, $z_0, z_1, \ldots, z_m$. These leaves, in turn, are the centers of the stars $S_0, S_1, \ldots, S_m$, and each star S_i contains in addition z_i also the vertices $v_j^{(i)}$, $j = 1, 2, \ldots, \Gamma$, for every index $i = 1, 2, \ldots, m$. The edges of the star S_i are $\{\{z_i, v_j^{(i)}\} \mid j = 1, 2, \ldots, \Gamma\}$. Also, for every index $j = 1, 2, \ldots, \Gamma$, the vertices $v_j^{(i)}$ are connected in such a way that they form a path P_j. Specifically, the edges $\{v_j^{(0)}, v_j^{(1)}\}, \{v_j^{(1)}, v_j^{(2)}\}, \ldots, \{v_j^{(m-1)}, v_j^{(m)}\}$ all belong to the path P_j. See Figure 1.

The leaves z_0 and z_m have special mission. Specifically, the leaf z_0 is also called the *sender*, and is denoted $s = z_0$. The leaf z_m is called the *receiver*, and is denoted by $r = z_m$. The sender s accepts as input a bit string χ, and the goal of the network is to deliver this string to the receiver r. To facilitate the delivery, the network is allowed to *corrupt the message to some extent.* The lower bound that we will show will naturally depend on the extent to which the string can be corrupted. In fact, obviously if the network is allowed to corrupt the string to an arbitrarily large extent,

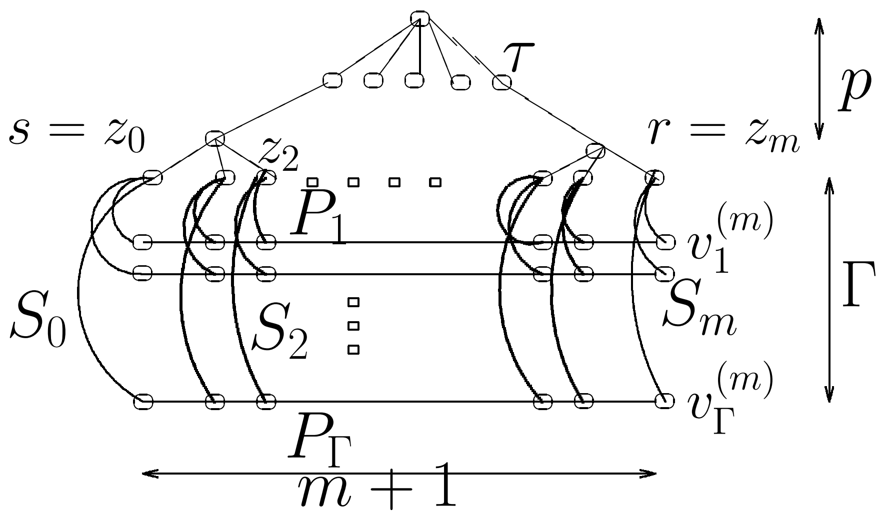

Figure 1: The graph $G = (V, E)$.

then the string can be "delivered" in zero time. (Because in this case the receiver r can return an arbitrary bit string χ' without any communication whatsoever.)

To formally define the *CorruptedMail* problem we need two additional parameters α and β, $0 < \alpha < \beta < 1$, that will be fixed later. For a bit string $\chi \in \{0,1\}^\Gamma$ and an index $j = 1, 2, \ldots, \Gamma$, let χ_j denote the jth bit of χ. The *Hamming weight* of χ, denoted $hwt(\chi)$, is the number of indices $j = 1, 2, \ldots, \Gamma$ such that $\chi_j = 1$. For two bit strings $\chi, \chi' \in \{0,1\}^\Gamma$, the string χ' is said to *dominate* χ if for each $j = 1, 2, \ldots, \Gamma$, $\chi_j = 1$ implies $\chi'_j = 1$. Consider some mapping $\phi : \{0,1\}^\Gamma \to \{0,1\}^\Gamma$, and suppose $\phi(\chi) = \chi'$.

CorruptedMail(α, β) problem is defined on the graph G. The input to this problem is a bit string $\chi \in \{0,1\}^\Gamma$ of length Γ with Hamming weight $hwt(\chi) = \alpha\Gamma$. The input is provided to the sender s only. The output, returned by the vertex r, is a string $\chi' \in \{0,1\}^\Gamma$ of Hamming weight at most $\beta\Gamma$, and it is required that the output string χ' will dominate the input string χ.

Observe that the restriction that χ' should dominate χ guarantees that the errors that are done throughout the delivery of the bit string χ are *one-sided* (i.e., zeroes can become ones, but not vice versa).

Let $\ell(\alpha, \beta)$ denote the expression

$$\ell(\alpha, \beta) \;=\; (\beta - \alpha)\log(\beta - \alpha) - (1 - \alpha)\log(1 - \alpha) - \beta \log \beta \;. \tag{3}$$

The following lemma can be proven by direct calculation.

Lemma 3.1 *For any deterministic protocol Π for the CorruptedMail(α, β) problem, its set of all possible outputs $\{\Pi(\chi) \mid \chi \in \{0,1\}^\Gamma, hwt(\chi) = \alpha\Gamma\}$ contains at least $\Omega(2^{\ell(\alpha,\beta)\Gamma})$ elements.*

We next show that for any protocol Π with worst-case running time at most t for t in certain range, its set of possible outputs (over all possible inputs) is at most exponential in t. (To argue this we use the determinism of the protocol, and the fact that the vertex r that returns the output could not get "too much" information about the input.) It will follow that the running time t of the protocol Π cannot be too small, unless Π is incorrect. To prove an upper bound on the size of the set of possible outputs of any correct protocol, we show that the set of all possible *configurations* of the vertex r in terms of t is relatively small.

Consider some fixed (deterministic) protocol Π, and an execution φ_χ of this protocol on a fixed bit string χ. The configuration of the vertex v on round t of the execution φ_χ, denoted $C(v,t,\chi)$, is the record of all the messages that the vertex v has received on previous rounds. For the sender s, its configuration also contains the input string χ. For a subset $U \subseteq V$ of vertices, its configuration $C(U,t,\chi)$ is the concatenation of the configurations of all the vertices of U (in some fixed order). Let $\mathcal{C}(U,t) = \{C(U,t,\chi) \mid \chi \in \{0,1\}^\Gamma\}$, and $\rho(U,t) = |\mathcal{C}(U,t)|$. In other words, $\rho(U,t)$ is the number of all possible different configurations of the subset U on round t over all possible input strings χ.

Cast to this terminology, our goal is to provide an upper bound in terms of t on $\rho(\{r\},t)$. For this purpose we define a sequence of *tail* sets $T_0, T_1, \ldots, T_m$, that satisfy $V \supseteq T_0 \supseteq T_1 \supseteq \ldots \supseteq T_m \ni r$, and provide an upper bound on $\rho(T_t,t)$, for each $t = 0,1,\ldots,m-1$. Consequently, the same upper bound applies to $\rho(\{r\},t)$.

Recall that the graph G contains as a subgraph the d-regular rooted tree (τ, rt) of height p, with $m+1$ leaves $s = z_0, z_1, \ldots, z_m = r$. For $i = 0,1,2,\ldots,m$, let $\tau(i)$ denote the connected subtree of τ with minimal number of vertices, such that its set of leaves, $Leaves(\tau(i))$, is equal to $\{z_i, z_{i+1}, \ldots, z_m\}$. Let the *root* of $\tau(i)$, denoted $rt(\tau(i))$, be the closest vertex of $\tau(i)$ to the root rt of the tree τ. We define the *tail* sets, $T_0, T_1, \ldots, T_m$ as follows. The tail set T_0 contains the entire vertex set V of G except the vertex s, i.e., $T_0 = V \setminus \{s\}$. For $i = 1,2,\ldots,m$, $T_i = \{v_j^{(i')} \mid j = 1,2,\ldots,\Gamma, i' = i, i+1, \ldots, m\} \cup V(\tau(i))$.

Assume (for convenience) that the rounds are indexed $0,1,2,\ldots$. At the beginning of round $t = 0$, i.e., at the beginning of the execution of the protocol, all the vertices except s are in some fixed initial state, that is independent of the input bit string χ. Hence, $\rho(T_0,0) = \rho(V \setminus \{s\}, 0) = 1$. We next prove an upper bound on the number of configurations of the tail set T_t, after a relatively small number of rounds $t \leq m-1$.

Lemma 3.2 *For* $t = 0,1,\ldots,m-1$, $\rho(T_t,t) \leq (2^{B+1}-1)^{t \cdot p \cdot d}$.

Proof: The proof is by induction on t. The induction base was argued above. We next prove the induction step. Observe that $T_0 \supseteq T_1 \supseteq \ldots \supseteq T_m$. Suppose we are given a configuration $C \in \mathcal{C}(T_i, i)$. Note that C uniquely determines the messages that are sent at round i by vertices of T_i. Therefore, the only messages that are not yet determined are those that are sent by the vertices of $V \setminus T_i$ to the vertices of T_{i+1}. Denote this set of edges by $E_{i,i+1}$. Observe that the edges of the paths $P_1, P_2, \ldots, P_\Gamma$ do not belong to $\bigcup_{i=0}^{m-1} E_{i,i+1}$. It follows that $E_{i,i+1} \subseteq E(\tau)$, for $i = 0,1,\ldots,m-1$. For $i = 0$, clearly, $E_{0,1} = \{\{z_0, w\}\}$, where w is the parent of z_0 in the tree τ. More generally, $E_{i,i+1}$ is a subset of edges of $E(\tau)$ that are incident both to $V(\tau(i+1))$ and to $V \setminus V(\tau(i))$, $i = 0,1,\ldots,m-1$. See Figure 2 for an illustration.

Let Cut_i denote the subset of edges of $E(\tau)$ that are in the cut between $V(\tau(i))$ and the rest of $V(\tau)$, for $i = 1,2,\ldots,m$. It follows that $E_{i,i+1} \subseteq Cut_{i+1}$, for $i = 0,1,\ldots,m-1$. We need an upper bound on the size of this cut.

Lemma 3.3 *For* $i = 1,2,\ldots,m$, $|Cut_i| \leq p \cdot d$.

The (technically not hard) proof of this lemma is omitted (see [9]).

It follows that $|E_{i,i+1}| \leq p \cdot d$ for $i = 0,1,\ldots,m-1$. In other words, for $i = 0,1,\ldots,m-1$, there are at most $p \cdot d$ edges that are incident to the tail set T_{i+1}, such that given a configuration C in $\mathcal{C}(T_i, i)$ of the vertices of T_i at the beginning of round i, the messages that are sent over these edges at round i are not determined by this configuration.

Recall that at most B bits can be delivered through an edge in each round. Therefore, the number of possible messages that may be sent through an edge in a given direction in one round is

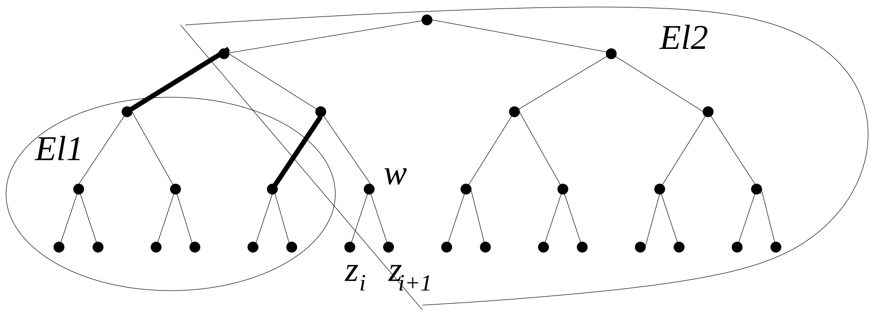

Figure 2: The sets $V(\tau) \setminus V(\tau(i))$ and $V(\tau(i+1))$ are depicted by ellipses $El1$ and $El2$, respectively. Edges of $E_{i,i+1}$ are depicted by thick solid lines. Note that the edge (z_i, w) does not belong to $E_{i,i+1}$, because $z_i \in V(\tau(i))$.

at most $\sum_{\ell=0}^{B} 2^\ell = 2^{B+1} - 1$. Observe that there is only one relevant direction of sending messages in our case, that is, towards the vertices of the tail set T_{i+1}. Hence, the number of possible messages that may be sent through at most $p \cdot d$ edges in one round is at most $(2^{B+1} - 1)^{p \cdot d}$. It follows that for $i = 0, 1, \ldots, m-1$, $\rho(T_{i+1}, i+1) \leq (2^{B+1} - 1)^{p \cdot d} \cdot \rho(T_i, i)$. ▮

Lemma 3.4 *The deterministic complexity of CorruptedMail problem is* $\Omega(\min\{m, \frac{\Gamma}{p \cdot B \cdot m^{1/p}} \cdot \alpha \cdot \log(1/\beta)\})$.

Proof: Consider a protocol Π for the *CorruptedMail* problem, and let t denote its worst-case running time on inputs of fixed size Γ. By Lemma 3.2, it follows that if $t \leq m - 1$ then $\rho(T_t, t) \leq (2^{B+1} - 1)^{t \cdot p \cdot d}$. On the other hand, observe that the number of possible configurations of the vertex r upon the termination of the protocol Π is at least the cardinality of some subset $\Upsilon' \subseteq \Upsilon$ of a subset of bit strings that dominates $\mathcal{A}$. By Lemma 3.1, $|\Upsilon'| = \Omega(2^{\ell(\alpha,\beta)\Gamma})$. It follows that the (worst-case) running time t of the protocol Π for the *CorruptedMail* problem is either at least $t \geq m$ rounds, or it satisfies the the following inequality

$$\Omega(2^{\ell(\alpha,\beta)\Gamma}) = |\Upsilon'| \leq \rho(\{r\}, t) \leq \rho(T_t, t) \leq (2^{B+1} - 1)^{t \cdot p \cdot d} \ . \tag{4}$$

Hence, $\Omega(\ell(\alpha, \beta)\Gamma) - O(1) \leq t \cdot p \cdot d \cdot B$. I.e., $t = \Omega(\ell(\alpha, \beta) \cdot \Gamma/(p \cdot d \cdot B))$. In other words, in both cases $t = \Omega(\min\{m, (\ell(\alpha, \beta) \cdot \Gamma)/(p \cdot d \cdot B)\})$. Recall that $N = O(\Gamma m)$, and $d = (m+1)^{1/p}$. Hence,

$$t = \Omega(\min\{m, \ell(\alpha, \beta) \frac{N}{p \cdot m^{1+1/p} \cdot B}\}) \ . \tag{5}$$

Recall that $\ell(\alpha, \beta)$ is given by (3). For small $\alpha > 0$,

$$\ell(\alpha, \beta) = \alpha \cdot \log(1/\beta) + o(\alpha) \ . \tag{6}$$

(When α does not tend to 0 when N grows to infinity, the inequality (6) may no longer hold, but the inequality $\ell(\alpha, \beta) = \Omega(\alpha \log(1/\beta))$ obviously holds, as both sides are independent of N.) Set β to be a constant between 0 and 1. For α either constant or tending to 0 when N grows, we get $t = \Omega(\min\{m, \frac{N \cdot \alpha}{p \cdot m^{1+1/p} \cdot B}\})$. ▮

Finally, set $m = \left(\frac{N \cdot \alpha}{p \cdot B}\right)^{1/2 - \frac{1}{2(2p+1)}}$. It follows that

$$t = \Omega\left(\left(\frac{N \cdot \alpha}{p \cdot B}\right)^{1/2 - \frac{1}{2(2p+1)}}\right) \ . \tag{7}$$

It follows that any deterministic protocol Π that solves the *CorruptedMail* problem on *every* input bit string χ requires $\Omega\left(\left(\frac{N\cdot\alpha}{p\cdot B}\right)^{1/2-\frac{1}{2(2p+1)}}\right)$ rounds in the worst case.

Furthermore, this statement readily generalizes to randomized protocols using Yao's Minimax theorem [33].

We next describe the reduction from the $CorruptedMail(\alpha,\beta)$ problem to the $\frac{\beta}{\alpha}$-approximate *MST* problem. The protocol Π_{Corr} for the $CorruptedMail(\alpha,\beta)$ problem proceeds in the following way. Given an instance (G,χ), $G \in \mathcal{G}$, $\chi \in \{0,1\}^{\Gamma}$, of the *CorruptedMail* problem, the vertex s computes the weights of edges $\{s, v_j^{(0)}\}$, $j = 1,2,\ldots,\Gamma$, in the following way. If $\chi_j = 0$ then the weight of the edge $\{s, v_j^{(0)}\}$ is set to zero, otherwise it is set to infinity. The weights of the other edges are set as follows. The edges of the paths $P_1, P_2, \ldots, P_\Gamma$, as well as the edges of the tree τ, are all assigned weight zero. The edges of the stars S_i for $i = 1,2,\ldots,m-1$ are all assigned weight infinity. All the edges of the star S_m are assigned unit weight. This setting of weights is performed locally by every vertex, and requires no distributed computation.

Next, the vertices invoke $\frac{\beta}{\alpha}$-approximation protocol Π for the obtained instance $G(\chi)$ of the *MST* problem. Upon the termination of the protocol, each vertex v knows which edges among the edges that are incident to v belong to the approximate *MST* tree τ_0 for $G(\chi)$, that was constructed by the protocol. The vertex r calculates the output bit string $\chi' \in \{0,1\}^{\Gamma}$ in the following way. For each index $j = 1,2,\ldots,\Gamma$, if the edge $\{r, v_j^{(m)}\}$ belongs to the tree τ_0, the vertex r sets $\chi'_j = 1$. Otherwise, it sets $\chi'_j = 0$. Finally, the vertex r returns the bit string χ'.

Observe that whenever the construction of the approximate *MST* tree τ_0 is completed, the computation of the bit string χ' is performed locally by the vertex r, and requires no distributed computation. It follows that the running time of the obtained protocol Π_{Corr} for the $CorruptedMail(\alpha,\beta)$ problem is precisely equal to the running time of the $\frac{\beta}{\alpha}$-approximation protocol Π for the *MST* problem.

The proof of the validity of this reduction is straightforward, and is omitted. Together with the lower bound on the complexity of $CorruptedMail(\alpha,\beta)$ problem it implies the following theorem.

Theorem 3.5 *[9] Any randomized H-approximation protocol for the MST problem on graphs of diameter at most Λ for $\Lambda = 4,6,8,\ldots$ requires $T = \Omega\left(\left(\frac{N}{H\cdot\Lambda\cdot B}\right)^{1/2-\frac{1}{2(\Lambda-1)}}\right)$ rounds of distributed computation. I.e., $T^{2+\frac{2}{\Lambda-2}}\cdot H = \Omega\left(\frac{n}{\Lambda\cdot B}\right)$.*

Substituting $\Lambda = \log\frac{N}{H\cdot B}$ implies the lower bound of $T = \Omega\left(\sqrt{\frac{N}{H\cdot B\cdot\log N}}\right)$. A more delicate argument (see [9]) enables to get rid of the logarithmic factor in the denominator.

3.2 The Minimum Vertex Cover Problem

In this section we describe the lower bound of Kuhn et al. [18] on the approximability of the minimum vertex cover and the minimum dominating set problems. This lower bound states that $(\Delta^{1/k}/k)$-approximation for either of these problems require $\Omega(k)$ rounds. Furthermore, $(N^{1/k^2}/k)$-approximation for either of these problems requires $\Omega(k)$ rounds as well. Unlike the lower bound of [9] for the *MST* problem that was discussed in the previous section, these lower bounds are meaningful even when the bandwidth parameter B is equal to infinity.

Consider the following recursive construction of the rooted edge-labeled tree τ_k. The labels of the edges belong to the set $\{0,1,\ldots,k\}$. Let $\tau_1 = \{V_1, E_1\}$, $V_1 = \{v_0, v_1, v_2, v_3\}$, $E_1 =$

$\{\{v_0, v_1\}, \{v_0, v_2\}, \{v_1, v_3\}\}$, $\ell(\{v_0, v_1\}) = \ell(\{v_1, v_2\}) = 0$, $\ell(\{v_0, v_3\}) = 1$. The vertex v_0 is considered to be the root of τ_1.

A vertex of degree 1 is called a *leaf*; a non-leaf vertex is called an *internal* vertex.

Given τ_{k-1} the tree τ_k is constructed by:

1. For each internal vertex v, add a new vertex w, connect v to w, and label the new edge $\{v, w\}$ by k.

2. For each leaf z of τ_{k-1} whose only incident edge is labeled by p, add k new vertices z'_j, $j \in \{0, 1, \ldots, k\} \setminus \{p + 1\}$, connect z to each z'_j, and label each edge $\{z, z'_j\}$ by j.

3. The root of τ_k is the root of τ_{k-1}.

See Figure 3 for an illustration of τ_2 and τ_3.

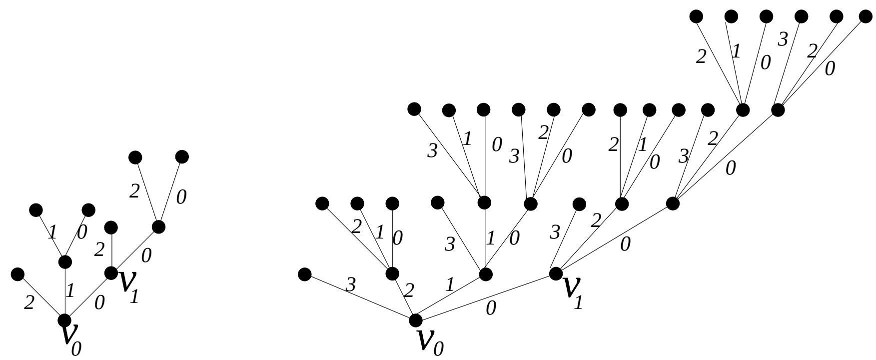

Figure 3: The trees τ_2 and τ_3.

We now form the graph CT_k based on $\tau_k = (V_k, E_k)$ in the following way. Let δ be a positive integer parameter. Replace each vertex $v \in V_k$ by a subset $C(v)$ of vertices. The subsets $C(v)$ will be also called *clusters*. The size of $C(v)$ is a function of δ, and of the location of v in τ_k, and will be determined shortly. Consider an edge $\{v, w\} \in E_k$. Let $p = \ell(e) \in \{0, 1, \ldots, k\}$ be the label of the edge $\{v, w\}$. The analogue of the edge $e = \{v, w\}$ in the graph CT_k is the bipartite graph $G(e) = (C(v), C(w), C(e))$. The degree of each vertex $x \in C(v)$ will be δ^p, and the degree of each vertex $y \in C(w)$ will be δ^{p+1}.

To complete the description of the graph CT_k we need only to specify the sizes of the clusters $C(v)$ for each vertex $v \in V_k$, and to argue that for each edge $e \in E_k$, the bipartite left- and right-regular graph $E(e)$ with left degree $\delta^{\ell(e)}$ and right degree $\delta^{\ell(e)+1}$ exists.

Observe that the *depth* of τ_k, that is, the maximum (unweighted) distance between the root to a leaf, is $k + 1$. For $j = 0, 1, \ldots, k + 1$, the *jth level* of τ_k is the set of vertices at distance j from the root. Consider some leaf z of τ_k. By construction, the label $p = \ell(e_z)$ of the edge $e_z = \{w, z\}$ that is incident to z is at most k. Consequently, the degrees of the vertices of $C(w)$ in the bipartite graph $C(e_z)$ is $\delta^p \leq \delta^k$. Hence we need to guarantee that $|C(z)| \geq \delta^k$. We set the sizes of clusters of level $k + 1$ to δ^k.

Observe that for any edge $\{u, w\} \in E_k$ of τ_k with u being the *parent* of w in the tree, the degree of the vertices of $C(u)$ is set to be smaller by a factor of δ than the degree of the vertices of $C(w)$. Consequently, $|C(u)| = \delta \cdot |C(w)|$. In other words, setting the sizes of the leaf-clusters of level $k + 1$ determines uniquely the sizes of all the other clusters of CT_k. Particularly, the size of the cluster

$C(v_0)$ that corresponds to the root v_0 is δ^{2k+1}, and the size of the cluster $C(v_1)$ (that corresponds to the vertex v_1 such that $\ell(\{v_0, v_1\}) = 0$) is δ^{2k}.

Consider an edge $e = \{u, w\} \in E_k$ such that u is the parent of w. Let $h = 0, 1, \ldots, k$ be the level of the parent u. Note that $|C(u)| = \delta^{2k+1-h}$, $|C(w)| = \delta^{2k-h}$. Let $p = \ell(e) \in \{0, 1, \ldots, k\}$ be the label of e. To construct a bipartite left- and right-regular graph $G(e)$ with δ^{2k+1-h} vertices on the left-hand side and δ^{2k-h} vertices on the right-hand side, and with left degree δ^p and right degree δ^{p+1}, just insert δ^{2k-h-p} complete bipartite graphs with δ^{p+1} vertices on the left-hand side and δ^p on the right-hand side. Note that assuming that $\delta > k+1$, the number of vertices N in the graph CT_k is smaller than δ^{2k+2}. Its maximum degree is $\Delta = \delta^{k+1}$.

Let τ'_k be the tree τ_k with each edge $e = \{u, w\}$ replaced by two arcs (u, w) and (w, u). The arc (u, w) retains the label $\ell(e)$ of the edge e, and the arc (w, u) is assigned the label $\ell(e) + 1$. For a vertex v, let $\Gamma_k(v)$ be the arborescence of depth k rooted in v formed in the following way. Its vertex set is the set of all (not necessarily simple) paths in τ_k of length at most k that start in v. There is an arc from a path $P_1 = (v = x_0, x_1, \ldots, x_h)$ to a path $P_2 = (v = x_0, x_1, \ldots, x_h, x_{h+1})$ if and only if there is an arc (x_h, x_{h+1}) in τ_k. See Figure 4 for illustration of $\Gamma_1(v_0)$ and $\Gamma_2(v_0)$.

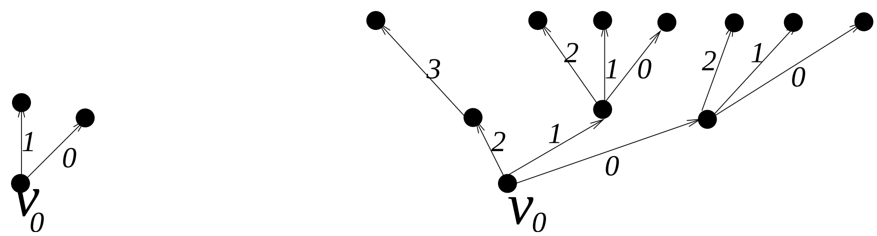

Figure 4: The neighborhoods $\Gamma_1(v_0)$ and $\Gamma_2(v_0)$.

It is not hard to verify that the arborescences $\Gamma_k(v_0)$ and $\Gamma_k(v_1)$ are identical. Intuitively, this observation suggests that the k-neighborhoods of vertices of $C(v_0)$ and $C(v_1)$ in CT_k should be identical as well. This, however, is not obvious, as each arc of τ_k is implemented in CT_k by a dense bipartite graph. To ensure that the k-neighborhoods of the vertices of $C(v_0)$ and $C(v_1)$ will have tree-like structure (and, actually, stronger than that: they all will be isomorphic to $\Gamma_k(v_0) = \Gamma_k(v_1)$), Kuhn et al. [18] use the so-called *Lazebnik-Ustimenko* (henceforth, LU) *transformation* (see [22]). This transformation, applied to CT_k converts it into a graph G_k with the same cluster structure that satisfies $girth(G_k) \geq 2k+1$.

In other words, for each cluster $C(v)$ of CT_k, G_k has a cluster $C'(v)$. For every left- and right-regular bipartite graph $G(e) = (C(v), C(w), C(e))$, $e = \{v, w\} \in E(\tau_k)$, the graph G_k has a left- and right-regular bipartite graph $(C'(v), C'(w), C'(e))$ that has the same left and right degrees as the graph $G(e)$. However, the "price" for the magical increase of girth is that the number of vertices grows exponentially in k. Specifically, the overall number N of vertices in the graph G_k obtained after this transformation is at most $2^{O(k)}\delta^{O(k^2)}$. For the details of LU transformation and its analysis see [22, 18].

Given the girth property it can now be shown that the k-neighborhoods of the vertices of the set $C'(v_0) \cup C'(v_1)$ are all isomorphic to $\Gamma_k(v_0)$ (and to $\Gamma_k(v_1)$). The proof of this claim is not hard, but quite technical, and is omitted (see [18]).

Observe that the behavior of a distributed algorithm on a graph G may depend not only on the graph itself, but rather also on the assignment of identities, henceforth referred as the *id-assignment*, to the vertices of the graph G. We now argue that for every deterministic algorithm A with running time at most k there is an id-assignment to the vertices of G_k so that either the algorithm does not construct a valid vertex cover for G_k, or the size of this vertex cover is more than $\delta/(2k)$ times greater than the size of the minimum one.

For an algorithm A and an id-assignment I to the vertices of G_k, let $VC(A,I)$ denote the vertex cover returned by the algorithm for this instance. Suppose that the id-assignment I is chosen uniformly at random, and consider the random variable $X = |VC(A,I) \cap C'(v_0)|$. Let $x_0 \in C'(v_0)$, $x_1 \in C'(v_1)$ be two adjacent vertices of the graph G_k. For any id-assignment I, either x_0 or x_1 must end up in the vertex cover. Consequently,

$$\mathbb{E}_I(x_0 \in VC(A,I)) + \mathbb{E}_I(x_1 \in VC(A,I)) \;\geq\; 1 \;.$$

Also, since the k-neighborhoods of x_0 and x_1 are isomorphic, it follows that $\mathbb{E}_I(x_0 \in VC(A,I)) = \mathbb{E}_I(x_1 \in VC(A,I)) \geq 1/2$. Hence

$$\mathbb{E}_I(X) \;=\; \sum_{x_0 \in C'(v_0)} \mathbb{E}(x_0 \in VC(A,I) \;\geq\; |C'(v_0)|/2 \;.$$

Hence there exists an id-assignment I to the vertices of G_k for which the algorithm A takes into the vertex cover one half or more vertices of $C'(v_0)$. Observe that $V(G_k) \setminus C'(v_0)$ is a vertex cover as well, and its size is more than by a factor $\delta/2k$ smaller than the size of $C'(v_0)$. Consequently, the vertex cover provided by the algorithm A on the instance (G_k, I) is not a $(\delta/(2k))$-approximation of the minimum vertex cover.

Simple calculation now implies the main theorem of this section.

Theorem 3.6 *[18] For every deterministic algorithm with running time k there exist instances of vertex cover with N vertices and maximum degree Δ on which the approximation guarantee of A is at least $\frac{N^{\Omega(1/k^2)}}{k}$ and $\Omega(\frac{\Delta^{1/k}}{k})$.*

It is not hard to see that the same argument generalizes to *randomized algorithms*. Furthermore, the same result applies to the *minimum dominating set* problem as well. This can be shown via a simple reduction from the minimum vertex cover problem. Kuhn et al. [17] have also applied this technique to the maximum matching problem, and derived analogous results.

4 Open Problems

The distributed approximability of many important problems, such as minimum and maximum cut and bisection, maximum independent set, maximum clique, minimum coloring, and other, remains unresolved. There is no satisfactory approximation algorithm for the *MST* problem. There is a significant gap between the state-of-the-art upper and lower bounds for the minimum dominating set, minimum vertex cover, and maximum matching problems.

No less interesting direction is to divide the distributed approximation problems into complexity classes, and to establish links between those classes and the classical complexity classes.

The author also wishes to draw attention to two problems in the message-passing model. Though these problems do not belong to the realm of distributed approximation, but rather have to do with the classical distributed computing, the author believes that their resolution would be instrumental for distributed approximation as well.

1. Consider a complete network of N processors in the message-passing model. Each edge of this clique is assigned a weight, but the weights do not affect the communication (i.e.,

delivering a short message through an edge requires one round independently of the weight of the edge). The bandwidth parameter B is $\log N$. Prove a non-trivial ($\omega(1)$) lower bound on the complexity of the single-source distance computation in this model. A lower bound on the all-pairs-shortest-paths problem in this model will be most interesting as well.

2. (This problem was suggested by Vitaly Rubinovich in a private conversation with the author few years ago.)

   What is the complexity of the shortest-path-tree problem in the message-passing model with small bandwidth parameter B (i.e., $B = \log N$)?

   A lower bound of $\Omega(\sqrt{\frac{N}{B}})$ [29, 9] is known. No non-trivial upper bound is currently known. (The trivial one is $O(N)$.)

References

[1] Y. Bartal, J. W. Byers, and D. Raz. Global optimization using local information with applications to flow control. In *Proc. of the 38th IEEE Symp. on the Foundations of Computer Science*, pages 303–312, 1997.

[2] A. Czygrinow, M. Hanckowiak, and M. Karonski. Distributed o(delta log n)-edge-coloring algorithm. In *Proc. 9th Annual European Symp. on Algorithms*, pages 345–355, 2001.

[3] A. Czygrinow, M. Hanckowiak, and E. Szymanska. Distributed algorithm for approximating the maximum matching, manuscript. 2003.

[4] G. de Marco and A. Pelc. Near-optimal distributed edge coloring. In *Proc. 3rd Annual European Symp. on Algorithms*, pages 448–459, 1995.

[5] E. W. Dijkstra. Self stabilizing systems in spite of distributed control. *Comm. of the ACM*, 17:643–644, 1974.

[6] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivisan. Fast distributed algorithm for (weakly) connected dominating sets and linear-size skeletons, 2003.

[7] F. Eisenbrand, S. Funke, N. Garg, and J. Konemann. A combinatorial algorithm for computing a maximum independent set in a t-perfect gaph. In *Proc. of the ACM-SIAM Symp. on Discrete Algorithms*, pages 517–522, 2003.

[8] M. Elkin. Computing almost shortest paths. In *Proc. 20th ACM Symp. on Principles of Distributed Computing*, pages 53–62, 2001.

[9] M. Elkin. An unconditional lower bound on the time-approximation tradeoff of the minimum spanning tree problem. In *Proc. of the 36th ACM Symp. on Theory of Comput. (STOC 2004)*, pages 331–340, 2004.

[10] M. Elkin and J. Zhang. Efficient algorithms for constructing $(1 + \epsilon, \beta)$-spanners in the distributed and streaming models. In *Proc. 23rd ACM Symp. on Principles of Distributed Computing (to appear)*, 2004.

[11] L. Fleisher. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.

[12] L. Fleisher. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proc. of the ACM-SIAM Symp. on Discrete Algorithms*, pages 1001–1010, 2004.

[13] R. G. Gallager. A shortest path routing algorithm with automatic resynch. *Technical Report LIDS-P-1175*, 1976.

[14] D. Grable and A. Panconesi. Nearly optimal distributed edge colouring in $o(loglogn)$ rounds. *Random Structures and Algorithms*, 10(3):385–405, 1997.

[15] D. Hochbaum. *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1996.

[16] L. Jia, R. Rajaraman, and R. Suel. An efficient distributed algorithm for constructing small dominating sets. In *Proc. of the 20th ACM Symp. on Principles of Distrib. Comput. (PODC)*, pages 33–42, 2001.

[17] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Lower and upper bounds for distributed packing and covering. *Technical Report 443, Dept. of Computer Science, ETH, Zurich*, 2004.

[18] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! *Technical Report 438, Dept. of Computer Science, ETH, Zurich*, 2004.

[19] F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. In *Proc. of the 22nd ACM Symp. on Principles of Distributed Computing*, 2003.

[20] F. Kuhn and R. Wattenhofer. Distributed combinatorial optimization. *Technical Report 426, Dept. of Computer Science, ETH, Zurich*, 2004.

[21] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge Press, 1996.

[22] F. Lazebnik and V. A. Ustimenko. Explicit construction of graphs with an arbitrary large girth and of large size. *Discrete Applied Mathematics*, 60(1-3):275–284, 1995.

[23] G. LeLann. Distributed systems, towards a formal approach. In *IFIP Congress*, pages 155–160, 1977.

[24] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21:193–201, 1992.

[25] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. of the 25th ACM Symp. on Theory of Computing*, pages 448–457, 1993.

[26] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM J. on Computing*, 26(2):350–368, 1997.

[27] C. Papadimitriou and M. Yannakakis. Linear programming without matrix. In *Proc. of the 25th ACM Symp. on Theory of Computing*, pages 121–129, 1993.

[28] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[29] D. Peleg and V. Rubinovich. A near-tight lower bound on the time complexity of distributed mst construction. In *Proc. 40th IEEE Symp. on Foundations of Computer Science*, pages 253–261, 1999.

[30] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics and Operations Research*, 20:257–301, 1995.

[31] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.

[32] M. Wattenhofer and R. Wattenhofer. Distributed weighted matching. *Technical Report 420, ETH, Zurich, Department of Computer Science*, 2003.

[33] A. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 17th IEEE Symp. on Foundations of Computer Science*, pages 222–227, 1977.