Survey of Local Algorithms

JUKKA SUOMELA, University of Helsinki and Helsinki Institute for Information Technology HIIT

A local algorithm is a distributed algorithm that runs in constant time, independently of the size of the network. Being highly scalable and fault tolerant, such algorithms are ideal in the operation of large-scale distributed systems. Furthermore, even though the model of local algorithms is very limited, in recent years we have seen many positive results for nontrivial problems. This work surveys the state-of-the-art in the field, covering impossibility results, deterministic local algorithms, randomized local algorithms, and local algorithms for geometric graphs.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems; F.1.1 [**Computation by Abstract Devices**]: Models of Computation; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Local algorithms

ACM Reference Format:
Suomela, J. 2013. Survey of local algorithms. ACM Comput. Surv. 45, 2, Article 24 (February 2013), 40 pages.
DOI = 10.1145/2431211.2431223 http://doi.acm.org/10.1145/2431211.2431223

1. INTRODUCTION

The running time of an algorithm typically increases with the size of the input; sorting one thousand names takes longer than sorting one hundred names. Only the most trivial problems can be solved in constant time, independently of the size of the input. There is one notable exception, though: there are nontrivial *distributed algorithms* that run in constant time.

In a distributed algorithm, the same computer network is both the input and the system that solves the problem; hence a larger input also implies a larger number of parallel computers. This does not make problems trivial to solve fast, but it turns out that there are several examples of computational problems that can be solved by using a constant-time distributed algorithm. Such algorithms are known as *local algorithms*.

1.1. Local Algorithms

A local algorithm is a distributed algorithm that runs in a constant number of synchronous communication rounds, independently of the number of nodes in the network.

A preliminary version of this survey was published as part of the author's Ph.D. thesis [Suomela 2009, Section 2]. Updates to this work will be posted online at http://www.cs.helsinki.fi/local-survey/.
This work was supported in part by the Academy of Finland, grants 116547, 132380, and 252018, by the Helsinki Graduate School in Computer Science and Engineering (Hecse), by the Foundation of Nokia Corporation, by the Finnish Cultural Foundation, and by Research Funds of the University of Helsinki.
Authors' address: J. Suomela, Helsinki Institute for Information Technology HIIT, University of Helsinki, P.O. Box 68, FI-00014 University of Helsinki, Finland; email: jukka.suomela@cs.helsinki.fi.

DOI 10.1145/2431211.2431223 http://doi.acm.org/10.1145/2431211.2431223

Table I. Problems that Cannot Be Solved with a Deterministic Local Algorithm, even If There Are Unique Node Identifiers

Problem	Graph family	References
Maximal independent set	cycle	Linial [1992]
Maximal matching	cycle	Linial [1992] cor.
Vertex 3-coloring	cycle	Linial [1992]
Vertex Δ-coloring	$(\Delta+1)$-colored tree	Kuhn [2005]
Edge coloring	cycle	Linial [1992] cor.
Weak coloring	$2k$-regular	Naor and Stockmeyer [1995]

The problems are defined in Section 4
cor. = corollary, see text.

Put otherwise, the output of a node in a local algorithm is a function of the input available within a constant-radius neighborhood of the node.

Research on local algorithms was pioneered by Angluin [1980], Linial [1992], and Naor and Stockmeyer [1995]. Angluin [1980] studied the limitations of anonymous networks without any unique identifiers. Linial [1992] proved seminal negative results for the case where each node has a unique identifier. Naor and Stockmeyer [1995] presented the first nontrivial positive results.

This work is a survey of local algorithms. We focus on algorithms whose running time and performance guarantees are independent of the number of nodes in the network; put simply, these are algorithms that could be used to control infinitely large networks in finite time. For a more general discussion on distributed algorithms, see, for example, Peleg [2000] and Elkin [2004].

Many of the negative results cited in this survey were not originally stated as negative results for local algorithms; they are more general results which, as a corollary, imply that a particular problem cannot be solved by any local algorithm. The emphasis is on results that have nontrivial implications for local algorithms. Further impossibility results for distributed algorithms are presented in the surveys by Lynch [1989] and Fich and Ruppert [2003].

1.2. Structure of This Work

We begin with some essential definitions in Section 2. Section 3 reviews the advantages and applications of local algorithms. Section 4 introduces the computational problems that we use as examples throughout this work, and Section 5 discusses what information each node has available in a local algorithm. Section 6 reviews negative results: what cannot be computed with a local algorithm. Section 7 reviews positive results: which deterministic local algorithms are known. In Section 8 we study the power of randomness, in comparison with deterministic local algorithms. Section 9 studies local algorithms in a geometric setting, in which each node knows its coordinates. Section 10 concludes this survey with some open problems.

For a quick summary of the negative results for deterministic local algorithms, see Tables I and II. The positive results for deterministic local algorithms are summarized in Tables III and IV. Many of the results summarized in the tables are corollaries that have not been stated explicitly in the literature.

2. DEFINITIONS

In this survey, all graphs are simple and undirected unless otherwise mentioned. Terminology related to directed graphs is introduced in Section 5.4, and geometric graphs such as unit-disk graphs are defined in Section 9.1.

Table II. Approximation Factors that Cannot Be Achieved with a Deterministic Local Algorithm, even If There Are Unique Node Identifiers

Problem	Approx. factor	Graph family	References
Independent set	$O(1)$	cycle	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011]
Matching	$O(1)$	general	Kuhn [2005], Kuhn et al. [2006b], Moscibroda [2006]
	$O(1)$	cycle	Czygrinow et al. [2008]
Edge cover	$2-\varepsilon$	cycle	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011] cor.
Vertex cover	$O(1)$	general	Kuhn [2005], Kuhn et al. [2004, 2010], Moscibroda [2006]
	$2-\varepsilon$	cycle	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011] cor.
Dominating set	$O(1)$	general	Kuhn [2005], Kuhn et al. [2004, 2010], Moscibroda [2006]
	$O(1)$	unit-disk	Lenzen and Wattenhofer [2008], Lenzen [2011]
	$2k+1-\varepsilon$	$2k$-regular	Czygrinow et al. [2008] cor.
	$k+1-\varepsilon$	$(2k+1)$-regular, 2-C	Åstrand et al. [2010]
	$5-\varepsilon$	4-regular, planar	Czygrinow et al. [2008]
	$3-\varepsilon$	cycle	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011]
Domatic partition	$3-\varepsilon$	cycle	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011] cor.
Edge domin. set	$3-\varepsilon$	cycle	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011] cor.
Maximum cut	$O(1)$	cycle	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011] cor.
Set cover	$k-\varepsilon$	k-regular	Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], Lenzen [2011], Åstrand and Suomela [2010]
0/1 packing LP	$O(1)$	general	Kuhn [2005], Kuhn et al. [2006b], Moscibroda [2006]
0/1 covering LP	$O(1)$	general	Kuhn [2005], Kuhn et al. [2004, 2010], Moscibroda [2006]
0/1 max-min LP	$\Delta_I(1-1/\Delta_K)$	bounded-degree	Floréen et al. [2008b, 2008c, 2011]

$\varepsilon > 0$.
2-C = bicolored graphs, i.e., a 2-coloring is given.
cor. = corollary, see text.

2.1. Graphs

For a graph $\mathcal{G} = (V, E)$, we use the following notation and terminology. An undirected edge between nodes $u \in V$ and $v \in V$ is represented by an unordered pair $\{u, v\} \in E$.

Table III. Deterministic Local Algorithms

Problem	Graph family	Model	References
Maximal matching	bicolored, B-D	P	Hańćkowiak et al. [1998]
ε-stable matching	bicolored, B-D	P	Floréen et al. [2010]
Vertex $(\Delta+1)$-coloring	k-colored, B-D	P	Barenboim and Elkin [2009], Cole and Vishkin [1986], Goldberg et al. [1988], Kuhn [2009]
Weak coloring	odd degree, B-D	P+O	Mayer et al. [1995], Naor and Stockmeyer [1995]

B-D = bounded-degree graph.
P = algorithm uses only a port numbering.
P+O = algorithm uses only a port numbering and an orientation.
The problems are defined in Section 4.

We write $\deg(v)$ for the degree (number of neighbors) of node $v \in V$. A node $v \in V$ is isolated if $\deg(v) = 0$.

Graph $\mathcal{G}$ is k-regular if $\deg(v) = k$ for each $v \in V$. Graph $\mathcal{G}$ is bipartite if $V = V_1 \cup V_2$ for disjoint sets V_1 and V_2 such that each edge $e \in E$ is of the form $e = \{u, v\}$ for $u \in V_1$ and $v \in V_2$. The complete graph on n nodes is denoted by K_n.

2.2. Neighborhoods

We use $d_{\mathcal{G}}(u, v)$ to denote the shortest-path distance (number of edges, hop count) between nodes u and v in graph $\mathcal{G}$, and

$$B_{\mathcal{G}}(v, r) = \{u \in V : d_{\mathcal{G}}(u, v) \leq r\}$$

to denote the radius-r neighborhood of node v in graph $\mathcal{G}$. We write $\mathcal{G}(v, r)$ for the subgraph of $\mathcal{G}$ induced by $B_{\mathcal{G}}(v, r)$. We occasionally refer to the subgraph $\mathcal{G}[v, r]$ of $\mathcal{G}(v, r)$; graph $\mathcal{G}[v, r]$ is constructed from $\mathcal{G}(v, r)$ by removing the edges $\{s, t\}$ with $d_{\mathcal{G}}(v, s) = d_{\mathcal{G}}(v, t) = r$.

2.3. Communication Graph

Throughout this work, graph $\mathcal{G} = (V, E)$ is the communication graph of a distributed system: each node $v \in V$ is a computational entity and an edge $\{u, v\} \in E$ denotes that nodes u and v can communicate with each other.

Often we have to make assumptions on the structure of the communication graph. Among others, we study the family of *bounded-degree graphs*. In this case we assume that there is a known constant Δ, and any node in any communication graph $\mathcal{G}$ that we may encounter is guaranteed to have at most Δ neighbors.

2.4. Port Numbering

We assume that there is a port numbering (local edge labeling) [Angluin 1980; Attiya et al. 1988; Yamashita and Kameda 1996] available for the communication graph $\mathcal{G}$. This means that each node of $\mathcal{G}$ imposes an ordering on its incident edges. Thus each edge $\{u, v\} \in E$ has two natural numbers associated with it: the port number at node u, denoted by $p(u, v)$, and the port number at node v, denoted by $p(v, u)$. If $p(u, v) = i$, we also say that the neighbor i of u is v.

See Figure 1 for an illustration. Figure 1(a) shows one possible way to assign the port numbers in a 3-cycle. The port number at node u for edge $e = \{u, v\} \in E$ is $p(u, v) = 2$, and the port number at node v for edge e is $p(v, u) = 1$. The neighbor 2 of u is v. Figure 1(b) shows another way to assign the port numbers in the same graph.

Table IV. Deterministic Local Approximation Algorithms

Problem	Approx. factor		Graph family	Model	References
Matching	$1+\varepsilon$	$*$	2-C, B-D	P	Hańćkowiak et al. [1998], Åstrand et al. [2010]
	$(\Delta+1)/2$	$*$	W-C, B-D	P	Åstrand et al. [2010]
Weighted matching	$2+\varepsilon$		2-C, B-D	P	Floréen et al. [2010]
Simple 2-matching	$2+\varepsilon$		B-D	P	Åstrand et al. [2010], Polishchuk and Suomela [2009] cor.
Semi-matching	$O(1)$		2-C, B-D	P	Czygrinow et al. [2011]
Edge cover	2	$*$	general	P	
Vertex cover	2	$*$	regular	P	
	6		unit-disk	P	Kuhn [2005], Wiese and Kranakis [2008a]
	$4+\varepsilon$		B-D		Moscibroda [2006]
	3		B-D	P	Polishchuk and Suomela [2009]
	$2+\varepsilon$		B-D		Kuhn [2005], Kuhn et al. [2006b]
	2	$*$	B-D	P	Åstrand et al. [2009], Åstrand and Suomela [2010]
Dominating set	$\Delta+1$		B-D	P	
	$2\lfloor\Delta/2\rfloor+1$	$*$	B-D	P+O	Åstrand et al. [2010]
	$(\Delta+1)/2$	$*$	W-C, B-D	P	Åstrand et al. [2010]
	$O(1)$		planar		Czygrinow et al. [2008], Lenzen et al. [2010], Lenzen [2011]
	$O(A\log\Delta)$		B-A, B-D	P	Lenzen and Wattenhofer [2010], Lenzen [2011]
Domatic partition	$(\delta+1)/2$		W-C, B-D	P	
Edge domin. set	$4-2/\Delta$		B-D	P	Suomela [2010]
Maximum cut	Δ		W-C, B-D	P	
Set cover	Δ_V	$*$	B-D	P	
	$\Delta_K+\varepsilon$		B-D		Kuhn [2005], Kuhn et al. [2006b]
	Δ_K	$*$	B-D	P	Åstrand and Suomela [2010]
Packing LP	Δ_I		B-D	P	Papadimitriou and Yannakakis [1993]
0/1 packing LP	$1+\varepsilon$	$*$	B-D		Kuhn [2005; Kuhn et al. [2006b]
0/1 covering LP	$1+\varepsilon$	$*$	B-D		Kuhn [2005], Kuhn et al. [2006b]
Max-min LP	Δ_I		B-D	P	Papadimitriou and Yannakakis [1993] cor.
	$\alpha+\varepsilon$	$*$	B-D	P	Floréen et al. [2008a, 2008b, 2008c, 2009, 2011]

The algorithms without references are trivial; see text.
$\varepsilon > 0$.
$\alpha = \Delta_I(1-1/\Delta_K)$.
δ = minimum degree of $\mathcal{G}$.
$*$ = tight approximation ratio (matching negative result).
B-D = graphs with node degrees at most Δ.
B-A = graphs with arboricity [Diestel 2005, Section 2.4] at most A.
2-C = bicolored graphs, i.e., a 2-coloring is given.
W-C = a weak 2-coloring is given or can be found locally; includes graphs where every node has an odd degree [Mayer et al. 1995; Naor and Stockmeyer 1995].
P = algorithm uses only a port numbering.
P+O = algorithm uses only a port numbering and an orientation.
cor. = corollary, see text.

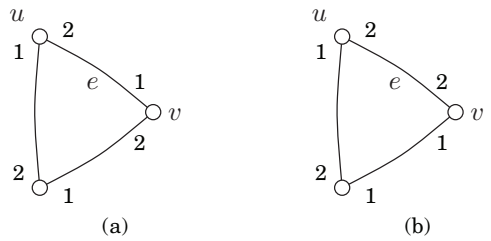

Fig. 1. A communication graph with a port numbering.

2.5. Model of Distributed Computing

We use Linial's [1992] model of computation; Peleg [2000] calls it the *local* model. Each node in the system executes the same algorithm $\mathcal{A}$. Initially, each node $v \in V$ knows a task-specific *local input* i_v. Each node $v \in V$ has to produce a *local output* o_v. We always assume that $\deg(v)$ is part of the local input i_v. The local input i_v may also contain auxiliary information such as unique node identifiers; this is discussed in more detail in Section 5.

The distributed system operates in a synchronous manner. Let r be the number of synchronous communication rounds. In each round $i = 1, 2, \ldots, r$, the following operations are performed, in this order:

(1) Each node performs local computation.
(2) Each node v sends one message to each port $1, 2, \ldots, \deg(v)$.
(3) Each node v receives one message from each port $1, 2, \ldots, \deg(v)$.

Finally, after round r, each node $v \in V$ performs local computation and announces its local output o_v. The size of a message is unbounded and local computation is free.

Example 2.1. Consider the graph in Figure 1(a). If node u sends a message m to port 2 in round i, the same message m is received by node v from port 1 in the same round i. Note that node u can include the outgoing port number 2 in message m; then the receiver v learns that the edge number 1 at v equals the edge number 2 at its neighbor u.

2.6. Local Algorithm and Local Horizon

We say that $\mathcal{A}$ is a *local algorithm* if the number of communication rounds r is a constant. The constant r may depend on the parameters of the problem family; for example, if we study bounded-degree graphs, the value of r may depend on the parameter Δ. However, the value of r cannot depend on the problem instance; in particular, it does not depend on the number of nodes in graph $\mathcal{G}$.

The constant r is called the *local horizon* of the local algorithm. In r synchronous communication rounds, information can be propagated for at most r hops in the network. The output o_v of a node $v \in V$ may depend on the local inputs i_u for all $u \in B_{\mathcal{G}}(v, r)$; however, it cannot depend on the local input i_u for any $u \notin B_{\mathcal{G}}(v, r)$. A decision must be made based on the information that is available within the local horizon.

Throughout this work, the original definition of a local algorithm by Naor and Stockmeyer [1995] is used: r must be a constant. In many papers, the term "local algorithm" is used in a less strict manner, and terminology such as "strictly local algorithm" or "$O(1)$-local algorithm" is used to refer to the case of a constant r.

2.7. Local Approximation

An *α-approximation algorithm* is an algorithm that produces a feasible output, and the utility of the output is guaranteed to be within factor α of the utility of an optimal solution. We use the convention that $\alpha \geq 1$ for both minimization and maximization problems [Ausiello et al. 2003]. Hence, for a minimization problem, an α-approximation algorithm produces a feasible solution with a cost of at most $\alpha \cdot \mathrm{OPT}$, where OPT is the cost of an optimal solution, and for a maximization problem, an α-approximation algorithm produces a feasible solution with a utility of at least OPT/α, where OPT is the utility of an optimal solution.

A *local α-approximation algorithm* is an α-approximation algorithm and a local algorithm. A *local approximation scheme* is a family of local algorithms such that for each $\varepsilon > 0$ there is a local $(1 + \varepsilon)$-approximation algorithm.

2.8. Distributed Constant-Size Problem

We say that a problem is of *distributed constant size* if $\mathcal{G}$ is a bounded-degree graph and the size of the local input i_v is bounded by a constant. If we have a local algorithm for a distributed constant-size problem, then each node needs to transmit and process only a constant number of bits; therefore local computations can be done in constant time, and the size of the local output o_v is bounded by a constant as well. Informally, a local algorithm for a distributed constant-size problem runs in constant time, regardless of the details of the model of distributed computing; we do not need to exploit the unbounded size of messages and unlimited local computation.

3. ADVANTAGES AND APPLICATIONS

The problems that we study in this survey are typically inspired by applications related to the operation of communication networks: for example, monitoring communication networks, scheduling the activities of the nodes in a network, or routing data in a network. We will discuss the problems in more detail in Section 4, but let us first study *why* it is advantageous to use local algorithms in such applications. In addition to discussing the practical uses of local algorithms in communication networks, we will also explore connections between local algorithms and other fields of computer science.

3.1. Fault Tolerance and Robustness

A local algorithm is not only highly scalable but also fault tolerant. A local algorithm recovers efficiently from failures, changes in the network topology, and changes in the input [Naor and Stockmeyer 1995]. If the input of a node $v \in V$ changes, this only affects the output within $B_{\mathcal{G}}(v, r)$.

In particular, a local algorithm can be used to maintain a feasible solution in a *dynamic graph* in which edges and nodes are added and deleted [Eppstein et al. 1999]. In the case of distributed constant-size problems, a local algorithm supports arbitrary updates in the graph in constant time per operation.

Naor and Stockmeyer [1995] point out the connection between local algorithms and *self-stabilizing algorithms* [Dijkstra 1974; Dolev 2000; Schneider 1993]. A self-stabilizing algorithm arrives at a legitimate state—"stabilizes"—in finite time regardless of the initial states of the nodes. Work on self-stabilizing algorithms [Awerbuch and Sipser 1988; Awerbuch and Varghese 1991] provides, as a simple special case, a mechanical way to transform a constant-time deterministic distributed algorithm into a self-stabilizing algorithm that stabilizes in constant time; see Lenzen et al. [2009] for more details on the connection between local and self-stabilizing algorithms.

3.2. Value of Information

In the model of local algorithms, we assume that local computation is free. Hence our focus is primarily on the amount of information needed in distributed decision making: what can we do with the information that is available in the constant-radius neighborhood of a node. Positive and negative results for local algorithms can be interpreted as information-theoretic upper and lower bounds; they give insight into the value of information [Papadimitriou and Yannakakis 1991, 1993].

3.3. Other Models of Computing

Local algorithms are closely connected to circuit complexity and the complexity class NC^0 [Aaronson et al. 2011]: if a distributed constant-size problem can be solved with a local algorithm, then for any bounded-degree graph $\mathcal{G}$ there is a bounded-fan-in Boolean circuit that maps the local inputs to the local outputs, and the depth of the circuit is independent of the size of $\mathcal{G}$. As pointed out by Wattenhofer and Wattenhofer [2004], a local algorithm provides an efficient algorithm in the PRAM model, but a PRAM algorithm is not necessarily local.

Sterling [2008] shows that lower bounds for local algorithms can be applied to derive lower bounds in the tile assembly model. Gibbons [2008] points out that the envisioned shape-shifting networks will require not only advances in hardware, but also novel local algorithms.

3.4. Sublinear-Time Centralized Algorithms

A local algorithm for a distributed constant-size problem provides a linear-time centralized algorithm: simply simulate the local algorithm for each node. Parnas and Ron [2007] show that in some cases it is possible to use a local algorithm to design a sublinear-time (or even constant-time) centralized approximation algorithm; see also Nguyen and Onak [2008] and Floréen et al. [2010].

For example, consider a local approximation algorithm $\mathcal{A}$ for the vertex cover problem (see Section 4.4 for the definition). For a given input graph $\mathcal{G}$, algorithm $\mathcal{A}$ produces a feasible and approximately optimal vertex cover C. From the point of view of a centralized algorithm, the local algorithm $\mathcal{A}$ can be interpreted as an oracle with which we can access the cover C: for any given node v, we can efficiently determine whether $v \in C$ or not by simulating algorithm $\mathcal{A}$ at node v. Therefore we can estimate the size of the cover C by sampling nodes uniformly at random; for each node we determine whether it is in C or not. Furthermore, as we know that C is approximately optimal, estimating the size of C allows us to estimate the size of the minimum vertex cover of graph $\mathcal{G}$ as well.

These kinds of algorithms can be used to obtain information about the global properties of very large graphs. Lovász [2008] gives examples of such graphs: the Internet, the social network of all living people, the human brain, and crystal structures. Many of these are not explicitly given and not completely known; however, it may be possible to obtain information about these graphs by sampling nodes and their local neighborhoods. From this perspective, the sublinear-time algorithm by Parnas and Ron [2007] puts together neighborhood sampling and a local approximation algorithm to estimate the global properties of huge graphs.

4. PROBLEMS

Now we proceed to give the definitions of the computational problems that we discuss in this survey. Most of these problems are classical combinatorial problems; for more details and background, see textbooks on graph theory [Diestel 2005], combinatorial optimization [Korte and Vygen 2006; Papadimitriou and Steiglitz 1998],

NP-completeness [Garey and Johnson 1979], and approximation algorithms [Ausiello et al. 2003; Vazirani 2001].

4.1. Encoding of Input and Output

When we study local algorithms, we assume that the problem instance is given in a distributed manner: each node in the communication graph $\mathcal{G}$ knows part of the input. For graph problems, the connection between the communication graph $\mathcal{G}$ and the structure of the problem instance is usually straightforward: we simply assume that the communication graph $\mathcal{G}$ is our input graph; note that we can easily modify the communication graph by removing edges and nodes that are irrelevant from the perspective of the graph problem that we are solving. For more general packing and covering problems (such as the set cover problem or packing LPs) there is more freedom. However, it is fairly natural to represent such problems in terms of bipartite graphs, and this has been commonly used in the literature [Bartal et al. 1997; Kuhn et al. 2006b; Papadimitriou and Yannakakis 1993]. We follow this convention.

The exact definitions of the local input i_v and output o_v are usually fairly straightforward. For unweighted graph problems, we do not need any task-specific information in the local input i_v; the structure of the communication graph $\mathcal{G}$ is enough. For weighted graph problems, the local input i_v contains the weight of node v and the weights of the incident edges. Hence all unweighted graph problems in bounded-degree graphs are distributed constant-size problems; weighted problems are distributed constant-size problems if the weights are represented with a bounded number of bits.

If the output is a subset $X \subseteq V$ of nodes, then the local output o_v is simply one bit of information: whether $v \in X$ or not. If the output is a subset $X \subseteq E$ of edges, then the local output o_v contains one bit for each incident edge e. The algorithm must produce a correct output no matter how we choose the port numbers in the communication graph $\mathcal{G}$.

4.2. Independent Sets

A set of nodes $I \subseteq V$ is an *independent set* if no two nodes in I are adjacent, that is, there is no edge $\{u, v\} \in E$ with $u \in I$ and $v \in I$.

In typical applications, graph $\mathcal{G}$ is interpreted as a conflict graph: an edge $\{u, v\} \in E$ indicates that the activities of u and v conflict with each other. For example, if $\mathcal{G}$ is a wireless network, an edge $\{u, v\}$ may indicate that the radio transmission of device u interferes with device v [Jain et al. 2005]. In such a setting, an independent set I is a conflict-free set of activities: all devices in I can be active simultaneously. Usually we are interested in finding large independent sets.

An independent set I is *maximal* if it cannot be extended, that is, $I \cup \{v\}$ is not an independent set for any $v \in V \setminus I$. In a centralized setting, a maximal independent set is easy to find by using a greedy algorithm, but as we will see in this survey, it is a difficult problem from the perspective of distributed local algorithms. Note that a maximal independent set is not necessarily a *maximum independent set*, that is, an independent set that maximizes $|I|$; finding a maximum independent set is a classical NP-hard optimization problem.

4.3. Matchings

A set of edges $M \subseteq E$ is a *matching* if the edges in M do not share a node, that is, if $\{t, u\} \in M$ and $\{t, v\} \in M$ then $u = v$. Again, a matching is *maximal* if it cannot be extended. A set of edges $M \subseteq E$ is a *simple* 2-*matching* if for each node u, the number of edges $e \in M$ with $u \in e$ is at most 2.

Matchings have applications that are similar to those of independent sets: they identify a conflict-free set of activities. For example, if an edge $\{u, v\} \in E$ represents a

data transmission between the devices u and v, and each device can take part in at most one data transmission at a time, then a matching identifies a set of data transmissions that can be active simultaneously [Hajek and Sasaki 1988].

Many distributed systems have a natural bipartite structure: for example, there are clients that are connected to servers, or mobile users that communicate with base stations. In such systems, a matching M can be interpreted as an assignment: which server serves which client.

If M is a matching in a bipartite graph, we say that an edge $\{u, v\} \in E \setminus M$ is *unstable* if $\{u, s\} \in M$ implies $p(u, s) > p(u, v)$ and $\{v, t\} \in M$ implies $p(v, t) > p(v, u)$. That is, if we interpret port numbers as a ranking of possible partners, both u and v would prefer each other to their current partners (if any). Matching M is *ε-stable* [Floréen et al. 2010] if the number of unstable edges is at most $\varepsilon|M|$, and the matching is *stable* [Gale and Shapley 1962; Gusfield and Irving 1989] if there is no unstable edge. Stable matchings are attractive if servers and clients are selfish agents: if a matching is stable, then there is no client-server pair that would prefer to unilaterally change the matching in their own benefit.

We can also study relaxations of matchings. Let $\mathcal{G}$ be a bipartite graph with the parts $V = V_1 \cup V_2$. Assume that $M \subseteq E$ is a subset of edges, and let $\deg_M(v) = |\{e \in M : v \in e\}|$ be the number of edges in M that are incident to $v \in V$. We say that M is a *semi-matching* [Harvey et al. 2006] if $\deg_M(u) = 1$ for all $u \in V_1$. Intuitively, a semi-matching assigns each client $u \in V_1$ to exactly one server $v \in V_2$, but a single server may receive multiple requests. Now if a server $v \in V_2$ processes the requests sequentially, the first request is handled in 1 time unit, the second request is handled in 2 time units, etc. Let $c(M, v) = 1 + 2 + \cdots + \deg_M(v)$ be the total processing delay of the tasks that are handled by the server $v \in V_2$, and let $c(M) = \sum_v c(M, v)$ be the total processing delay of all tasks. An aptimal semi-matching M minimizes the cost function $c(M)$; intuitively, and optimal semi-matching balances the load as equally as possible, in order to minimize the average waiting times of the clients.

4.4. Domination and Covers

A set of nodes $D \subseteq V$ is a *dominating set* if every node in $V \setminus D$ has a neighbor in D. A dominating set D is *connected* if the subgraph of $\mathcal{G}$ induced by D is connected.

Applications of dominating sets can be found, for example, in wireless sensor networks [Krishnamachari 2005]. In such networks, we can interpret $\mathcal{G}$ as a redundancy graph: an edge $\{u, v\} \in E$ indicates that the sensor nodes u and v are so close to each other that they are pairwise redundant; whenever device u is active, device v can be asleep and vice versa [Cardei et al. 2002]. Hence if D is a dominating set, then all other devices $V \setminus D$ can be asleep and conserve energy.

There are many problems that are related to dominating sets. We will here focus on three examples: edge dominating sets, edge covers, and vertex covers.

A set of edges $D \subseteq E$ is an *edge dominating set* [Chlebík and Chlebíková 2006; Fujito and Nagamochi 2002; Yannakakis and Gavril 1980] if for each edge $\{u, v\} \in E \setminus D$ there is an edge $e \in D$ incident to u or v or both. Edge dominating sets are closely related to matchings: a maximal matching is an edge dominating set, and a minimum-size maximal matching is a minimum-size edge dominating set [Allan and Laskar 1978; Yannakakis and Gavril 1980]. The connection between dominating sets and independent sets is weaker: a maximal independent set is a dominating set, but a minimum-size maximal independent set is not necessarily a minimum-size dominating set.

A set of edges $C \subseteq E$ is an *edge cover* if for each node $v \in V$ there is an edge $e \in C$ with $v \in e$. An edge cover exists if and only if there are no isolated nodes. A set of nodes $C \subseteq V$ is a *vertex cover* if $V \setminus C$ is an independent set. In other words, C is a vertex cover if for each edge $\{u, v\} \in E$ either $u \in C$ or $v \in C$ or both.

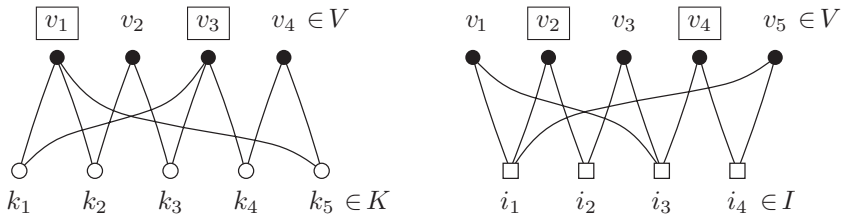

Fig. 2. (a) A set cover instance with $\Delta_V = 3$ and $\Delta_K = 2$; a minimum-size solution is $X = \{v_1, v_3\}$. (b) A set packing instance with $\Delta_V = 2$ and $\Delta_I = 3$; a maximum-size solution is $X = \{v_2, v_4\}$.

4.5. Partitions

A *domatic partition* [Cockayne and Hedetniemi 1975; Feige et al. 2002] is a partition of V into disjoint dominating sets. A domatic partition $V = D_1 \cup D_2 \cup \cdots \cup D_k$ can be interpreted as a schedule that controls a wireless sensor network [Cardei et al. 2002]: first we activate all nodes in D_1 and put all other nodes asleep, then we activate all nodes in D_2 and put all other nodes asleep, etc.

A *vertex k-coloring* of $\mathcal{G}$ assigns a color from the set $\{1, 2, \ldots, k\}$ to each node of $\mathcal{G}$ such that adjacent nodes have different colors. An *edge coloring* is analogous: one assigns a color to each edge such that adjacent edges have different colors. Put otherwise, a vertex coloring partitions V into disjoint independent sets, and edge coloring partitions E into disjoint matchings; such partitions can be used to schedule the activities of nodes and edges in a conflict-free manner.

Naor and Stockmeyer [1995] define the problem of *weak coloring*. A weak k-coloring of $\mathcal{G}$ assigns labels $\{1, 2, \ldots, k\}$ to the nodes of $\mathcal{G}$ such that each nonisolated node has at least one neighbor with a different label. Any graph admits a weak 2-coloring. Note that weak coloring is related to domatic partitions: if there are no isolated nodes, then a weak 2-coloring is also a domatic partition of size 2.

A *cut* is an arbitrary partition of V into two sets, $V = X \cup Y$. The size of the cut is the number of edges $\{u, v\} \in E$ with $u \in X$ and $v \in Y$.

4.6. Covering Problems

Let $\mathcal{G} = (V \cup K, E)$ be a bipartite graph. Each edge $\{v, k\} \in E$ joins an *agent* $v \in V$ and a *customer* $k \in K$; each node knows its role. The maximum degree of an agent $v \in V$ is Δ_V, and the maximum degree of a customer $k \in K$ is Δ_K. A subset $X \subseteq V$ is a *set cover* if each customer is covered by at least one agent in X, that is, for each customer $k \in K$ there is an adjacent agent $v \in X$ with $\{k, v\} \in E$. See Figure 2(a).

The vertex cover problem is a special case of the set cover problem with $\Delta_K = 2$: each customer (edge) can be covered by 2 agents (nodes). The edge cover problem is a special case of the set cover problem with $\Delta_V = 2$: each agent (edge) covers 2 customers (nodes). The dominating set problem in a graph with maximum degree Δ is a special case of the set cover problem with $\Delta_V = \Delta_K = \Delta + 1$.

The problem of finding a minimum-size set cover can be written as an integer program

$$\begin{aligned} &\text{minimize} && \sum_{v \in V} x_v \\ &\text{subject to} && \sum_{v \in V} c_{kv} x_v \geq 1 \qquad \forall\, k \in K, \\ &&& x_v \in \{0, 1\} \quad \forall\, v \in V, \end{aligned} \tag{1}$$

where $c_{kv} = 0$ if $\{k, v\} \notin E$ and $c_{kv} = 1$ if $\{k, v\} \in E$. The LP relaxation of (1) is a 0/1 *covering LP*

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & \sum_{v \in V} c_{kv} x_v \geq 1 \quad \forall\, k \in K, \\ & x_v \geq 0 \quad \forall\, v \in V. \end{aligned} \tag{2}$$

In a general *covering LP* we can have an arbitrary $c_{kv} \geq 0$ for each edge $\{k, v\} \in E$.

4.7. Packing Problems

Let $\mathcal{G} = (V \cup I, E)$ be a bipartite graph. Each edge $\{v, i\} \in E$ joins an *agent* $v \in V$ and a *constraint* $i \in I$; each node knows its role. The maximum degree of an agent $v \in V$ is Δ_V, and the maximum degree of a constraint $i \in I$ is Δ_I. A subset $X \subseteq V$ is a *set packing* if each constraint is covered by at most one agent in X, that is, for each constraint $i \in I$ there is at most one adjacent agent $v \in X$ with $\{v, i\} \in E$. See Figure 2(b).

The independent set problem is a special case of the set packing problem with $\Delta_I = 2$. The maximum matching problem is a special case of the set packing problem with $\Delta_V = 2$.

The problem of finding a maximum-size set packing can be written as an integer program

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & \sum_{v \in V} a_{iv} x_v \leq 1 \quad \forall\, i \in I, \\ & x_v \in \{0, 1\} \quad \forall\, v \in V, \end{aligned} \tag{3}$$

where $a_{iv} = 0$ if $\{i, v\} \notin E$ and $a_{iv} = 1$ if $\{i, v\} \in E$. The LP relaxation of (3) is a 0/1 *packing LP*

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & \sum_{v \in V} a_{iv} x_v \leq 1 \quad \forall\, i \in I, \\ & x_v \geq 0 \quad \forall\, v \in V. \end{aligned} \tag{4}$$

In a general *packing LP* we can have an arbitrary $a_{iv} \geq 0$ for each $\{i, v\} \in E$. A packing LP is a dual of a covering LP and vice versa.

4.8. Mixed Packing and Covering

Finally, we can study linear programs with both packing constraints (constraints of the form $A\mathbf{x} \leq \mathbf{1}$ for a nonnegative matrix A) and covering constraints (constraints of the form $C\mathbf{x} \geq \mathbf{1}$ for a nonnegative matrix C). In general, it may be that there is no feasible solution that satisfies both packing and covering constraints; however, we can formulate a linear program where the objective is to violate the covering constraints as little as possible (the case of violating the packing constraints as little as possible is analogous). We arrive at a *max-min LP* where the objective is to maximize ω subject to $A\mathbf{x} \leq \mathbf{1}$, $C\mathbf{x} \geq \omega\mathbf{1}$, and $\mathbf{x} \geq \mathbf{0}$.

In a distributed setting, we have a bipartite graph $\mathcal{G} = (V \cup I \cup K, E)$. Each edge $e \in E$ is of the form $e = \{v, i\}$ or $e = \{v, k\}$ where $v \in V$ is an *agent*, $i \in I$ is a *constraint*, and

$k \in K$ is a *customer* (or an *objective*); each node knows its role. The maximum degree of an agent $v \in V$ is Δ_V, the maximum degree of a constraint $i \in I$ is Δ_I, and the maximum degree of a customer $k \in K$ is Δ_K. The objective is to

$$\begin{aligned} &\text{maximize} && \omega \\ &\text{subject to} && \sum_{v \in V} a_{iv} x_v \leq 1 && \forall\, i \in I, \\ &&& \sum_{v \in V} c_{kv} x_v \geq \omega && \forall\, k \in K, \\ &&& x_v \geq 0 && \forall\, v \in V. \end{aligned} \tag{5}$$

Again, $a_{iv} = 0$ if $\{i, v\} \notin E$, $a_{iv} \geq 0$ if $\{i, v\} \in E$, $c_{kv} = 0$ if $\{k, v\} \notin E$, and $c_{kv} \geq 0$ if $\{k, v\} \in E$. In a 0/1 *max-min LP* we have $a_{iv}, c_{kv} \in \{0, 1\}$ for all $i \in I$, $k \in K$, and $v \in V$.

The applications of max-min LPs and analogous min-max LPs include tasks related to fair bandwidth allocation in communication networks and lifetime maximization in wireless sensor networks [Floréen et al. 2011].

5. AUXILIARY INFORMATION AND LOCAL VIEWS

In the model defined in Section 2, we have not assumed that the local algorithm has access to any information beyond the port numbering and the task-specific local input i_v. If we do not have any auxiliary information such as unique node identifiers in i_v, we call the network *anonymous*. In this section we will explore the fundamental limitations of anonymous networks, and possible extensions of the model.

5.1. Symmetry Breaking

In an anonymous network, a port numbering does not provide enough information to break the symmetry [Angluin 1980; Johnson and Schneider 1985; Yamashita and Kameda 1996]. To see this, consider a deterministic local algorithm and the network in Figure 1(a). The port-numbered graph is symmetric. It is easy to see that we cannot break the symmetry with the local algorithm if the local inputs are identical. Whatever message node u sends to its port $x \in \{1, 2\}$ on the first communication round, node v sends the same message to its port x if both run the same deterministic algorithm. Whatever message node u receives from its port x on the first communication round, node v receives the same message from its port x. The local state of node u after r communication rounds is equal to the local state of node v after r communication rounds. Eventually, the local output o_u is identical to the local output o_v.

More generally, we can choose the port numbers in an n-cycle so that for each node the port number 1 leads in a counterclockwise direction and the port number 2 leads in a clockwise direction. If the local inputs are identical, the local outputs are identical as well, regardless of the local horizon r. From the point of view of most combinatorial problems, this is discouraging: an empty set is the only matching or independent set that can be constructed by any local algorithm in this case; the set of all nodes is the only dominating set or vertex cover that can be constructed; and vertex coloring or edge coloring is not possible.

However, there are some positive examples of local algorithms that do not require any auxiliary information besides a port numbering. To better understand the possibilities and limitations of this model, we first introduce the concepts of covering graphs and unfoldings.

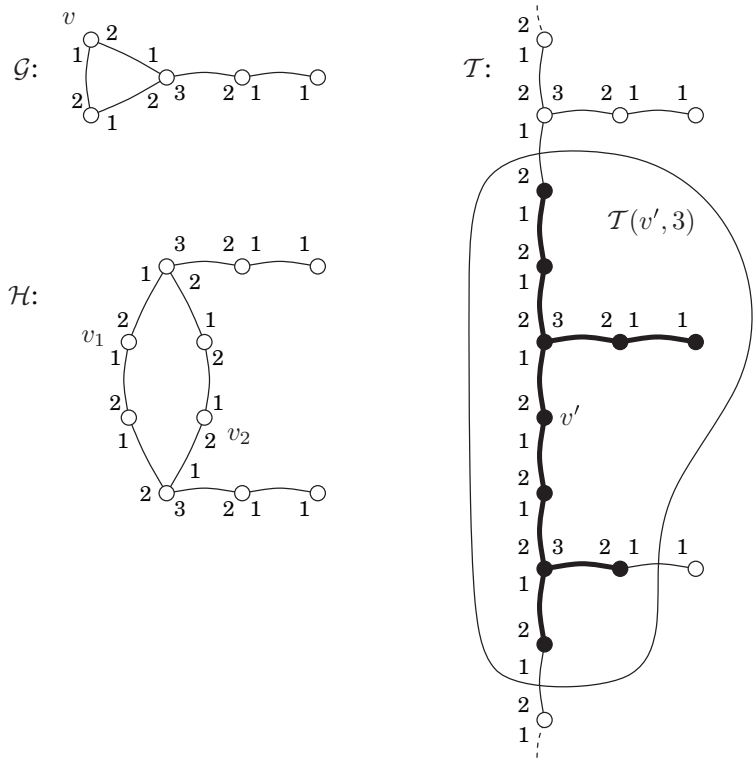

Fig. 3. Covering graphs.

5.2. Covering Graphs and Unfoldings

We say that a port-numbered graph $\mathcal{H} = (V_\mathcal{H}, E_\mathcal{H})$ is a *covering graph* of $\mathcal{G} = (V, E)$ if there is a surjective mapping $f\colon V_\mathcal{H} \to V$ with the following property: for each $v \in V_\mathcal{H}$ and for each integer x, the neighbor x of v in $\mathcal{H}$ is u if and only if the neighbor x of $f(v)$ in $\mathcal{G}$ is $f(u)$. The surjection f is a *covering map*. See Figure 3 for an illustration: graph $\mathcal{H}$ is a covering graph of $\mathcal{G}$; we can choose a covering map f with $f(v_1) = f(v_2) = v$.

The *unfolding* or the universal covering graph [Angluin 1980] of a connected graph $\mathcal{G}$ is an acyclic, connected covering graph $\mathcal{T}$. The unfolding always exists, it is unique (up to isomorphism), and it is finite if and only if $\mathcal{G}$ is a tree. See Figure 3 for an illustration: the infinite tree $\mathcal{T}$ is the unfolding of $\mathcal{G}$; we can choose a covering map f with $f(v') = v$. The tree $\mathcal{T}$ is also the unfolding of $\mathcal{H}$; we can choose, for example, a covering map f with $f(v') = v_1$. This is no coincidence; because $\mathcal{H}$ and $\mathcal{G}$ have a common covering graph (in this case $\mathcal{H}$) they also have the same unfolding.

Informally, we can construct the unfolding $\mathcal{T}$ of a graph $\mathcal{G}$ as follows. Choose an arbitrary node of $\mathcal{G}$ as a starting point. Traverse graph $\mathcal{G}$ in a breadth-first manner; if we revisit a node because of a cycle, treat it as a new node.

This simple intuitive explanation of the unfolding is sufficient for our purposes. See, for example, Godsil and Royle [2004, Section 6.8] for more information on covering graphs in a pure graph-theoretic setting; note that the term "lift" has also been used to refer to a covering graph [Amit et al. 2001; Hoory 2002]. For more information on universal covering graphs, see, for example, Angluin [1980]. An analogous concept in topology is a universal covering space; see, for example, Hocking and Young [1961, Section 4.8] or Munkres [2000, Section 80].

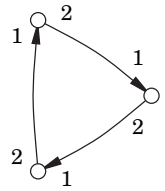

Fig. 4. A communication graph with a port numbering and an orientation; compare to Figure 1.

5.3. Local View

Let v be a node in an anonymous, port-numbered network $\mathcal{G}$. Let $\mathcal{T}$ be the unfolding of $\mathcal{G}$, and let v' be a preimage of v in the covering map f, as in the example of Figure 3.

The *radius-r local view* of node v is the subgraph $\mathcal{T}(v', r)$ of $\mathcal{T}$ induced by $B_{\mathcal{T}}(v', r)$. Put otherwise, the radius-r local view of v is the radius-r neighborhood of its preimage v' in the unfolding. See Figure 3 for an illustration in the case $r = 3$. The local view does not depend on the choice of $v' \in f^{-1}(\{v\})$.

Now we are ready to characterize exactly what we can do in the port numbering model. We begin with the good news. In a deterministic local algorithm with local horizon r, each node v can construct its radius-r local view [Boldi and Vigna 2001; Yamashita and Kameda 1996]. There is a simple local algorithm that gathers this information in r communication rounds: Initially, each node knows its radius-0 local view. In communication round i, each node floods its radius-$(i-1)$ local view to each neighbor, and includes the outgoing port number in the message. After round i, each node pieces together the local views received from its neighbors; this results in the radius-i local view. We can also gather the local input for each node in the local view. Hence, in a local algorithm each node v can choose its output o_v based on all information that is available in its radius-r local view. If the local views of nodes u and v differ, then the local outputs of nodes u and v can differ as well.

The bad news is that choosing the local output based on the local view is, in a sense, the only thing that one can do in a local algorithm [Angluin 1980; Boldi and Vigna 2001; Yamashita and Kameda 1996]. This is easily understood if we consider the covering map f from the unfolding $\mathcal{T}$ to the communication graph $\mathcal{G}$, and apply the same local algorithm $\mathcal{A}$ in both $\mathcal{T}$ and $\mathcal{G}$. Initially, for each node v' in $\mathcal{T}$, the local state of v' in $\mathcal{T}$ and $v = f(v')$ in $\mathcal{G}$ is the same. Furthermore, on each communication round, v' and v perform the same local computation, send the same messages, and receive the same messages; here we use the fact that f is a local isomorphism that preserves the port numbering. Hence, after r communication rounds, both v' and v must produce the same output. Therefore the output of v in a local algorithm with local horizon r only depends on its local view $B_{\mathcal{T}}(v', r)$.

Among others, this shows that a local algorithm cannot distinguish between $\mathcal{G}$ and $\mathcal{H}$ in Figure 3 because they have the same unfolding $\mathcal{T}$. Node v in $\mathcal{G}$, nodes v_1 and v_2 in $\mathcal{H}$, and node v' in $\mathcal{T}$ all produce the same output. We can see that a local algorithm in an anonymous network cannot even detect if there are triangles (3-cycles) in the network.

5.4. Graphs with Orientation

So far we have assumed that we have a port numbering in the communication graph $\mathcal{G}$. We proceed to study a slightly stronger assumption [Mayer et al. 1995]: in addition to the port numbering, we are given an *orientation* of graph $\mathcal{G}$. That is, for each edge $\{u, v\} \in E$, we have chosen exactly one direction, either (u, v) or (v, u). See Figure 4 for an illustration. In a port numbering, each node chooses an ordering on incident edges, while in an orientation, each *edge* chooses an ordering on incident *nodes*.

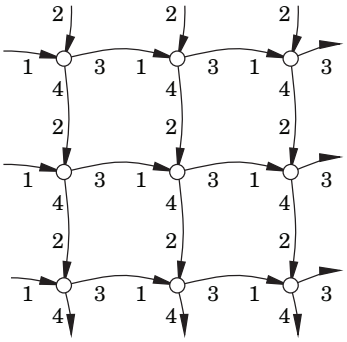

Fig. 5. A 4-regular graph with a port numbering and an orientation.

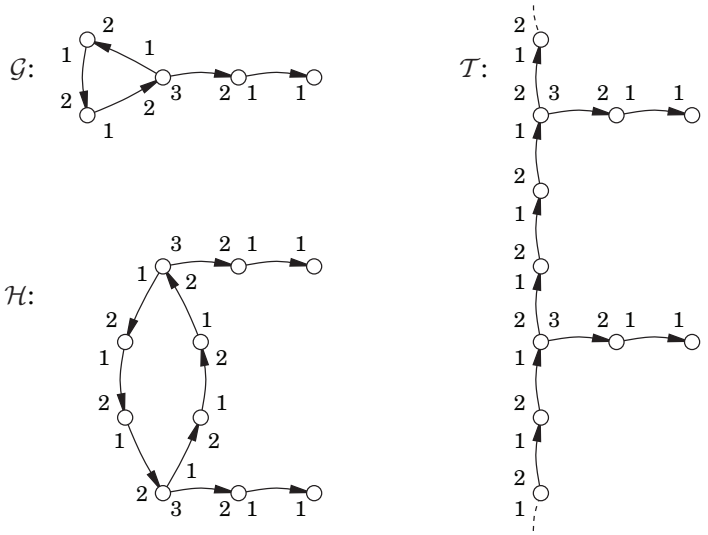

Fig. 6. Covering graphs with a port numbering and an orientation; compare to Figure 3.

We use the standard terminology for directed graphs: If the edge $\{u, v\}$ has the orientation (u, v), then u is a *predecessor* of v and v is a *successor* of u. The *in-degree* of a node is the number of predecessors, that is, the number of edges entering the node. Similarly, the *out-degree* of a node is the number of successors, that is, the number of edges leaving the node.

At first sight, having an arbitrary orientation in addition to an arbitrary port numbering does not seem to help much. In an n-cycle, we can have all edges directed consistently in a clockwise direction, as shown in Figure 4. Hence we obtain the same negative results as in the case of an n-cycle with only a port numbering. More generally, we can construct a d-regular graph for any *even* constant d such that the local view of each node is identical [Naor and Stockmeyer 1995], in spite of a port numbering and an orientation; see Figure 5 for an example. Furthermore, as shown in Figure 6, having an orientation does not help one to tell a graph from its cover.

Surprisingly, it turns out that in graphs where every node has an *odd* degree, an orientation together with a port numbering is enough to break the symmetry in the

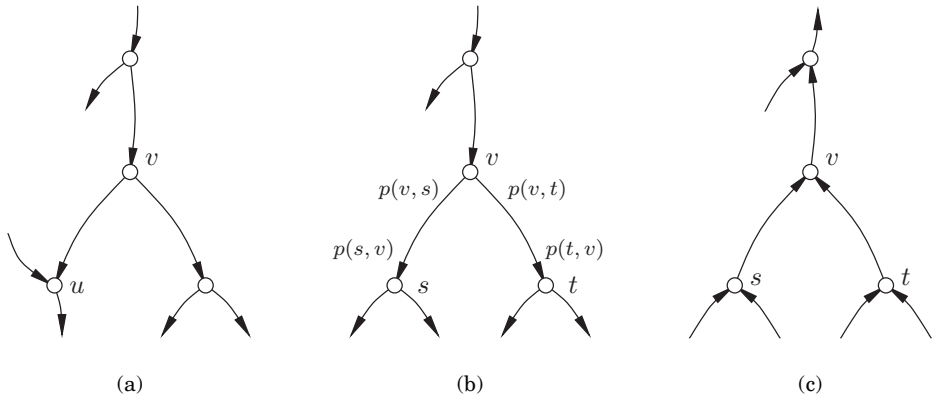

Fig. 7. A 3-regular graph with a port numbering and an orientation. (a) Different out-degrees. (b) Out-degree is 2. (c) Out-degree is 1.

following sense: the output of a nonisolated node v is different from the output of at least one neighbor u of v.

Example 5.1. Consider a 3-regular graph with a port numbering and an orientation. Let v be an arbitrary node; we show that the local view of v is different from the local view of at least one neighbor of v. Figure 7 illustrates the three possible cases. In Figure 7(a), node v and its neighbor u have different out-degrees; hence the local view of v differs from the local view of u. Otherwise the out-degree of v and each neighbor of v is the same. The common out-degree of v and its neighbors is either 1 or 2. Figure 7(b) illustrates the case where the common out-degree is 2. In this case the local view of s is necessarily different from the local view of t: both have exactly one predecessor, and the port numbers assigned to these unique incoming edges are different because $p(v, s) \neq p(v, t)$. Therefore the local view of v is different from the local view of s or t (or both). Figure 7(c) illustrates the case where the common out-degree is 1; this is analogous to the case of the out-degree 2.

This argument can be generalized to any graph, as long as the degree of each node is odd. We present the details in Section 7.3 when we review a local algorithm for weak coloring [Mayer et al. 1995; Naor and Stockmeyer 1995].

5.5. Graphs with Unique Identifiers

We can make an even stronger assumption: each node v has a *globally unique identifier* as part of its local input i_v. Usually the identifiers are assumed to be a permutation of $\{1, 2, \ldots, |V|\}$ or a subset of $\{1, 2, \ldots, \text{poly}(|V|)\}$. Naturally we require that the local algorithm solves the problem for any choice of unique identifiers.

Globally unique identifiers are a standard assumption in the field of local algorithms. This is a strictly stronger assumption than having only port numbers and an orientation available; all negative results for the case of globally unique identifiers imply negative results in anonymous networks and anonymous oriented networks.

Unfortunately, the assumption on globally unique identifiers means that the problem is not of distributed constant size: the number of bits required to encode the identifiers increases as the size of the network increases. In practice, for many algorithms that are designed under the assumption of globally unique identifiers, it is sufficient to have *locally unique identifiers*. That is, we assume that a local algorithm with local horizon r has access to identifiers that are unique within every radius-r neighborhood

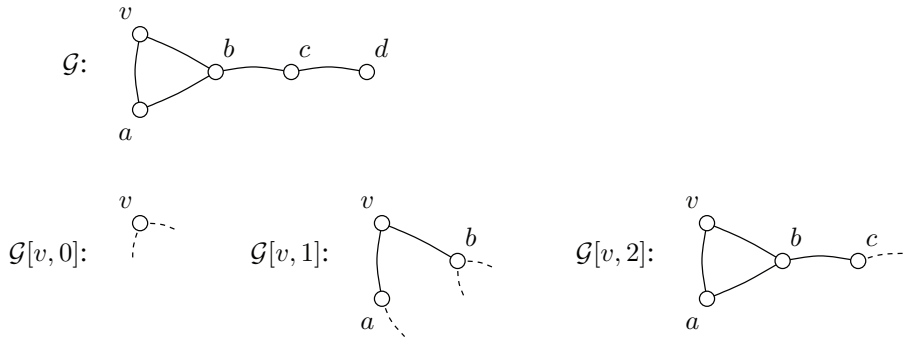

Fig. 8. The local view of node v in a graph $\mathcal{G}$ with unique node identifiers.

in the communication graph $\mathcal{G}$. In a bounded-degree graph, it is then possible to choose identifiers that are locally unique but have constant size.

With globally or locally unique identifiers, a node v in graph $\mathcal{G}$ can tell for each pair of nodes s, t in its radius-r local view whether $f(s) = f(t)$, that is, whether they represent the same node in the original graph $\mathcal{G}$. This implies that each node v can reconstruct the subgraph $\mathcal{G}[v, r]$ of $\mathcal{G}$; see Section 2.2 for the definition and Figure 8 for an illustration. Hence, when we study local algorithms in networks with unique identifiers, it is sufficient to present a function that maps the subgraph $\mathcal{G}[v, r]$ to the local output o_v [Naor and Stockmeyer 1995].

We note that the difference between $\mathcal{G}(v, r)$ and $\mathcal{G}[v, r]$ is usually immaterial. After all, $\mathcal{G}[v, r + 1]$ contains $\mathcal{G}(v, r)$ as a subgraph, and $\mathcal{G}(v, r)$ contains $\mathcal{G}[v, r]$ as a subgraph. We are typically not interested in additive constants in the local horizon r. Hence we can use either $\mathcal{G}(v, r)$ or $\mathcal{G}[v, r]$ to derive both positive and negative results, whichever is more convenient. If the local output cannot be determined based on $\mathcal{G}(v, r)$ for any constant r, then there is no local algorithm for the task; if it can be determined based on $\mathcal{G}(v, r)$ for some constant r, then there is a local algorithm. We can even go as far as to use this as the definition of a local algorithm, if we study networks with unique identifiers.

6. NEGATIVE RESULTS

In this section we review negative results for local algorithms. Nontrivial results are summarized in Tables I and II.

6.1. Preliminary Observations

There are two simple arguments that can be used to show that a problem cannot have a local algorithm: *inherently non-local problems* and the *impossibility of symmetry breaking*.

A problem is inherently non-local if the output at a node u may depend on the input at a node v with $d_{\mathcal{G}}(u, v) = \Omega(|V|)$. By definition, a local algorithm cannot solve a problem that is inherently nonlocal. Constructing a spanning tree is a classical example of a simple problem that is inherently nonlocal; see Figure 9 for an illustration. Finding a stable matching is another example of a nonlocal problem [Floréen et al. 2010].

The problem of finding a maximal matching is, in a sense, much more local. For example, if we already have a solution M for a graph $\mathcal{G}$, a local change in $\mathcal{G}$ requires only local changes in the solution M. However, as we discussed in Section 5.1, it is not possible to break the symmetry with a local algorithm in an n-cycle if the nodes are anonymous; a port numbering and an orientation of $\mathcal{G}$ do not help. Therefore it

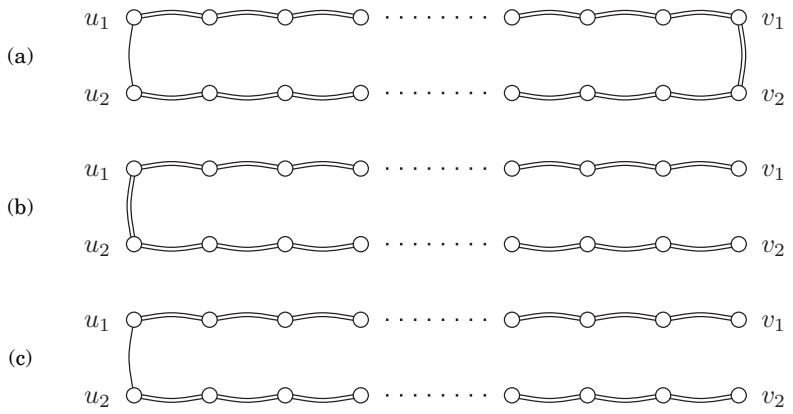

Fig. 9. (a) An n-cycle $\mathcal{G}$; the double lines show a spanning tree. (b) A local change in graph $\mathcal{G}$ near nodes v_1 and v_2 requires a nonlocal change in the spanning tree near nodes u_1 and u_2. (c) A local algorithm cannot even verify whether a given set of edges is a spanning tree or not [Korman and Kutten 2006]. In every local neighborhood, this nontree looks similar to a spanning tree in part (a) or (b).

is not possible to find a maximal matching in an anonymous n-cycle, oriented or not. The same negative results apply to the problems of finding a maximal independent set, a vertex coloring, an edge coloring, a weak coloring, an $O(1)$-approximation of a maximum matching, an $O(1)$-approximation of a maximum independent set, a $(3-\varepsilon)$-approximation of a minimum dominating set, and a $(2-\varepsilon)$-approximation of a minimum vertex cover in anonymous n-cycles. As a straightforward generalization, there is no local $o(\Delta)$-approximation algorithm for the minimum dominating set problem in anonymous bounded-degree graphs.

In what follows, we focus on negative results that hold even if globally unique identifiers are available.

6.2. Comparable Identifiers

Let us first focus on *order-invariant* algorithms: we assume that unique node identifiers are available, but the algorithm is only allowed to compare the identifiers and not access their numerical value.

It turns out that being able to compare identifiers does not help much in symmetry breaking. For example, in an n-cycle, we can assign the node identifiers $1, 2, \ldots, n$ in an increasing order. If we pick any two nodes $u, v \in U = \{r+1, r+2, \ldots, n-r\}$, then the radius-$r$ neighborhood of u looks identical to the radius-r neighborhood of v, assuming that a node can only exploit the ordering of the identifiers. Therefore every node in U must make the same decision; see, for example, Kuhn [2005, Section 2.7.2] and Floréen et al. [2008d]. We immediately obtain the same negative results that we had for anonymous networks in an n-cycle: for example, there is no local algorithm for finding a maximal matching, a maximal independent set, a vertex coloring, an edge coloring, or a weak coloring.

6.3. Numerical Identifiers

A natural approach would be to exploit the numerical values of the identifiers; after all, this is exactly what the classical (distributed but not constant-time) algorithm for vertex coloring by Cole and Vishkin [1986] does.

Unfortunately, a general result by Naor and Stockmeyer [1995] shows that local algorithms for so-called locally checkable labelings (these include vertex colorings and

maximal independent sets in bounded-degree graphs) do not benefit from the numerical values of the identifiers: if there is a local algorithm that uses the numerical values, there is an order-invariant local algorithm as well.

More specifically, Linial [1992] shows that a synchronous distributed algorithm for vertex 3-coloring in an n-cycle with unique identifiers requires $\Omega(\log^* n)$ communication rounds. Here $\log^* n$ denotes the iterated (base-2) logarithm of n, that is, the smallest integer $k \geq 0$ such that k iterated applications of the function $x \mapsto \log_2 x$ to the initial value n results in a value at most 1.

Linial's result holds even under the assumption that there is a consistent clockwise orientation in the n-cycle. As a direct implication, an algorithm for finding an edge 3-coloring, a maximal independent set, or a maximal matching in an n-cycle requires $\Omega(\log^* n)$ communication rounds as well. The barrier of $\Omega(\log^* n)$ is hard to break even if we are allowed to provide arbitrary instance-specific *advice* to some nodes in the network [Fraigniaud et al. 2007].

6.4. Approximations for Combinatorial Problems

So far we have seen that there are no local algorithms for problems such as vertex coloring, edge coloring, maximal independent set, or maximal matching. However, it is possible to find a feasible independent set or a matching with a local algorithm (the empty set), and similarly there is a trivial local algorithm for finding a vertex cover or a dominating set (the set of all nodes). This raises the question of whether there is a local approximation algorithm for any of these problems, with a nontrivial approximation guarantee.

Unfortunately, this does not seem to be the case. Czygrinow et al. [2008], Lenzen and Wattenhofer [2008], and Lenzen [2011, Section 11] show that it is not possible to find a constant-factor approximation of a maximum independent set or a maximum matching in an n-cycle with a deterministic local algorithm. Czygrinow et al.'s elegant proof uses Ramsey's theorem [Graham et al. 1980; Ramsey 1930]; Lenzen and Wattenhofer build on Linial's [1992] work.

These results imply that there is no local constant-factor approximation algorithm for the maximum cut problem in an n-cycle. If we can find a cut $\{X, Y\}$ of size k in an n-cycle, then it is possible to find a matching with at least $k/3$ edges as well. The edges that cross the cut form a bipartite graph $\mathcal{H}$, the cut $\{X, Y\}$ is a 2-coloring of the bipartite graph $\mathcal{H}$, and the algorithm that we will describe in Section 7.1 finds a maximal matching in the 2-colored graph $\mathcal{H}$.

As another corollary, there is no local $(2 - \varepsilon)$-approximation algorithm for the edge cover problem in a cycle. To see this, consider a $2n$-cycle $\mathcal{G} = (V, E)$. A minimum edge cover has n edges, and a maximum matching has n edges as well. Hence a $(2 - \varepsilon)$-approximate edge cover $C \subseteq E$ has at most $(2-\varepsilon)n$ edges, and its complement $M = E \setminus C$ has at least εn edges. Furthermore, each $v \in V$ is covered by at least one edge in C; therefore each $v \in V$ is covered by at most one edge in M. Hence M is a matching, and within factor $1/\varepsilon$ of the optimum.

An analogous argument shows that there is no local $(2-\varepsilon)$-approximation algorithm for the vertex cover problem in an n-cycle. Furthermore, there is no local $(3 - \varepsilon)$-approximation algorithm for the dominating set problem. Note that the complement $X = V \setminus D$ of a dominating set D in an n-cycle can be turned into an independent set [Czygrinow et al. 2008]: a node $v \in X$ is adjacent to at most one other node $u \in X \setminus \{v\}$. Exchanging the roles of edges and nodes, the same argument shows that there is no local $(3 - \varepsilon)$-approximation algorithm for the edge dominating set problem in a cycle. Moreover, we cannot find more than one disjoint dominating set in a cycle; because a $3n$-cycle has 3 disjoint dominating sets, this shows that no local algorithm can find a $(3 - \varepsilon)$-approximation of a maximum domatic partition.

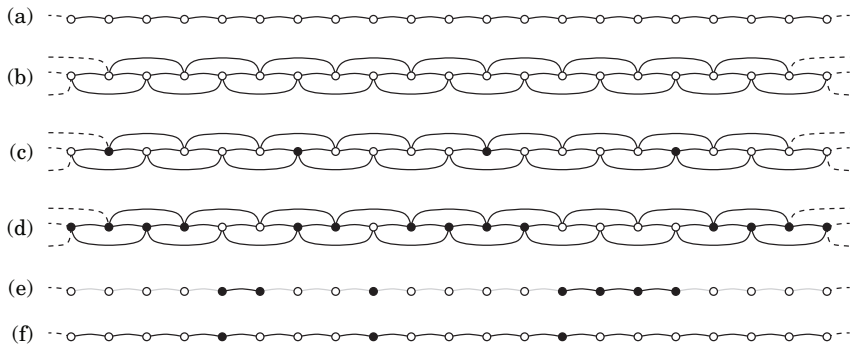

Fig. 10. There is no local $(2k+1-\varepsilon)$-approximation algorithm for the dominating set problem in $2k$-regular graphs (the case $k = 2$).

More generally, Czygrinow et al. [2008] and Lenzen and Wattenhofer [2008] show that for any constant $\varepsilon > 0$, a local algorithm cannot produce a factor $2k + 1 - \varepsilon$ approximation of a minimum dominating set in $2k$-regular graphs. The basic argument is as follows. Figure 10(a) shows a $(2k + 1)n$-cycle $\mathcal{G} = (V, E)$. Using a local algorithm, we can construct a $2k$-regular graph $\mathcal{H} = (V, E')$, as illustrated in Figure 10(b). A minimum dominating set of $\mathcal{H}$ has n nodes (Figure 10(c)); therefore a hypothetical $(2k + 1 - \varepsilon)$-approximation algorithm has to return a dominating set D with at most $(2k + 1 - \varepsilon)n$ nodes (Figure 10(d)). Therefore its complement $X = V \setminus D$ has at least εn nodes. Furthermore, because D is a dominating set, there is no path with more than $2k$ nodes in the subgraph of $\mathcal{G}$ induced by X (Figure 10(e)). Hence we can construct an independent set I with at least $\varepsilon n/(2k)$ nodes (Figure 10(f)), which is a contradiction with the local inapproximability of the independent set problem in cycles.

Czygrinow et al. [2008] consider the case $k = 2$ to show that a local algorithm cannot find a factor $5-\varepsilon$ approximation of a minimum dominating set in planar graphs. Lenzen and Wattenhofer [2008] consider a general k to show that a local algorithm cannot find a constant-factor approximation of a minimum dominating set in unit-disk graphs (see Section 9.1 for the definition).

In the preceding proof, we have focused on regular graphs with an even degree. As we saw in Section 5.4, some amount of symmetry breaking is possible in graphs where each node has an odd degree. Nevertheless, we can derive a slightly weaker result for regular graphs with an odd degree: a local algorithm cannot produce a factor $k+1-\varepsilon$ approximation of a minimum dominating set in $(2k+1)$-regular graphs [Åstrand et al. 2010]. To see this, consider a $(k + 1)n$-cycle $\mathcal{G} = (V, E)$; see Figure 11(a). We can use a local algorithm to construct a $(2k + 1)$-regular graph $\mathcal{H}$ as illustrated in Figure 11(b); each original node $v \in V$ simulates a pair of nodes, v' and v'', in $\mathcal{H}$. A minimum dominating set of $\mathcal{H}$ has n nodes (Figure 11(c)). A hypothetical $(k + 1 - \varepsilon)$-approximation algorithm has to return a dominating set D with at most $(k + 1 - \varepsilon)n$ nodes (Figure 11(d)). Let $X \subseteq V$ consist of the nodes $v \in V$ such that both $v' \notin D$ and $v'' \notin D$ (Figure 11(e)). We know that the size of X is at least εn; furthermore, X does not induce paths with more than $2k$ nodes in $\mathcal{G}$. Hence we can construct an independent set I with at least $\varepsilon n/(2k)$ nodes (Figure 11(f)), a contradiction.

This negative result holds even in bipartite graphs where a 2-coloring is given as part of the local input. Incidentally, the construction in Figure 11 is the so-called bicolored double cover of the construction in Figure 10; we will revisit bicolored double covers in more detail when we present positive results in Section 7.1.

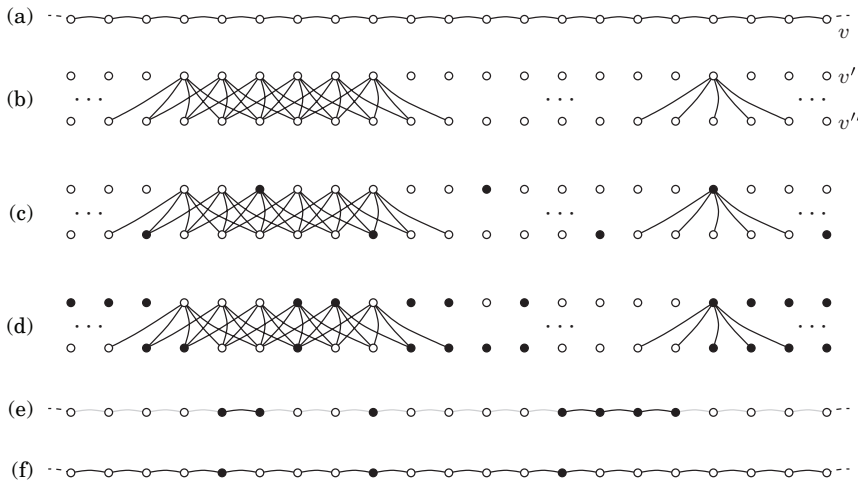

Fig. 11. There is no local $(k + 1 - \varepsilon)$-approximation algorithm for the dominating set problem in $(2k + 1)$-regular graphs (the case $k = 2$).

6.5. Approximations for LPs

The negative results in the previous section build on the impossibility of symmetry breaking in combinatorial problems. However, there are also negative results for linear programs.

Bartal et al. [1997] observe that a $(1 + \varepsilon)$-approximation algorithm for packing and covering LPs requires $\Omega(1/\varepsilon)$ communication rounds.

Kuhn [2005, 2008], Kuhn et al. [2004, 2006b, 2010], and Moscibroda [2006] show that it is not possible to find a constant-factor approximation of a minimum vertex cover, minimum dominating set, or maximum matching in general graphs with a local algorithm, if there is no degree bound. The results extend to the LP relaxations of these problems as well, and hence to 0/1 packing LPs and 0/1 covering LPs.

Floréen et al. [2008b, 2008c, 2011] present a tight lower bound for the local approximability of max-min LPs. In bounded-degree graphs, no local algorithm can achieve the approximation factor $\Delta_I(1 - 1/\Delta_K)$.

7. POSITIVE RESULTS

In spite of all negative results that we saw in Section 6, a few local algorithms are known. In this section, we review known deterministic local algorithms; prominent positive results are also summarized in Tables III and IV.

We begin with two different techniques, both of which yield a local approximation algorithm for the vertex cover problem: Section 7.1 presents a technique based on bicolored covering graphs; Section 7.2 presents a linear programming approach.

7.1. Bicolored Matchings and Vertex Covers

In a centralized setting, there is a simple 2-approximation algorithm for the vertex cover problem: find any maximal matching and take the endpoints.[1] We cannot find a maximal matching with a local algorithm in general graphs; nevertheless, we can apply the same basic idea indirectly to design a local approximation algorithm for the vertex cover problem.

[1]Papadimitriou and Steiglitz [1998] attribute this algorithm to Fanica Gavril and Mihalis Yannakakis.

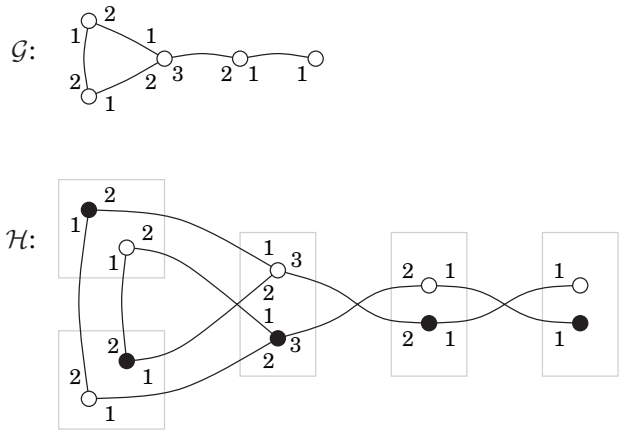

Fig. 12. The bicolored graph $\mathcal{H}$ is the bipartite double cover of graph $\mathcal{G}$. Note that the graphs are isomorphic to those in Figure 3.

We say that the communication graph $\mathcal{G}$ is *bicolored* if a vertex 2-coloring of $\mathcal{G}$ is given as part of the local input; each node knows whether it is black or white. Clearly graph $\mathcal{G}$ has to be bipartite; otherwise there is no 2-coloring.

Hańćkowiak et al. [1998] present a simple local algorithm for the problem of finding a maximal matching in a bicolored bounded-degree graph. A port numbering is enough; unique node identifiers are not needed. The algorithm performs the following two steps repeatedly:

(1) Each unmatched black node sends a proposal to one of its white neighbors. The neighbors are chosen in the order of port numbers.
(2) Each white node accepts the first proposal that it receives. If several proposals are received in the same round, ties are broken with port numbers.

After 2Δ steps, this results in a maximal matching M. To see that M is maximal, consider an edge $e = \{u, v\} \in E \setminus M$ such that u is a black node and v is a white node. One of the following holds: (i) u never sent a proposal to v, or (ii) v rejected the proposal from u. In case (i), node u is matched, and in case (ii), node v is matched. Hence $M \cup \{e\}$ is not a matching.

Now we know how to find a maximal matching in a bicolored graph. It turns out that with the help of this simple algorithm, it is possible to find a 3-approximation of a minimum vertex cover in an arbitrary bounded-degree graph [Polishchuk and Suomela 2009]. The key observation is the following: for any graph $\mathcal{G}$, we can construct the bicolored graph $\mathcal{H}$ that is the *bipartite double cover* [Angluin 1980; Bottreau and Métivier 1996; Imrich and Pisanski 2008] of $\mathcal{G}$; see Figure 12 for an illustration.

The bipartite double cover of $\mathcal{G}$, also known as the Kronecker double cover, is the Kronecker product [Weichsel 1962] of the graphs $\mathcal{G}$ and K_2. In essence, for each original node v in graph $\mathcal{G}$, we create two copies: a black copy and a white copy. If u and v are adjacent in the original graph $\mathcal{G}$, then the black copy of u is adjacent to the white copy of v in graph $\mathcal{H}$ and vice versa. Port numbers can be inherited from $\mathcal{G}$. It follows that $\mathcal{H}$ is a covering graph of $\mathcal{G}$ (see Section 5.2). Furthermore, it is a *double* cover: the covering map f maps exactly 2 nodes of $\mathcal{H}$ onto each node of $\mathcal{G}$.

Now let $\mathcal{A}$ be the local algorithm for maximal matchings in bicolored graphs. A local algorithm that runs in a general graph $\mathcal{G}$ can *simulate* the behavior of $\mathcal{A}$ in the bicolored double cover $\mathcal{H}$; a node v in $\mathcal{G}$ is responsible for simulating the behavior of

both its black copy and its white copy. Once the simulation completes, each node can inspect the output produced by its two copies.

Hence we can find a maximal matching M in the bicolored double cover $\mathcal{H}$ and map it back to the original graph $\mathcal{G}$. This gives us a subset of edges $X \subseteq E$ in $\mathcal{G} = (V, E)$ with the following properties: (i) for each node $v \in V$, there are at most 2 edges incident to v in X; and (ii) for each edge $\{u, v\} \in E$, at least one of u, v is incident to an edge $x \in X$. Put otherwise, X is a simple 2-matching and its endpoints are a vertex cover $C \subseteq V$.

A minimum-size vertex cover C^* must cover all edges, including the edges in the simple 2-matching X. A node $v \in C^*$ can cover at most 2 edges in X, and an edge in X is covered by at most 2 nodes in C. Hence $|C| \leq 2|X| \leq 4|C^*|$. We have a simple algorithm that finds a 4-approximation of a minimum vertex cover; the algorithm is local and it does not need unique node identifiers.

A more careful analysis shows that $|C| \leq 3|C^*|$, so this is actually a local 3-approximation algorithm for the vertex cover problem [Polishchuk and Suomela 2009]; the bottleneck is a path of length 2 in the set X. A repeated application of bicolored double covers can be used to find a 2-approximation of a minimum vertex cover [Åstrand et al. 2009]. See Hańćkowiak et al. [2001] for an example of a nonlocal distributed algorithm that exploits bicolored double covers.

7.2. Linear Programs and Vertex Covers

In a centralized setting, another 2-approximation algorithm for the vertex cover problem can be obtained by deterministic LP rounding [Hochbaum 1982]: (i) solve the LP relaxation of the vertex cover problem; and (ii) output the set of nodes $v \in V$ with $x_v \geq 1/2$. Unlike the algorithm based on a maximal matching, this directly generalizes to the problem of finding a minimum-weight vertex cover as well.

To obtain the LP relaxation of a vertex cover instance, we first write the vertex cover instance as a set cover instance. The set cover instance determines an integer program of the form (1). The LP relaxation of the vertex cover instance is the corresponding 0/1 covering LP (2).

A local algorithm cannot find an exact solution of a linear program; the problem is inherently non-local. However, local approximation algorithms for packing and covering LPs are known. The first such algorithm was presented by Papadimitriou and Yannakakis [1993]. In this simple algorithm, the "capacity" of each constraint in a packing LP is distributed evenly among the adjacent agents; the approximation factor is Δ_I. Kuhn and Wattenhofer [2005] present improved local approximation algorithms for special cases of packing and covering LPs. Finally, Kuhn [2005; 2008] and Kuhn et al. [2006b] present local approximation algorithms for general packing LPs and covering LPs. Among others, they show that 0/1 packing LPs and 0/1 covering LPs admit local approximation schemes in bounded-degree graphs.

Hence we can find a factor $1 + \varepsilon$ approximation of the LP relaxation of the minimum vertex cover problem in bounded-degree graphs. Therefore deterministic LP rounding yields a factor $2 + \varepsilon$ approximation of a minimum vertex cover in bounded-degree graphs, for any $\varepsilon > 0$. Note that the vertex cover problem is a special case of the set cover problem with $\Delta_K = 2$; the same technique of deterministic LP rounding can be applied to design a local $(\Delta_K + \varepsilon)$-approximation algorithm for the set cover problem in bounded-degree graphs, for an arbitrary Δ_K.

It is also possible to use the primal-dual schema [Papadimitriou and Steiglitz 1998; Vazirani 2001] to design an algorithm that finds an approximation of a minimum vertex cover directly without a rounding step. Moscibroda [2006, Section 6.1] uses this approach to design a $(4 + \varepsilon)$-approximation algorithm for the vertex cover problem in bounded-degree graphs.

7.3. Weak Coloring

Weak coloring provided the first example of a nontrivial combinatorial problem that admits a local algorithm. Naor and Stockmeyer [1995] show that if $\mathcal{G}$ is a bounded-degree graph and every node of $\mathcal{G}$ has an *odd* degree, then there is a local algorithm that finds a weak 2-coloring of $\mathcal{G}$. Mayer et al. [1995] further show that this is possible without unique identifiers; a port numbering and an orientation is enough.

Let us now show how to find a weak coloring with $\Delta^{O(\Delta)}$ colors, using only a port numbering and an orientation. Each node v is colored with a vector $\mathbf{x}(v)$ that consists of the following components:

—the out-degree and in-degree of v;
—$p(u, v)$ for each successor u of v, in the order of increasing $p(v, u)$;
—$p(u, v)$ for each predecessor u of v, in the order of increasing $p(v, u)$.

Example 7.1. Assume that v has three neighbors, u_1, u_2, u_3, with the port numbers $p(v, u_1) = 1$, $p(v, u_2) = 2$, and $p(v, u_3) = 3$. Assume that the edges are oriented (v, u_1), (u_2, v), and (v, u_3). Then

$$\mathbf{x}(v) = \big(2,\ 1,\ p(u_1, v),\ p(u_3, v),\ p(u_2, v)\big),$$

that is, the out-degree, the in-degree, two port numbers for the edges leaving v, and one port number for the edges entering v. Note that the elements are port numbers of the form $p(\cdot, v)$ but they are ordered by the port numbers $p(v, \cdot)$.

To see that the vectors $\mathbf{x}(v)$ are a weak coloring of the graph, we generalize the argument that we saw in Section 5.4. Let v be an arbitrary node. We need to show that there is a neighbor u of v such that $\mathbf{x}(u) \neq \mathbf{x}(v)$. If the first two components (out-degree and in-degree) are equal in $\mathbf{x}(v)$ and $\mathbf{x}(u)$ for each neighbor u of v, we can consider the following two cases.

First, assume that the in-degree of v is k and the out-degree of v is at least $k + 1$. Then there are at least $k + 1$ successors of v, and each successor has k predecessors. By the pigeonhole principle, there are at least two successors of v, call them s and t, and an index i with the following property: the element i of the vector $\mathbf{x}(s)$ is $p(v, s)$ and the element i of the vector $\mathbf{x}(t)$ is $p(v, t)$. Therefore $\mathbf{x}(s) \neq \mathbf{x}(t)$, and we cannot have $\mathbf{x}(v) = \mathbf{x}(s) = \mathbf{x}(t)$. Put otherwise, v has a different color from s or t (or both).

Second, assume that the in-degree of v is at least $k + 1$ and the out-degree of v is k. This case is analogous: we apply the pigeonhole principle to the predecessors of v.

Note that this analysis does not go through in a graph with an even degree. We may have in-degrees equal to out-degrees, and therefore we cannot invoke the pigeonhole principle; consider, for example, the 4-regular graph in Figure 5.

So far we have seen how to find a weak coloring with a constant but large number of colors. In the following section, we review techniques that can be used to reduce the number of colors.

7.4. Color Reduction

A local algorithm cannot find a vertex coloring, but it can decrease the number of colors. Given a k-coloring for a constant k, it is easy to design a local algorithm that finds a $(\Delta + 1)$-coloring. In essence, we run a greedy algorithm. The original k-coloring partitions the network in k subsets $X_1, X_2, \ldots, X_k$. In the step $i = 1, 2, \ldots, k$, the nodes in subset X_i choose a color that is not used by any of their neighbors in $X_1 \cup X_2 \cup \cdots \cup X_{i-1}$. Each subset X_i is an independent set; hence all nodes in X_i can make their choices independently in parallel. In the worst case, $\Delta + 1$ colors are needed.

A much more efficient algorithm can be designed by exploiting the numerical values of the original colors. In a cycle, the technique originally presented by Cole and

Vishkin [1986] allows one to decrease the number of colors from k to $O(\log k)$ in one step. Iterating the procedure, we can turn a k-coloring into a 3-coloring with a local algorithm in $O(\log^* k)$ steps. The textbook by Cormen et al. [1990, Section 30.5] has a good illustration of the Cole-Vishkin technique; in essence, a node with a b-bit label relabels itself with an $O(\log b)$-bit label (i, x) that identifies the index i and the value x of the first bit in its old label that differs from the next node in the cycle.

The same basic idea can be applied in general bounded-degree graphs [Goldberg et al. 1988]. If k is a constant, a k-coloring can be turned into a $(\Delta + 1)$-coloring with a local algorithm in $O(\Delta + \log^* k)$ steps [Barenboim and Elkin 2009; Kuhn 2009]; see also Attiya et al.'s [1999] algorithm that finds a $(\Delta + 1)$-coloring assuming that a so-called t-orientation is given. Naor and Stockmeyer [1995] show how to turn a weak k-coloring into a weak 2-coloring.

Naturally, if we are given a k-coloring for a constant k, we can also solve a number of other problems with a local algorithm. The symmetry has been broken and, for example, finding a maximal independent set is then easy [Attiya et al. 1999; Awerbuch et al. 1989]. The following algorithm provides a reasonable trade-off between efficiency and simplicity: first apply a color reduction algorithm, and then find a maximal independent set with a greedy algorithm. However, if Δ is large, more efficient alternatives exist; see, for example, the techniques used by Schneider and Wattenhofer [2008].

7.5. Matchings

As we have seen in Section 6.4, there is no local algorithm for finding a constant-factor approximation of a maximum matching in any family of graphs that contains n-cycles. However, positive results are known for bicolored graphs and for graphs where each node has an odd degree.

We have already seen the algorithm by Hańćkowiak et al. [1998] for finding a maximal matching in bicolored bounded-degree graphs. A maximal matching is a 2-approximation of a maximum matching. It is possible to improve the approximation factor to $1 + \varepsilon$ for any $\varepsilon > 0$; it is enough to make sure that there is no augmenting path of length $O(1/\varepsilon)$ [Åstrand et al. 2010]. With the help of the bicolored double cover from Section 7.1, this yields a $(2 + \varepsilon)$-approximation of a maximum-size simple 2-matching in general bounded-degree graphs. Maximal matchings can also be used to find approximations of semi-matchings [Czygrinow et al. 2011].

Bounded-degree graphs where each node has an odd degree admit a local Δ-approximation algorithm for the maximum matching problem; the algorithm proceeds as follows. We can first find a weak 2-coloring $c\colon V \to \{1, 2\}$ with the algorithm by Naor and Stockmeyer [1995]. Let $X \subseteq E$ consist of the edges $\{u, v\} \in E$ with $c(u) \neq c(v)$. Let $\mathcal{H} = (U, X)$ be the subgraph of $\mathcal{G}$ induced by the edges in X, that is, U consists of the endpoints of the edges in X. Now $\mathcal{H}$ together with c is a bicolored graph; therefore we can find a maximal matching M in $\mathcal{H}$; this is also a matching in $\mathcal{G}$. Let us now establish the approximation ratio. Let M^* be a maximum matching. Let $U \subseteq V$ consist of the nodes matched in M and let $U^* \subseteq V$ consists of the nodes matched in M^*. If $v \in U^* \setminus U$, then v is nonisolated in $\mathcal{G}$ and hence it is adjacent to a node u with the opposite color, by the definition of a weak 2-coloring. Therefore $\{u, v\}$ is an edge in the subgraph $\mathcal{H}$, and since M is maximal, we have $u \in U$. Furthermore, u is adjacent to at least one node in U; therefore u is adjacent to at most $\Delta - 1$ nodes in $U^* \setminus U$. Summing over all nodes in U, we have $|U^* \setminus U| \leq (\Delta - 1)|U|$, that is, $|U^*| \leq \Delta|U|$ and $|M^*| \leq \Delta|M|$. The approximation factor can be further improved to $(\Delta + 1)/2$, which is tight [Åstrand et al. 2010].

The problem of finding a stable matching is inherently nonlocal, even in bicolored graphs. However, it is possible to find an ε-stable matching in a bicolored bounded-degree graph with a local algorithm [Floréen et al. 2010]. In essence, the local algorithm runs Gale and Shapley's [1962] algorithm for a constant number of rounds. The

same local algorithm finds a $(2+\varepsilon)$-approximation of a maximum-weight matching in bicolored bounded-degree graphs.

7.6. Domination

In a bounded-degree graph, the set of all nodes is a factor $\Delta+1$ approximation of a minimum dominating set. If each node has an odd degree, it is possible to find a factor Δ approximation with a local algorithm, using a port numbering and an orientation: find a weak 2-coloring [Mayer et al. 1995; Naor and Stockmeyer 1995] and output all nodes of color 1 and all isolated nodes. This set is, by definition, a dominating set: a nonisolated node of color 2 is adjacent to at least one node of color 1. To establish the approximation ratio, assume that $\mathcal{G}=(V,E)$ is connected; otherwise we can apply the result to each connected component. If $|V|=1$, the claim is trivial. Otherwise, let D^* be an optimal dominating set in $\mathcal{G}$, let D_1 consist of the nodes of color 1, and let $D_2=V\setminus D_1$ consist of the nodes of color 2. Now D^*, D_1, and D_2 are dominating sets in $\mathcal{G}$. Furthermore, for any dominating set D, it holds that $|D|\geq |V|/(\Delta+1)$ because a node in D can only dominate at most Δ nodes outside D. Hence

$$|D_1| = |V| - |D_2| \leq |V| - \frac{|V|}{\Delta+1} = \frac{\Delta|V|}{\Delta+1} \leq \Delta|D^*|,$$

that is, D_1 is a Δ-approximation of a minimum dominating set in $\mathcal{G}$. Moreover, (D_1, D_2) is a domatic partition of size 2 if there are no isolated nodes, and a maximum domatic partition has at most $\delta+1$ disjoint dominating sets where δ is the minimum degree of $\mathcal{G}$. Hence a weak 2-coloring provides a factor $(\delta+1)/2$ approximation of a maximum domatic partition if there are no isolated nodes; the trivial solution (V) is optimal if there is an isolated node (i.e., if $\delta=0$).

It is possible to perform local modifications in a weak 2-coloring so that the number of color-1 nodes is at most as large as the number of color-2 nodes [Åstrand et al. 2010]. This approach provides a factor $(\Delta+1)/2$ approximation algorithm for the dominating set problem in graphs of odd degree, and a Δ-approximation in general graphs for an odd Δ.

Czygrinow et al. [2008], Lenzen et al. [2010], and Lenzen [2011, Section 13] present a local, constant-factor approximation algorithm for the dominating set problem in planar graphs. The current best-known approximation factor is 130 [Lenzen et al. 2010; Lenzen 2011].

In Section 7.1 we saw how to use a maximal matching in a bicolored double cover to find a constant-factor approximation for the vertex cover problem. The same technique provides a 4-approximation of a minimum edge dominating set as well, and with minor modifications the approximation factor can be improved to $4-2/\Delta$ [Suomela 2010].

7.7. Trivial Algorithms

For some families of graphs, there is a trivial local approximation algorithm for the vertex cover problem. For example, the set of all nodes is a 2-approximation of a minimum vertex cover in regular graphs, and the set of all nonisolated nodes is a 6-approximation of a minimum vertex cover in unit-disk graphs [Kuhn 2005; Wiese and Kranakis 2008a].

There is a trivial, local approximation algorithm for the edge cover problem: independently and in parallel, each node $v\in V$ chooses one neighbor $x(v)\in V$ with $\{v,x(v)\}\in E$. The set $C=\{\{v,x(v)\} : v\in V\}$ is a factor 2 approximation of a minimum edge cover. This generalizes to the minimum-weight edge cover as well: each node chooses the least expensive edge.

The edge cover problem is a special case of the set cover problem with $\Delta_V=2$. The trivial 2-approximation algorithm for the edge cover problem can be applied to

approximating set cover as well: each customer $k \in K$ chooses independently and in parallel which agent $v \in V$ covers it. The approximation factor is Δ_V.

7.8. Local Verification and Locally Checkable Proofs

Korman et al. [2005, 2010] and Korman and Kutten [2006, 2007] and Fraigniaud et al. [2011] study the problem of *verifying* a solution with a local algorithm. We have seen that the problem of finding a spanning tree is inherently nonlocal, and if we are given a spanning tree in a natural way as a subset of edges, a local algorithm cannot verify whether it really is a spanning tree or not; recall Figure 9.

However, it is possible to give a spanning tree together with a *proof* that can be verified with a local algorithm, so that an invalid input is detected by at least one node (assuming that the communication graph $\mathcal{G}$ is connected). For example, we can orient the spanning tree towards an arbitrary root node. For each node, the proof consists of: (i) the identity of the root node, (ii) the distance to the root node in the spanning tree, and (iii) the edge that points towards the root node.

To encode the preceding proof, we need $O(\log |V|)$ bits per node; we say that the *local proof complexity* of the spanning tree problem is $O(\log |V|)$. In general, we can classify graph problems according to their local proof complexity: for many classical graph problems, the proof complexity is either 0, $\Theta(1)$, $\Theta(\log |V|)$, or $\mathrm{poly}(|V|)$ bits per node [Göös and Suomela 2011].

7.9. Other Problems

Kuhn et al. [2006a] study a generalization of covering LPs, with upper bounds for variables x_v.

Floréen et al. [2008a, 2008b, 2008c, 2009, 2011] present local algorithms for max-min LP. It is possible to find a factor $\Delta_I(1 - 1/\Delta_K) + \varepsilon$ approximation of a max-min LP in bounded-degree graphs; this approximation ratio is tight.

A weak 2-coloring also determines a cut with at least $|E|/\Delta$ edges. This gives a partial answer to Elkin's [2004] question regarding the distributed approximability of the maximum cut problem.

Positive results for the circuit complexity class NC^0 [Applebaum et al. 2006; Cryan and Miltersen 2001; Håstad 1987] may also have positive implications in the field of local algorithms; this possible connection calls for further research.

8. RANDOMIZED LOCAL ALGORITHMS

In the previous section, we have seen that many graph problems can be solved (at least approximately) by using a *deterministic* local algorithm. However, the strong negative results of Section 6 make it impossible to solve classical graph problems such as graph coloring, maximal independent sets, maximal matchings, etc., with a deterministic local algorithm. A natural idea is to extend the model by introducing a source of random bits.

8.1. Nonconstant Guarantees

Randomness is a powerful and classical technique in the design of distributed algorithms, and it is particularly useful in breaking the symmetry [Luby 1986; Alon et al. 1986; Israeli and Itai 1986]. However, in a typical randomized distributed algorithm, either the running time or the performance guarantee depends on the number of nodes.

Luby's [1986] algorithm for maximal independent sets is a good example of the former case. When the algorithm terminates, the output is a maximal independent set; however, the expected running time is logarithmic in the number of nodes.

Kuhn's [2005, Section 4.5] algorithm for one-round graph coloring is a good example of the latter case. The algorithm always terminates in one round, and with a high probability, it produces a proper coloring; however, the number of colors is logarithmic in the number of nodes.

As explained in Section 1.1, our focus is on algorithms for which both the running time and the performance guarantees are independent of the number of nodes; that is, we would obtain a finite performance guarantee even in an infinitely large network. In what follows, we examine how much randomness helps in such a limited setting.

8.2. Negative Results

There are two key techniques which make randomness a practical tool in the design of centralized algorithms, both familiar from textbooks on randomized algorithms [Mitzenmacher and Upfal 2005; Motwani and Raghavan 1995]. First, the probability of a "failure"—the event of producing an infeasible output—can be made negligible by adapting the algorithm to the global properties of the input; for example, the number of iterations can depend on the size of the input. Second, it may be possible to detect incorrect output and reexecute the algorithm until the output is correct, turning a Monte Carlo algorithm into a Las Vegas one.

In a strictly local setting, neither of these two techniques can be applied as such. First, the execution of a local algorithm cannot depend on the size of the input. Second, it is not possible to gather the output in a central location for inspection and reexecute the algorithm depending on whether the output is incorrect. Indeed, if a randomized local algorithm has a nonzero probability of failure given some input, then we can simply take several copies of the input to boost the probability that the algorithm makes a failure in at least one copy; see, for example, Kuhn [2005, Section 4.5].

There are strong negative results on the power of randomness in the local setting. Linial [1992] and Naor [1991] show that randomness does not help in local algorithms where the objective is to color a graph, and Kuhn [2005, Section 4.5] extends this to the problem of color reduction. Naor and Stockmeyer [1995] show that randomness does not help in a local algorithm for any locally checkable labeling; this includes, among others, the problem of finding a maximal matching or a maximal independent set in a bounded-degree graph.

Nevertheless, there are positive results where a randomized local algorithm provides a probabilistic approximation guarantee. For example, in some cases it is possible to give an upper bound on the expected approximation factor. Here the expectation is over the random coin tosses made by the algorithm; nothing is assumed about the input. If we are content with a probabilistic approximation guarantee, it is possible to overcome some of the negative results in Section 6.

Interestingly, Kuhn [2005, Section 2.7.1] shows that probabilistic approximation guarantees do not help with linear programs: if there is a local, randomized algorithm with the expected approximation factor α, then there is a local, deterministic α-approximation algorithm as well. Therefore all positive examples in this section involve combinatorial problems.

8.3. Matchings and Independent Sets

As we have seen in Section 6.4, packing problems such as maximum matching and maximum independent set do not admit deterministic, local approximation algorithms, not even in the case of an n-cycle with unique node identifiers. With these problems, randomness clearly helps.

Wattenhofer and Wattenhofer [2004] present a randomized local approximation algorithm for the maximum-weight matching problem in trees; the expected weight of

the matching is within factor 4 of the optimum. Hoepman et al. [2006] improve the expected approximation ratio to $2 + \varepsilon$.

Nguyen and Onak [2008] present a randomized local algorithm for the maximum matching problem in bounded-degree graphs; the approximation ratio is $1 + \varepsilon$ with high probability.

Czygrinow et al. [2008] present a randomized local algorithm for finding a maximum independent set in a planar graph; the approximation ratio is $1+\varepsilon$ with high probability.

8.4. Maximum Cut and Maximum Satisfiability

Another negative result from Section 6.4 shows that there is no deterministic local approximation algorithm for the maximum cut problem. Again, a randomized local algorithm exists. In this case, we can resort to a very simple algorithm, familiar from introductory courses to randomized algorithms: flip a fair coin for each node to determine its side [Luby 1993; Mitzenmacher and Upfal 2005; Motwani and Raghavan 1995]. The expected size of the cut is $|E|/2$; hence the expected approximation ratio is 2. The algorithm is clearly local; no communication is needed.

A similar approach can be applied to the *maximum satisfiability* (MAX-SAT) problem: choose a random assignment [Mitzenmacher and Upfal 2005; Motwani and Raghavan 1995]. See, for example, Ausiello et al. [2003, Problem LO1] or Garey and Johnson [1979, Problem LO5] for the definition of the problem. However, unlike the maximum cut problem, MAX-SAT has a simple, local, deterministic 2-approximation algorithm: First, for each clause, remove all but one of the literals; in essence, we arrive at a MAX-1-SAT instance. Second, satisfy at least half of the clauses by using a local version of Johnson's [1974] 2-approximation algorithm.

8.5. LP Rounding

Kuhn [2005], Kuhn and Wattenhofer [2005], and Kuhn et al. [2006a, 2006b] present a general framework for designing randomized local algorithms for the set cover and set packing problems in bounded-degree graphs. The solution is obtained in three phases: (1) Solve the LP relaxation of the problem approximately with a local algorithm. (2) Apply randomized rounding to find a candidate solution; at this point, the solution is integral but it is not necessarily feasible. (3) Apply a deterministic algorithm to make the solution feasible.

As discussed earlier in Section 7, the LP relaxations of the set cover and set packing problems have local approximation schemes in bounded-degree graphs. Together with these algorithms, the LP rounding scheme yields the expected approximation ratio $O(\log \Delta_V)$ for the set cover problem and $O(\Delta_V)$ for the set packing problem.

In bounded-degree graphs, these results imply the following expected approximation ratios: $O(\log \Delta)$ for vertex covers, $O(\log \Delta)$ for dominating sets, $O(\Delta)$ for maximum independent sets, and $O(1)$ for maximum matchings.

9. GEOMETRIC PROBLEMS

Now we turn our attention to geometric graphs. In a geometric graph, each node is embedded in a low-dimensional space, typically in the two-dimensional plane.

9.1. Models

Most research has focused on the case where $\mathcal{G}$ is a *unit-disk graph*: a pair of nodes is connected by an edge if and only if the Euclidean distance between them is at most 1. See Figure 13(a).

Work has also been done on generalizations of unit-disk graphs. A *quasi unit-disk graph* [Barrière et al. 2003; Kuhn et al. 2008] is a graph where the nodes are embedded in the two-dimensional plane, the length of an edge is at most 1, and nodes which are

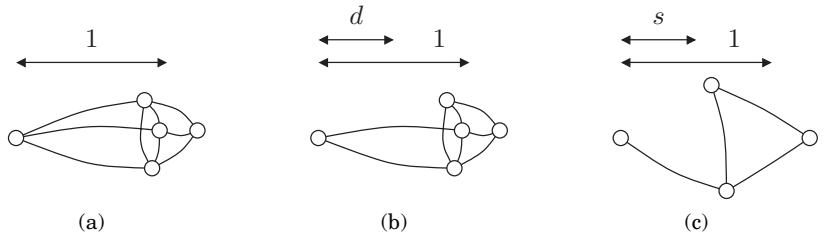

Fig. 13. (a) A unit-disk graph. (b) A quasi unit-disk graph, with $d = 1/2$. (c) A civilized graph, with $s = 1/2$. This is also a quasi unit-disk graph, with $d < s$.

within distance d from each other are always connected by an edge; here $0 < d < 1$ is a constant. See Figure 13(b). A *civilized graph* (graph drawn in a civilized manner) [Doyle and Snell 1984, Section 8.5] is a graph where the nodes are embedded in the two-dimensional plane, the length of an edge is at most 1, and the distance between any pair of nodes is at least s; here $0 < s < 1$ is a constant. See Figure 13(c).

By definition, a civilized graph with parameter s is a quasi unit-disk graph with parameter $d < s$; therefore all positive results for quasi unit-disk graphs apply directly to civilized graphs. Furthermore, a civilized graph is a bounded-degree graph, as can be seen by a simple packing argument.

9.2. Partial Geometric Information

For some problems, it is sufficient to have a local knowledge of the embedding. Kuhn [2005] and Kuhn et al. [2005] show that packing and covering LPs admit local constant-factor approximation algorithms in unit-disk graphs. It is enough that the distances between the nodes are known so that each node can construct a *local coordinate system*.

To give another example, Floréen et al. [2007, 2008d] study scheduling problems in a semigeometric setting in which the coordinates of the nodes are not known, but a small amount of symmetry-breaking information is available.

Most positive results, however, assume that there is a global coordinate system and each node knows its coordinate (so-called *location-aware* nodes). We review these results in the following.

9.3. Algorithms from Simple Tilings

A simple approach for designing local algorithms in a geometric setting is to partition the two-dimensional plane into rectangles, and color the rectangles with a constant number of colors [Hassinen et al. 2008, 2011; Kuhn 2005; Kuhn et al. 2005; Moscibroda 2006; Wiese 2007; Wiese and Kranakis 2008b, 2009a]. Partitioning the two-dimensional plane into rectangles also partitions the network into clusters. If each node knows its coordinates, it knows into which cluster it belongs.

As a concrete example, we can partition the plane into rectangles of size 2×1 and color them with 3 colors so that the distance between a pair of nodes in two different rectangles of the same color is larger than 1. See Figure 14 for illustration; we use the names white, light, and dark for the colors.

By construction, we know that if nodes u and v are in two different rectangles of the same color, then there is no edge $\{u, v\}$ in a (quasi) unit-disk graph. Furthermore, a packing argument shows that there is a constant upper bound D on the diameter of a connected component of a cluster. In Awerbuch et al.'s [1989] terminology, these colored rectangles provide a $(3, D)$-decomposition of the network. In Attiya et al.'s [1999] terminology, the colored rectangles provide a t-orientation of graph $\mathcal{G}$ for $t = 3D$.

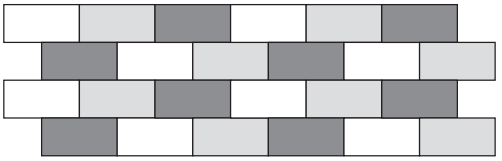

Fig. 14. Partitioning the two-dimensional plane into 3-colored rectangles with dimensions 2×1.

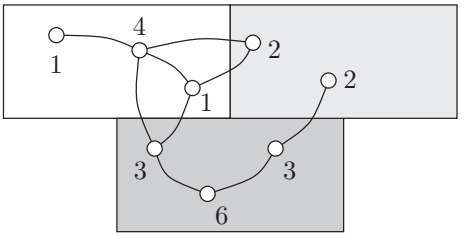

Fig. 15. Factor 3 approximation for vertex coloring. The edges that cross the boundaries of the tiles can be safely ignored in the algorithm.

Now it is easy to design a local 3-approximation algorithm for vertex coloring [Hassinen et al. 2011; Kuhn 2005; Wiese 2007; Wiese and Kranakis 2009a]. We handle each connected component in each cluster independently in parallel. A local algorithm finds an optimal vertex coloring within each component; the components have bounded diameter and hence a local algorithm can gather full information about the component. Connected components in white rectangles assign colors $1, 4, 7, \ldots$, connected components in light rectangles assign colors $2, 5, 8, \ldots$, and connected components in dark rectangles assign colors $3, 6, 9, \ldots$. Put together, we obtain a feasible vertex coloring, and the number of colors that we use is within factor 3 of the optimum; see Figure 15.

A similar idea (with larger 3-colored rectangles) can be used to design local 3-approximation algorithms for the following problems: edge coloring [Hassinen et al. 2011], vertex cover [Hassinen et al. 2008, 2011; Wiese 2007], and dominating set [Hassinen et al. 2008; Hassinen et al. 2011; Wiese 2007]. These are examples of local algorithms that unscrupulously exploit the assumption that local computation is free; nevertheless, Hunt et al. [1998] show how to solve the subproblem of finding a minimum-size dominating set or vertex cover within a rectangle in polynomial time.

Another algorithm design technique that employs the same 3-colored tiling is the sequential greedy strategy. Consider, for example, the task of finding a maximal independent set. We can proceed in three phases as follows [Attiya et al. 1999; Awerbuch et al. 1989; Hassinen et al. 2011]. First, each white rectangle finds a maximal independent set with a greedy algorithm. Then each light rectangle extends the independent set greedily, taking into account the output in neighboring white rectangles. Finally, each dark rectangle extends the independent set greedily, taking into account neighboring white and light rectangles. The same technique can be applied to find a maximal matching [Hassinen et al. 2011; Wiese and Kranakis 2008b] and a vertex $(\Delta + 1)$-coloring [Attiya et al. 1999; Awerbuch et al. 1989; Hassinen et al. 2011]. A local algorithm for maximal matching then gives a 2-approximation of a minimum vertex cover as well.

Finally, it is possible to find a factor 4 approximation of a maximum independent set by using a similar 3-colored tiling [Hassinen et al. 2011; Wiese 2007]. First, each cluster finds a maximum-size independent set in parallel. This may cause conflicts. The conflicts are then resolved; first those that involve white rectangles and then those

that involve light rectangles. At each conflict resolution we lose at most one half of the nodes; hence the remaining nodes provide a factor 4 approximation.

9.4. Other Algorithms

Wiese [2007] and Wiese and Kranakis [2008b, 2009b, 2009c] present local approximation schemes for dominating sets, connected dominating sets, vertex covers, maximum matchings, and maximum independent sets in unit-disk graphs.

Czyzowicz et al. [2008] present a 5-approximation algorithm for the dominating set problem and a 7.46-approximation algorithm for the connected dominating set problem. Wiese [2007] and Wiese and Kranakis [2008a] study local approximation algorithms with local horizon $r \leq 2$ for dominating sets, connected dominating sets, vertex covers, and independent sets in unit-disk graphs. Kuhn and Moscibroda [2007] present a local approximation algorithm for the capacitated dominating set problem in unit-disk graphs; this is a variant of the dominating set problem in which each node has a limited capacity that determines how many neighbors it can dominate.

Šparl and Žerovnik [2005] present a 4/3-approximation algorithm for multicoloring hexagonal graphs. Kaski et al. [2008] present a local approximation scheme for link scheduling (a variant of the graph coloring problem) in geometric graphs.

9.5. Planar Subgraphs and Geographic Routing

In geographic routing [Gąsieniec et al. 2008; Zollinger 2008], it is of interest to construct a connected planar subgraph $\mathcal{H} = (V, E')$ of a unit-disk graph $\mathcal{G} = (V, E)$, with the original set of nodes V but a smaller set of edges $E' \subset E$.

There are local algorithms for constructing planar subgraphs. For example, Gabriel graphs [Gabriel and Sokal 1969] and relative neighborhood graphs [Jaromczyk and Toussaint 1992; Toussaint 1980] can be constructed with simple local rules.

Once we have constructed a planar subgraph of a unit-disk graph, it is possible to route messages with local geometric rules, assuming that we know the coordinates of the target node [Bose et al. 2001; Kranakis et al. 1999].

9.6. Spanners

In applications such as topology control, merely having a connected planar subgraph $\mathcal{H}$ is not enough. Among others, it is desirable that $\mathcal{H}$ is a *geometric t-spanner*. In a t-spanner, for any pair u, v of nodes in $\mathcal{G}$, the shortest path between u and v in $\mathcal{H}$ is at most t times as long as the shortest path between u and v in $\mathcal{G}$; here the length of a path is the sum of the Euclidean lengths of the edges. The constant t is the *stretch factor* of the spanner.

Gabriel graphs and relative neighborhood graphs are not t-spanners for any constant t. Yao graphs [Yao 1982] and θ-graphs [Keil and Gutwin 1992] provide classical examples of spanners that can be constructed with a simple local algorithm. However, these constructions lack some desirable properties; in particular, they do not have a constant upper bound on the node degree.

Examples of more recent work include the following. Wang and Li [2006] present a local algorithm for constructing a planar, bounded-degree spanner in unit-disk graphs. The local algorithms by Li et al. [2004] and Kanj et al. [2008] further guarantee that the total edge length of the spanner is at most a constant factor larger than the total edge length of a minimum spanning tree. Chávez et al. [2006] generalize the results by Li et al. [2004] to quasi unit-disk graphs. Wattenhofer and Zollinger [2004] present a local algorithm that can be applied in arbitrary weighted graphs, not only in geometric graphs.

9.7. Colored Subgraphs

Local algorithms have also been presented for constructing colored subgraphs. Urrutia [2007] presents a local algorithm that constructs a connected, planar, edge-colored subgraph of a unit-disk graph. Wiese [2007] and Wiese and Kranakis [2009a] present a local algorithm that constructs a connected, planar, vertex-colored subgraph of a unit-disk graph. Czyzowicz et al. [2011] present a local algorithm for coloring the nodes in an arbitrary planar subgraph of a unit-disk graph. Czyzowicz et al. [2009] present a local algorithm for coloring the edges in certain subgraphs of unit-disk graphs.

10. OPEN PROBLEMS

We conclude this survey with some open problems related to deterministic local algorithms. We recall that in this work a local algorithm refers to a constant-time algorithm.

Problem 10.1. Is there a local approximation scheme for general packing LPs or covering LPs in bounded-degree graphs?

The local approximation scheme by Kuhn et al. [2006b] assumes not only a degree bound but also an upper bound for the ratio of largest coefficient to smallest coefficient in the LP. Techniques by Luby and Nisan [1993] and Bartal et al. [1997] can be applied to avoid the dependency on coefficients, but this comes at the cost of adding a dependency on the size of the input [Kuhn et al. 2006c].

Problem 10.2. Is there a combinatorial packing problem that admits a nontrivial, deterministic, local approximation algorithm?

Finding a simple 2-matching is a packing problem, but it is a slightly contrived example. It would be interesting to see other, more natural examples of packing problems that can be solved locally, without any auxiliary information.

A partial answer is provided by the local approximation algorithm for the maximum matching problem, based on the weak 2-coloring algorithm by Naor and Stockmeyer [1995]. However, this can be applied only in a graph where every node has an odd degree, a rather stringent assumption.

Problem 10.3. Is there a problem that: (i) can be solved with a local algorithm that exploits the numerical values of the identifiers, and (ii) cannot be solved with an order-invariant local algorithm that merely compares the identifiers?

Naor and Stockmeyer [1995] show that order-invariant local algorithms are sufficient for locally checkable labelings: if there is a local algorithm for a locally checkable labeling problem, then there is an order-invariant algorithm as well. Hence we need to seek for an example outside locally checkable labelings.

If we assume that the set of node identifiers is $\{1, 2, \ldots, |V|\}$, then we can find some examples of problems that admit a local algorithm and do not admit an order-invariant local algorithm. For example, in this case leader election is trivial with a local algorithm (the node number 1 is the leader) but there is no order-invariant local algorithm for the task. However, this example is no longer valid if the unique node identifiers are an arbitrary subset of, say, $\{1, 2, \ldots, 2|V|\}$.

Problem 10.4. How does the local horizon depend on Δ?

Typically, the running time of a local algorithm depends on the maximum degree Δ. However, we do not yet understand thoroughly how the local horizon depends on Δ: in many cases, the fastest known algorithms have running times that are polynomial in Δ [Åstrand and Suomela 2010; Barenboim and Elkin 2009; Floréen et al. 2010;

Hańćkowiak et al. 1998; Kuhn 2009; Suomela 2010] while the well-known lower bounds are typically polylogarithmic in Δ [Kuhn et al. 2004, 2010].

ACKNOWLEDGMENTS

I am grateful to Patrik Floréen, Keijo Heljanko, Juho Hirvonen, Petteri Kaski, Evangelos Kranakis, Christoph Lenzen, Valentin Polishchuk, Joel Rybicki, Christian Scheideler, Roger Wattenhofer, and anonymous reviewers for their helpful comments and suggestions.

REFERENCES

Aaronson, S., Kuperberg, G., and Granade, C. 2011. Complexity zoo. http://www.complexityzoo.com/.

Allan, R. B. and Laskar, R. 1978. On domination and independent domination numbers of a graph. *Discrete Math. 23,* 2, 73–76.

Alon, N., Babai, L., and Itai, A. 1986. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algor. 7,* 4, 567–583.

Amit, A., Linial, N., Matoušek, J., and Rozenman, E. 2001. Random lifts of graphs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Philadelphia, PA, 883–894.

Angluin, D. 1980. Local and global properties in networks of processors. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, New York, 82–93.

Applebaum, B., Ishai, Y., and Kushilevitz, E. 2006. Cryptography in NC^0. *SIAM J. Comput. 36,* 4, 845–888.

Åstrand, M., Floréen, P., Polishchuk, V., Rybicki, J., Suomela, J., and Uitto, J. 2009. A local 2-approximation algorithm for the vertex cover problem. In *Proceedings of the 23rd International Symposium on Distributed Computing (DISC).* Lecture Notes in Computer Science, vol. 5805, Springer, 191–205.

Åstrand, M., Polishchuk, V., Rybicki, J., Suomela, J., and Uitto, J. 2010. Local algorithms in (weakly) coloured graphs. arXiv:1002.0125 [cs.DC].

Åstrand, M. and Suomela, J. 2010. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proceedings of the 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA).* ACM Press, 294–302.

Attiya, H., Shachnai, H., and Tamir, T. 1999. Local labeling and resource allocation using preprocessing. *SIAM J. Comput. 28,* 4, 1397–1413.

Attiya, H., Snir, M., and Warmuth, M. K. 1988. Computing on an anonymous ring. *J. ACM 35,* 4, 845–875.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. 2003. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer.

Awerbuch, B., Goldberg, A. V., Luby, M., and Plotkin, S. A. 1989. Network decomposition and locality in distributed computation. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS).* IEEE, 364–369.

Awerbuch, B. and Sipser, M. 1988. Dynamic networks are as fast as static networks. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS).* IEEE, 206–219.

Awerbuch, B. and Varghese, G. 1991. Distributed program checking: A paradigm for building self-stabilizing distributed protocols. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS).* IEEE, 258–267.

Barenboim, L. and Elkin, M. 2009. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC).* ACM Press, 111–120.

Barrière, L., Fraigniaud, P., Narayanan, L., and Opatrny, J. 2003. Robust position-based routing in wireless ad-hoc networks with irregular transmission ranges. *Wirel. Comm. Mobile Comput. J. 3,* 2, 141–153.

Bartal, Y., Byers, J. W., and Raz, D. 1997. Global optimization using local information with applications to flow control. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS).* IEEE Computer Society Press, 303–312.

Boldi, P. and Vigna, S. 2001. An effective characterization of computability in anonymous networks. In *Proceedings of the 15th International Symposium on Distributed Computing (DISC).* Lecture Notes in Computer Science, vol. 2180, Springer, 33–47.

Bose, P., Morin, P., Stojmenović, I., and Urrutia, J. 2001. Routing with guaranteed delivery in ad hoc wireless networks. *Wirel. Netw. 7,* 6, 609–616.

Bottreau, A. and Mé tivier, Y. 1996. The Kronecker product and local computations in graphs. In *Proceedings of the 21st International Colloquium on Trees in Algebra and Programming (CAAP).* Lecture Notes in Computer Science, vol. 1059, Springer, 2–16.

CARDEI, M., MACCALLUM, D., CHENG, M. X., MIN, M., JIA, X., LI, D., AND DU, D.-Z. 2002. Wireless sensor networks with energy efficient organization. *J. Interconnection Netw. 3,* 3–4, 213–229.

CHÁVEZ, E., DOBREV, S., KRANAKIS, E., OPATRNY, J., STACHO, L., AND URRUTIA, J. 2006. Local construction of planar spanners in unit disk graphs with irregular transmission ranges. In *Proceedings of the 7th Latin American Theoretical Informatics Symposium (LATIN).* Lecture Notes in Computer Science, vol. 3887, Springer, 286–297.

CHLEBÍK, M. AND CHLEBÍKOVÁ , J. 2006. Approximation hardness of edge dominating set problems. *J. Combin. Optim. 11,* 3, 279–290.

COCKAYNE, E. J. AND HEDETNIEMI, S. T. 1975. Optimal domination in graphs. *IEEE Trans. Circ. Syst. 22,* 11, 855–857.

COLE, R. AND VISHKIN, U. 1986. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control 70,* 1, 32–53.

CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. 1990. *Introduction to Algorithms*. The MIT Press, Cambridge, MA.

CRYAN, M. AND MILTERSEN, P. B. 2001. On pseudorandom generators in NC^0. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS).* Lecture Notes in Computer Science, vol. 2136, Springer, 272–284.

CZYGRINOW, A., HAŃĆKOWIAK, M., KRZYWDZIŃSKI, K., SZYMAŃSKA, E., AND WAWRZYNIAK, W. 2011. Brief announcement: Distributed approximations for the semi-matching problem. In *Proceedings of the 25th International Symposium on Distributed Computing (DISC).* Lecture Notes in Computer Science, vol. 6950, Springer, 200–201.

CZYGRINOW, A., HAŃĆKOWIAK, M., AND WAWRZYNIAK, W. 2008. Fast distributed approximations in planar graphs. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC).* Lecture Notes in Computer Science, vol. 5218, Springer, 78–92.

CZYZOWICZ, J., DOBREV, S., FEVENS, T., GONZÁLEZ-AGUILAR, H., KRANAKIS, E., OPATRNY, J., AND URRUTIA, J. 2008. Local algorithms for dominating and connected dominating sets of unit disk graphs with location aware nodes. In *Proceedings of the 8th Latin American Theoretical Informatics Symposium (LATIN).* Lecture Notes in Computer Science, vol. 4957, Springer, 158–169.

CZYZOWICZ, J., DOBREV, S., GONZÁLEZ-AGUILAR, H., KRÁLOVIČ, R., KRANAKIS, E., OPATRNY, J., STACHO, L., AND URRUTIA, J. 2011. Local 7-coloring for planar subgraphs of unit disk graphs. *Theor. Comput. Sci. 412,* 18, 1696–1704.

CZYZOWICZ, J., DOBREV, S., KRANAKIS, E., OPATRNY, J., AND URRUTIA, J. 2009. Local edge colouring of Yao-like subgraphs of unit disk graphs. *Theor. Comput. Sci. 410,* 14, 1388–1400.

DIESTEL, R. 2005. *Graph Theory* 3rd Ed. Springer.

DIJKSTRA, E. W. 1974. Self-stabilizing systems in spite of distributed control. *Comm. ACM 17,* 11, 643–644.

DOLEV, S. 2000. *Self-Stabilization*. The MIT Press, Cambridge, MA.

DOYLE, P. G. AND SNELL, J. L. 1984. *Random Walks and Electric Networks*. Number 22 in The Carus Mathematical Monographs. The Mathematical Association of America, Washington, DC.

ELKIN, M. 2004. Distributed approximation: A survey. *ACM SIGACT News 35,* 4, 40–57.

EPPSTEIN, D., GALIL, Z., AND ITALIANO, G. F. 1999. Dynamic graph algorithms. In *Algorithms and Theory of Computation Handbook*, M. J. Atallah, Ed., CRC Press, Boca Raton, FL, Chapter 8.

FEIGE, U., HALLDÓRSSON, M. M., KORTSARZ, G., AND SRINIVASAN, A. 2002. Approximating the domatic number. *SIAM J. Comput. 32,* 1, 172–195.

FICH, F. AND RUPPERT, E. 2003. Hundreds of impossibility results for distributed computing. *Distrib. Comput. 16,* 2–3, 121–163.

FLORÉEN, P., HASSINEN, M., KAASINEN, J., KASKI, P., MUSTO, T., AND SUOMELA, J. 2011. Local approximability of max-min and min-max linear programs. *Theory Comput. Syst. 49,* 4, 672–697.

FLORÉEN, P., HASSINEN, M., KASKI, P., AND SUOMELA, J. 2008a. Local approximation algorithms for a class of 0/1 max-min linear programs. arXiv:0806.0282 [cs.DC].

FLORÉEN, P., HASSINEN, M., KASKI, P., AND SUOMELA, J. 2008b. Tight local approximation results for max-min linear programs. In *Proceedings of the 4th International Workshop on Algorithmic Aspects of Wireless Sensor Networks.* Lecture Notes in Computer Science, vol. 5389, Springer, 2–17.

FLORÉEN, P., KAASINEN, J., KASKI, P., AND SUOMELA, J. 2009. An optimal local approximation algorithm for max-min linear programs. In *Proceedings of the 21st Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA).* ACM Press, 260–269.

FLORÉEN, P., KASKI, P., MUSTO, T., AND SUOMELA, J. 2008c. Approximating max-min linear programs with local algorithms. In *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS).* IEEE.

FLORÉEN, P., KASKI, P., MUSTO, T., AND SUOMELA, J. 2008d. Local approximation algorithms for scheduling problems in sensor networks. In *Proceedings of the 3rd International Workshop on Algorithmic Aspects of Wireless Sensor Networks.* Lecture Notes in Computer Science, vol. 4837, Springer, 99–113.

FLORÉEN, P., KASKI, P., POLISHCHUK, V., AND SUOMELA, J. 2010. Almost stable matchings by truncating the Gale–Shapley algorithm. *Algorithmica 58,* 1, 102–118.

FLORÉEN, P., KASKI, P., AND SUOMELA, J. 2007. A distributed approximation scheme for sleep scheduling in sensor networks. In *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON).* IEEE, 152–161.

FRAIGNIAUD, P., GAVOILLE, C., ILCINKAS, D., AND PELC, A. 2007. Distributed computing with advice: Information sensitivity of graph coloring. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP).* Lecture Notes in Computer Science, vol. 4596, Springer, 231–242.

FRAIGNIAUD, P., KORMAN, A., AND PELEG, D. 2011. Local distributed decision. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science* (FOCS). IEEE Computer Society Press, Los Alamitos, CA. (to appear).

FUJITO, T. AND NAGAMOCHI, H. 2002. A 2-approximation algorithm for the minimum weight edge dominating set problem. *Discr. Appl. Math. 118,* 3, 199–207.

GABRIEL, K. R. AND SOKAL, R. R. 1969. A new statistical approach to geographic variation analysis. *Syst. Zool. 18,* 3, 259–278.

GALE, D. AND SHAPLEY, L. S. 1962. College admissions and the stability of marriage. *Amer. Math. Month. 69,* 1, 9–15.

GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York.

GASIENIEC, L., SU, C., AND WONG, P. 2008. Routing in geometric networks. In *Encyclopedia of Algorithms*, M.-Y. Kao, Ed., Springer, New York.

GIBBONS, P. B. 2008. Fun with networks: Social, sensor, and shape shifting. In *Proceedings of the* 22nd *International Symposium on Distributed Computing (DISC).*

GODSIL, C. AND ROYLE, G. 2004. *Algebraic Graph Theory*. Graduate Texts in Mathematics Series, vol. 207, Springer, New York.

GOLDBERG, A. V., PLOTKIN, S. A., AND SHANNON, G. E. 1988. Parallel symmetry-breaking in sparse graphs. *SIAM J. Discr. Math. 1,* 4, 434–446.

GÖÖS, M. AND SUOMELA, J. 2011. Locally checkable proofs. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC).* ACM Press, 159–168.

GRAHAM, R. L., ROTHSCHILD, B. L., AND SPENCER, J. H. 1980. *Ramsey Theory*. John Wiley & Sons, New York.

GUSFIELD, D. AND IRVING, R. W. 1989. *The Stable Marriage Problem: Structure and Algorithms*. Foundations of Computing. The MIT Press, Cambridge, MA.

HAJEK, B. AND SASAKI, G. 1988. Link scheduling in polynomial time. *IEEE Trans. Inf. Theory 34,* 5, 910–917.

HAŃĆKOWIAK, M., KAROŃSKI, M., AND PANCONESI, A. 1998. On the distributed complexity of computing maximal matchings. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* SIAM, 219–225.

HAŃĆKOWIAK, M., KAROŃSKI, M., AND PANCONESI, A. 2001. On the distributed complexity of computing maximal matchings. *SIAM J. Discr. Math. 15,* 1, 41–57.

HARVEY, N. J. A., LADNER, R. E., LOVÁSZ, L., AND TAMIR, T. 2006. Semi-matchings for bipartite graphs and load balancing. *J. Algor. 59,* 1, 53–78.

HASSINEN, M., KAASINEN, J., KRANAKIS, E., POLISHCHUK, V., SUOMELA, J., AND WIESE, A. 2011. Analysing local algorithms in location-aware quasi-unit-disk graphs. *Discr. Appl. Math. 159,* 15, 1566–1580.

HASSINEN, M., POLISHCHUK, V., AND SUOMELA, J. 2008. Local 3-approximation algorithms for weighted dominating set and vertex cover in quasi unit-disk graphs. In *Proceedings of the 2nd International Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks (LOCALGOS).* V.9–V.12.

HÅSTAD, J. 1987. One-way permutations in NC^0. *Inf. Process. Lett. 26,* 3, 153–155.

HOCHBAUM, D. S. 1982. Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput. 11,* 3, 555–556.

HOCKING, J. G. AND YOUNG, G. S. 1961. *Topology*. Addison-Wesley, Reading, MA.

HOEPMAN, J.-H., KUTTEN, S., AND LOTKER, Z. 2006. Efficient distributed weighted matchings on trees. In *Proceedings of the 13th International Colloquium on Structural Information and Communication Complexity (SIROCCO).* Lecture Notes in Computer Science, vol. 4056, Springer, 115–129.

Hoory, S. 2002. On graphs of high girth. Ph.D. thesis, Hebrew University, Jerusalem.

Hunt, H. B., III, Marathe, M. V., Radhakrishnan, V., Ravi, S. S., Rosenkrantz, D. J., and Stearns, R. E. 1998. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algor. 26,* 2, 238–274.

Imrich, W. and Pisanski, T. 2008. Multiple Kronecker covering graphs. *Euro. J. Combin. 29,* 5, 1116–1122.

Israeli, A. and Itai, A. 1986. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett. 22,* 2, 77–80.

Jain, K., Padhye, J., Padmanabhan, V. N., and Qiu, L. 2005. Impact of interference on multi-hop wireless network performance. *Wirel. Netw. 11,* 4, 471–487.

Jaromczyk, J. W. and Toussaint, G. T. 1992. Relative neighborhood graphs and their relatives. *Proc. IEEE 80,* 9, 1502–1517.

Johnson, D. S. 1974. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci. 9*, 256–278.

Johnson, R. E. and Schneider, F. B. 1985. Symmetry and similarity in distributed systems. In *Proceedings of the 4th Annual ACM Symposium on Principles of Distributed Computing (PODC).* ACM Press, 13–22.

Kanj, I. A., Perković, L., and Xia, G. 2008. Computing lightweight spanners locally. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC).* Lecture Notes in Computer Science, vol. 5218, Springer, 365–378.

Kaski, P., Penttinen, A., and Suomela, J. 2008. Coordinating concurrent transmissions: A constant factor approximation of maximum-weight independent set in local conflict graphs. *Ad Hoc Sensor Wirel. Netw.: Int. J. 6,* 3–4, 239–263.

Keil, J. M. and Gutwin, C. A. 1992. Classes of graphs which approximate the complete Euclidean graph. *Discr. Comput. Geom. 7,* 1, 13–28.

Korman, A. and Kutten, S. 2006. On distributed verification. In *Proceedings of the 8th International Conference on Distributed Computing and Networking (ICDCN).* Lecture Notes in Computer Science, vol. 4308, Springer, 100–114.

Korman, A. and Kutten, S. 2007. Distributed verification of minimum spanning trees. *Distrib. Comput. 20,* 4, 253–266.

Korman, A., Kutten, S., and Peleg, D. 2005. Proof labeling schemes. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC).* ACM Press, 9–18.

Korman, A., Peleg, D., and Rodeh, Y. 2010. Constructing labeling schemes through universal matrices. *Algorithmica 57*, 4, 641–652.

Korte, B. and Vygen, J. 2006. *Combinatorial Optimization: Theory and Algorithms,* 3rd Ed., Springer.

Kranakis, E., Singh, H., and Urrutia, J. 1999. Compass routing on geometric networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry* (CCCG).

Krishnamachari, B. 2005. *Networking Wireless Sensors*. Cambridge University Press, Cambridge, UK.

Kuhn, F. 2005. The price of locality: Exploring the complexity of distributed coordination primitives. Ph.D. thesis, ETH Zurich.

Kuhn, F. 2008. Local approximation of covering and packing problems. In *Encyclopedia of Algorithms*, M.-Y. Kao, Ed., Springer.

Kuhn, F. 2009. Weak graph colorings: Distributed algorithms and applications. In *Proceedings of the 21st Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA).* ACM Press, 138–144.

Kuhn, F. and Moscibroda, T. 2007. Distributed approximation of capacitated dominating sets. In *Proceedings of the 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA).* ACM Press, 161–170.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2004. What cannot be computed locally! In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC).* ACM Press, 300–309.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2005. On the locality of bounded growth. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC).* ACM Press, 60–68.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2006a. Fault-tolerant clustering in ad hoc and sensor networks. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS).* IEEE Computer Society Press.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2006b. The price of being near-sighted. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* ACM Press, 980–989.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2006c. The price of being near-sighted. Tech. rep. 229, ETH Zurich, Computer Engineering and Networks Laboratory.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2010. Local computation: Lower and upper bounds. arXiv:1011.5470 [cs.DC].

KUHN, F. AND WATTENHOFER, R. 2005. Constant-time distributed dominating set approximation. *Distrib. Comput. 17,* 4, 303–310.

KUHN, F., WATTENHOFER, R., AND ZOLLINGER, A. 2008. Ad hoc networks beyond unit disk graphs. *Wirel. Netw. 14,* 5, 715–729.

LENZEN, C. 2011. Synchronization and symmetry breaking in distributed systems. Ph.D. thesis, ETH Zurich.

LENZEN, C., OSWALD, Y. A., AND WATTENHOFER, R. 2010. What can be approximated locally? TIK rep. 331, ETH Zurich, Computer Engineering and Networks Laboratory.

LENZEN, C., SUOMELA, J., AND WATTENHOFER, R. 2009. Local algorithms: Self-stabilization on speed. In *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS).* Lecture Notes in Computer Science, vol. 5873, Springer, 17–34.

LENZEN, C. AND WATTENHOFER, R. 2008. Leveraging Linial's locality limit. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC).* Lecture Notes in Computer Science, vol. 5218, Springer, 394–407.

LENZEN, C. AND WATTENHOFER, R. 2010. Minimum dominating set approximation in graphs of bounded arboricity. In *Proceedings of the 24th International Symposium on Distributed Computing (DISC).* Lecture Notes in Computer Science, vol. 6343, Springer, 510–524.

LI, X.-Y., WANG, Y., AND SONG, W.-Z. 2004. Applications of k-local MST for topology control and broadcasting in wireless ad hoc networks. *IEEE Trans. Parall. Distrib. Syst. 15,* 12, 1057–1069.

LINIAL, N. 1992. Locality in distributed graph algorithms. *SIAM J. Comput. 21,* 1, 193–201.

LOVÁSZ, L. 2008. Very large graphs. arXiv:0902.0132 [math.CO].

LUBY, M. 1986. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput. 15,* 4, 1036–1053.

LUBY, M. 1993. Removing randomness in parallel computation without a processor penalty. *J. Comput. Syst. Sci. 47,* 2, 250–286.

LUBY, M. AND NISAN, N. 1993. A parallel approximation algorithm for positive linear programming. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC).* ACM Press, 448–457.

LYNCH, N. A. 1989. A hundred impossibility proofs for distributed computing. In *Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing (PODC).* ACM Press, 1–28.

MAYER, A., NAOR, M., AND STOCKMEYER, L. 1995. Local computations on static and dynamic graphs. In *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS).* IEEE, 268–278.

MITZENMACHER, M. AND UPFAL, E. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, Cambridge, UK.

MOSCIBRODA, T. 2006. Locality, scheduling, and selfishness: Algorithmic foundations of highly decentralized networks. Ph.D. thesis, ETH Zurich.

MOTWANI, R. AND RAGHAVAN, P. 1995. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK.

MUNKRES, J. R. 2000. *Topology* 2nd Ed., Prentice Hall, Upper Saddle River, NJ.

NAOR, M. 1991. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM J. Discr. Math. 4,* 3, 409–412.

NAOR, M. AND STOCKMEYER, L. 1995. What can be computed locally? *SIAM J. Comput. 24,* 6, 1259–1277.

NGUYEN, H. N. AND ONAK, K. 2008. Constant-time approximation algorithms via local improvements. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS).* IEEE Computer Society Press, 327–336.

PAPADIMITRIOU, C. H. AND STEIGLITZ, K. 1998. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., Mineola, NY.

PAPADIMITRIOU, C. H. AND YANNAKAKIS, M. 1991. On the value of information in distributed decision-making. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 61–64.

PAPADIMITRIOU, C. H. AND YANNAKAKIS, M. 1993. Linear programming without the matrix. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 121–129.

PARNAS, M. AND RON, D. 2007. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci. 381,* 1–3, 183–196.

PELEG, D. 2000. *Distributed Computing – A Locality-Sensitive Approach*. SIAM, Philadelphia, PA.

POLISHCHUK, V. AND SUOMELA, J. 2009. A simple local 3-approximation algorithm for vertex cover. *Inf. Process. Lett. 109,* 12, 642–645.

RAMSEY, F. P. 1930. On a problem of formal logic. *Proc.London Math. Soc. 30*, 264–286.

SCHNEIDER, J. AND WATTENHOFER, R. 2008. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proceedings of the 27th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 35–44.

SCHNEIDER, M. 1993. Self-stabilization. *ACM Comput. Surv. 25,* 1, 45–67.

ŠPARL, P. AND ŽEROVNIK, J. 2005. 2-local 4/3-competitive algorithm for multicoloring hexagonal graphs. *J. Algor. 55,* 1, 29–41.

STERLING, A. D. 2008. A limit to the power of multiple nucleation in self-assembly. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC)*. Lecture Notes in Computer Science, vol. 5218, Springer, 451–465.

SUOMELA, J. 2010. Distributed algorithms for edge dominating sets. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 365–374.

SUOMELA, J. 2009. Optimisation problems in wireless sensor networks: Local algorithms and local graphs. Ph.D. thesis, Department of Computer Science, University of Helsinki, Helsinki, Finland.

TOUSSAINT, G. T. 1980. The relative neighbourhood graph of a finite planar set. *Pattern Recogn. 12,* 4, 261–268.

URRUTIA, J. 2007. Local solutions for global problems in wireless networks. *J. Discr. Algor. 5,* 3, 395–407.

VAZIRANI, V. V. 2001. *Approximation Algorithms*. Springer, Berlin.

WANG, Y. AND LI, X.-Y. 2006. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Netw. Appl. 11,* 2, 161–175.

WATTENHOFER, M. AND WATTENHOFER, R. 2004. Distributed weighted matching. In *Proceedings of the 18th International Symposium on Distributed Computing (DISC)*. Lecture Notes in Computer Science, vol. 3274, Springer, 335–348.

WATTENHOFER, R. AND ZOLLINGER, A. 2004. XTC: A practical topology control algorithm for ad-hoc networks. In *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society Press.

WEICHSEL, P. M. 1962. The Kronecker product of graphs. *Proc. Amer. Math. Soc. 13,* 1, 47–52.

WIESE, A. 2007. Local approximation algorithms in unit disk graphs. M.S. thesis, Technische Universität Berlin.

WIESE, A. AND KRANAKIS, E. 2008a. Impact of locality on location aware unit disk graphs. *Algor. 1*, 2–29.

WIESE, A. AND KRANAKIS, E. 2008b. Local maximal matching and local 2-approximation for vertex cover in UDGs. In *Proceedings of the 7th International Conference on Ad-Hoc Networks & Wireless*. Lecture Notes in Computer Systems, vol. 5198, Springer, 1–14.

WIESE, A. AND KRANAKIS, E. 2009a. Local construction and coloring of spanners of location aware unit disk graphs. *Discr. Math., Algor. Appl. 1,* 4, 555–588.

WIESE, A. AND KRANAKIS, E. 2009b. Local PTAS for dominating and connected dominating set in location aware unit disk graphs. In *Proceedings of the 6th Workshop on Approximation and Online Algorithms (WAOA)*. Lecture Notes in Computer Science, vol. 5426, Springer, 227–240.

WIESE, A. AND KRANAKIS, E. 2009c. Local PTAS for independent set and vertex cover in location aware unit disk graphs. *Ad Hoc Sensor Wirel. Netw. Int. J. 7,* 3–4, 273–293.

YAMASHITA, M. AND KAMEDA, T. 1996. Computing on anonymous networks: Part I – Characterizing the solvable cases. *IEEE Trans. Parallel Distrib. Syst. 7,* 1, 69–89.

YANNAKAKIS, M. AND GAVRIL, F. 1980. Edge dominating sets in graphs. *SIAM J. Appl. Math. 38,* 3, 364–372.

YAO, A. C.-C. 1982. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput. 11,* 4, 721–736.

ZOLLINGER, A. 2008. Geographic routing. In *Encyclopedia of Algorithms*, M.-Y. Kao, Ed., Springer.

Received February 2009; revised January 2011; accepted October 2011