A simple local 3-approximation algorithm for vertex cover

Valentin Polishchuk, Jukka Suomela

Helsinki Institute for Information Technology HIIT
Helsinki University of Technology and University of Helsinki
P.O. Box 68, FI-00014 University of Helsinki, Finland

Abstract

We present a local algorithm (constant-time distributed algorithm) for finding a 3-approximate vertex cover in bounded-degree graphs. The algorithm is deterministic, and no auxiliary information besides port numbering is required.

Key words: approximation algorithms, distributed computing, graph algorithms, local algorithms

1. Introduction

Given a graph $\mathcal{G} = (V, E)$, a subset of nodes $C \subseteq V$ is a *vertex cover* if each edge $\{u, v\} \in E$ has $u \in C$ or $v \in C$. In this work, we present a constant-time distributed algorithm for finding a factor 3 approximation for minimum vertex cover in bounded-degree graphs.

A distributed algorithm that runs in constant time (constant number of synchronous communication rounds) is called a *local algorithm* [14]. In a local algorithm, the output of a node is a function of the input that is available within its constant-radius neighbourhood; this implies not only high scalability but also high fault-tolerance, making local algorithms desirable for real-world large-scale distributed systems.

Unfortunately, to date most results on local algorithms have been negative, even if we use Linial's [12] model of distributed computing where the message size is unbounded and local computation is free. Linial's [12] seminal work shows that there is no local algorithm for finding a maximal independent set, maximal matching, or 3-colouring of an n-cycle. This holds even if each node is assigned a unique identifier from the set $\{1, 2, \ldots, n\}$. Randomness does not help either; more generally, Naor and Stockmeyer [14] show that randomness does not help in so-called locally checkable labellings; maximal matching in a bounded-degree graph is an example of such a problem.

Kuhn et al. [8, 9] show that there is no local, constant-factor approximation algorithm for minimum vertex cover, minimum dominating set, or maximum matching in general graphs (without a degree bound). For more negative results, see Czygrinow et al. [2] and Lenzen and Wattenhofer [11].

Prior positive results on local algorithms for combinatorial problems typically rely on randomness, and the approximation guarantees only hold in expectation or with high probability. Examples include randomised local algorithms for weighted matching in trees [7, 16] and for finding a maximum independent set in a planar graph [2]. A general framework for approximating covering and packing problems by local algorithms is based on solving the LP relaxation and applying randomised rounding [9].

Another line of research has studied local algorithms in a setting where auxiliary information is available. For example, if each node in a unit-disk graph knows its coordinates, then there is a local $(1 + \epsilon)$-approximation for vertex cover [17].

However, without randomness or auxiliary information, positive results are scarce. Some deterministic local algorithms exist for linear programs [3, 4, 9, 15], but very few are known for combinatorial problems – in the light of strong negative results, this is not particularly surprising. Naor

Email addresses: valentin.polishchuk@cs.helsinki.fi (Valentin Polishchuk), jukka.suomela@cs.helsinki.fi (Jukka Suomela)

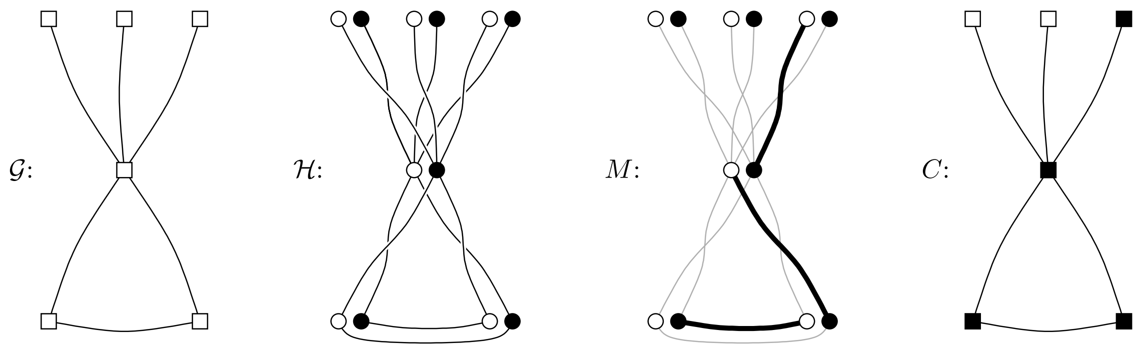

Figure 1: Algorithm overview.

and Stockmeyer [14] give a deterministic local algorithm for so-called weak colouring in graphs of odd degree. Lenzen et al. [10] present a deterministic local 74-approximation algorithm for minimum dominating set in planar graphs. Vertex covering in bounded-degree graphs is known to admit a local constant-factor approximation; however, prior algorithms rely on linear programming techniques, either LP approximation schemes and rounding [9] or primal-dual approaches [13].

In this work, we give a new example of a simple, deterministic, constant-time, constant-factor approximation algorithm for vertex covering. The algorithm is purely combinatorial; in particular, it does not resort to an LP approximation scheme and rounding. Our result is summarised in the following theorem.

Theorem 1. *A 3-approximation for minimum vertex cover in a bounded-degree graph can be found by a deterministic local algorithm in $2\Delta + 1$ communication rounds, where Δ is the maximum degree of the graph. The algorithm does not need unique node identifiers; port numbering is sufficient.*

By *port numbering* [1] we mean that each node of $\mathcal{G}$ imposes an ordering on its adjacent edges. Port numbering without any unique identifiers is an extremely weak assumption. For example, it does not help to break the symmetry in an n-cycle or an n-clique: in the worst case, every node is bound to make the same decision. In spite of that, we show that even in this very restricted model, it is possible to approximate the vertex cover to within a factor of 3, which is not much worse than what can be obtained in a centralised setting by the best known polynomial-time approximation algorithms.

One explanation for this positive result is the following. Indeed, we cannot break the symmetry in a symmetric graph. However, in a symmetric graph – or, more generally, in a regular graph – the trivial choice of all nodes is a factor 2 approximation for vertex cover. Hence the instances that require a nontrivial choice are exactly those which cannot be entirely symmetric; there must be variation in the node degrees.

The only assumption that we make is some constant upper bound on the degree of the nodes. This is unavoidable, if we want a constant-time, constant-factor approximation algorithm for vertex cover [8].

2. Overview

To obtain a 2-approximation for vertex cover in a centralised setting, one could simply find a maximal matching $M \subseteq E$ and output all matched nodes. Unfortunately, Linial's [12] lower bound shows that the same technique cannot be applied in a local setting: even if unique node identifiers are available, we cannot find a maximal matching. However, Hańćkowiak et al. [5] show, in passing, that if the input graph is 2-coloured (not only bipartite but also each node knows its part) then it is possible to overcome Linial's bound. Their distributed algorithm for maximal matching uses a subroutine called *LowDegreeMatch*; this subroutine is a local algorithm for finding a maximal matching in bounded-degree 2-coloured graphs.

How does this result help us though if we want to find a vertex cover in general (not 2-coloured) graphs? The idea is illustrated in Figure 1. Given the graph $\mathcal{G}$, we replace each node with two copies, a black copy and a white copy. If the nodes u and v are adjacent in the original graph, then the black copy of u is adjacent to the white copy of v in the new graph, and vice versa. We obtain a bipartite, 2-coloured graph $\mathcal{H}$. Now we can apply a local algorithm to find a maximal matching M in the graph $\mathcal{H}$. The vertex cover C for $\mathcal{G}$ consists of those nodes whose black copy or white copy (or both) were matched in $\mathcal{H}$. This turns out to be within factor 3 of the optimum, because the edges of the matching in $\mathcal{H}$ form a set of cycles and paths in $\mathcal{G}$. We present the full algorithm in detail in Section 3, and we prove the approximation guarantee in Section 4.

Formally, $\mathcal{H}$ is a covering graph of $\mathcal{G}$. More specifically, $\mathcal{H}$ is the bipartite double cover of $\mathcal{G}$, also known as the Kronecker double cover and canonical double cover; it is the Kronecker product $\mathcal{G} \times K_2$. Usually, covering graphs are used in the field of distributed computing to prove impossibility results [1]; our work uses them for algorithm design. The same approach – finding a maximal matching in a bipartite double cover – has been used previously by Hańćkowiak et al. [6] as a subroutine in a non-local distributed algorithm.

3. Algorithm

We describe the local algorithm that finds a vertex cover $C \subseteq V$.

In the port numbering model, it is assumed that each node $v \in V$ knows its own degree $d(v) \leq \Delta$. The node has $d(v)$ ports, each leading to one of its neighbours; the ports are numbered in an arbitrary order by $1, 2, \ldots, d(v)$. A node can send a message to a given port, and the respective neighbour can receive it on the next time step.

The node $v \in V$ maintains the following variables: $a(v)$ and $b(v)$ are two chosen neighbours (identified by port numbers), and $i(v)$ is a counter. The output of the node is $c(v) \in \{\text{true}, \text{false}\}$ which determines whether $v \in C$ or not.

Initially, $a(v) = \bot$, $b(v) = \bot$, $i(v) = 0$, and $c(v) = \text{false}$.

On an odd time step, each node $v \in V$ performs the following read–compute–write cycle.

1. If $a(v) = \bot$ and $1 \leq i(v) \leq d(v)$, then receive a message m from the port $i(v)$. If $m =$ 'accept' then $a(v) \leftarrow i(v)$ and $c(v) \leftarrow \text{true}$.
2. If $a(v) = \bot$ and $i(v) \leq d(v)$ then $i(v) \leftarrow i(v) + 1$.
3. If $a(v) = \bot$ and $i(v) \leq d(v)$ then send the message 'propose' to the port $i(v)$.

On an even time step, each node $v \in V$ performs the following read–compute–write cycle.

1. Receive messages from all neighbours.
2. For each j such that a message 'propose' was received from the port j, in increasing order:
   (a) If $b(v) = \bot$ then send the message 'accept' to the port j. Set $b(v) \leftarrow j$ and $c(v) \leftarrow$ true.
   (b) Otherwise, send the message 'reject' to the port j.

Clearly, after $2\Delta + 1$ time steps, the algorithm stops, as no messages are sent any more.

4. Analysis

Let us first show that the set $C = \{v \in V : c(v) = \text{true}\}$ is a vertex cover when the algorithm stops. Consider an arbitrary edge $e = \{u, v\} \in E$. If $a(u) \neq \bot$, then $c(u) = \text{true}$. Otherwise u has sent a 'propose' message to v, and v has sent a 'reject' message; hence $b(v) \neq \bot$ and $c(v) = \text{true}$. We conclude that C covers the edge e.

Let us now establish the approximation ratio. Let C^* be a minimum vertex cover.

Let $v \in V$ be such that $a(v) \neq \bot$. Then the port $a(v)$ in v leads to a node $u \in V$ such that $b(u) \neq \bot$. Furthermore, the port $b(u)$ in u leads back to the node v. We say that u and v form a *pair*.

Let $P \subseteq E$ consist of all edges $\{u, v\}$ such that u and v form a pair and consider the subgraph $\mathcal{G}_1 = (V, P)$ of $\mathcal{G}$. We make the following observations.

1. The degree of a node $v \in V$ in $\mathcal{G}_1$ is at most 2. Indeed, at most one of its neighbours is determined by $a(v)$, and at most one of its neighbours is determined by $b(v)$.
2. The set of non-isolated nodes (nodes with degree at least 1) in $\mathcal{G}_1$ is equal to the set C.

Discard the isolated nodes to obtain the subgraph $\mathcal{G}_2 = (C, P)$ of $\mathcal{G}$. Each connected component of $\mathcal{G}_2$ is a path or a cycle, and there are no isolated nodes.

Consider an arbitrary connected component $\mathcal{C}$ of $\mathcal{G}_2$. Either $\mathcal{C}$ is a path $\mathcal{P}$, or we can remove one edge arbitrarily to obtain a path. The paths form a partition of the cover C; each $v \in C$ belongs to exactly one such path.

Let $m \geq 1$ be the number of edges on the path $\mathcal{P}$. As $\mathcal{P}$ is a subgraph of $\mathcal{G}$, each edge of $\mathcal{P}$ must have at least one endpoint in the optimal cover C^*. Hence at least $\lceil m/2 \rceil$ nodes of $\mathcal{P}$ are in C^*, which is at least a fraction 1/3 of the total number of nodes in $\mathcal{P}$ (the worst case being $m = 2$).

Summing over all paths, we conclude that $|C| \leq 3|C^*|$. This completes the proof of Theorem 1.

5. Discussion

Textbooks and introductory courses on distributed algorithms mention hardly any results related to constant-time distributed algorithms. One of the obstacles has been the lack of examples of algorithms that are sufficiently simple to be explained to a non-expert audience, yet not completely trivial. The present work is a step in this direction; further work is needed to find new interesting examples of simple local algorithms.

Acknowledgements

We thank Patrik Floréen and Petteri Kaski for discussions and comments. This research was supported in part by the Academy of Finland, Grants 116547 and 118653 (ALGODAN), by Helsinki Graduate School in Computer Science and Engineering (Hecse), and by the Foundation of Nokia Corporation.

References

[1] D. Angluin, Local and global properties in networks of processors, in: Proc. 12th Symposium on Theory of Computing (STOC 1980), ACM Press, 1980, pp. 82–93. doi:10.1145/800141.804655.

[2] A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak, Fast distributed approximations in planar graphs, in: Proc. 22nd Symposium on Distributed Computing (DISC 2008), Vol. 5218 of LNCS, Springer, 2008, pp. 78–92. doi:10.1007/978-3-540-87779-0_6.

[3] P. Floréen, M. Hassinen, P. Kaski, J. Suomela, Tight local approximation results for max-min linear programs, in: Proc. 4th Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors 2008), Vol. 5389 of LNCS, Springer, 2008, pp. 2–17. doi:10.1007/978-3-540-92862-1_2.

[4] P. Floréen, P. Kaski, T. Musto, J. Suomela, Approximating max-min linear programs with local algorithms, in: Proc. 22nd International Parallel and Distributed Processing Symposium (IPDPS 2008), IEEE, 2008. doi:10.1109/IPDPS.2008.4536235.

[5] M. Hańćkowiak, M. Karoński, A. Panconesi, On the distributed complexity of computing maximal matchings, in: Proc. 9th Symposium on Discrete Algorithms (SODA 1998), SIAM, 1998, pp. 219–225.

[6] M. Hańćkowiak, M. Karoński, A. Panconesi, On the distributed complexity of computing maximal matchings, SIAM Journal on Discrete Mathematics 15 (1) (2001) 41–57. doi:10.1137/S0895480100373121.

[7] J.-H. Hoepman, S. Kutten, Z. Lotker, Efficient distributed weighted matchings on trees, in: Proc. 13th Colloquium on Structural Information and Communication Complexity (SIROCCO 2006), Vol. 4056 of LNCS, Springer, 2006, pp. 115–129. doi:10.1007/11780823_10.

[8] F. Kuhn, T. Moscibroda, R. Wattenhofer, What cannot be computed locally!, in: Proc. 23rd Symposium on Principles of Distributed Computing (PODC 2004), ACM Press, 2004, pp. 300–309. doi:10.1145/1011767.1011811.

[9] F. Kuhn, T. Moscibroda, R. Wattenhofer, The price of being near-sighted, in: Proc. 17th Symposium on Discrete Algorithms (SODA 2006), ACM Press, 2006, pp. 980–989. doi:10.1145/1109557.1109666.

[10] C. Lenzen, Y. A. Oswald, R. Wattenhofer, What can be approximated locally?, in: Proc. 20th Symposium on Parallel Algorithms and Architectures (SPAA 2008), ACM Press, 2008, pp. 46–54. doi:10.1145/1378533.1378540.

[11] C. Lenzen, R. Wattenhofer, Leveraging Linial's locality limit, in: Proc. 22nd Symposium on Distributed Computing (DISC 2008), Vol. 5218 of LNCS, Springer, 2008, pp. 394–407. doi:10.1007/978-3-540-87779-0_27.

[12] N. Linial, Locality in distributed graph algorithms, SIAM Journal on Computing 21 (1) (1992) 193–201. doi:10.1137/0221015.

[13] T. Moscibroda, Locality, scheduling, and selfishness: Algorithmic foundations of highly decentralized networks, Ph.D. thesis, ETH Zürich (2006).

[14] M. Naor, L. Stockmeyer, What can be computed locally?, SIAM Journal on Computing 24 (6) (1995) 1259–1277. doi:10.1137/S0097539793254571.

[15] C. H. Papadimitriou, M. Yannakakis, Linear programming without the matrix, in: Proc. 25th Symposium on Theory of Computing (STOC 1993), ACM Press, 1993, pp. 121–129. doi:10.1145/167088.167127.

[16] M. Wattenhofer, R. Wattenhofer, Distributed weighted matching, in: Proc. 18th Symposium on Distributed Computing (DISC 2004), Vol. 3274 of LNCS, Springer, 2004, pp. 335–348. doi:10.1007/b101206.

[17] A. Wiese, E. Kranakis, Local PTAS for independent set and vertex cover in location aware unit disk graphs, in: Proc. 4th Conference on Distributed Computing in Sensor Systems (DCOSS 2008), Vol. 5067 of LNCS, Springer, 2008, pp. 415–431. doi:10.1007/978-3-540-69170-9_28.