J. Parallel Distrib. Comput. 74 (2014) 2193–2202

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Distributed algorithm for the maximal 2-packing in geometric outerplanar graphs

Joel Antonio Trejo-Sánchez [a,b,*], José Alberto Fernández-Zepeda [a]

[a] Department of Computer Science, Center for Scientific Research and Higher Education of Ensenada (CICESE), Ensenada 22860, B.C., Mexico
[b] Department of Basic Sciences, Universidad del Caribe, Cancun 77528, Quintana Roo, Mexico

HIGHLIGHTS

- We propose a distributed algorithm for the maximal 2-packing set in a geometric outerplanar graph.
- The execution time of this algorithm is $O(n)$ steps.
- The algorithm has three phases: leader election, graph exploration, and vertex coloring.
- The vertex coloring phase resembles an ear decomposition of the input graph.
- When the input graph is a ring, the algorithm computes the maximum 2-packing set.

ARTICLE INFO

Article history:
Received 8 March 2013
Received in revised form
30 November 2013
Accepted 5 December 2013
Available online 19 December 2013

Keywords:
Distributed algorithm
Geometric graph
Outerplanar graph
Ear decomposition
2-packing set

ABSTRACT

In this paper, we present a deterministic distributed algorithm that computes the maximal 2-packing set in a geometric outerplanar graph. In a geometric outerplanar graph, all the vertices have location coordinates in the plane and lie on the boundary of the graph. Our algorithm consists of three phases. First, it elects a vertex as the leader. Second, it explores the graph to determine relevant information about the structure of the input graph. Third, with this information, it computes a maximal 2-packing set. When the input graph is a ring, the algorithm computes a maximum 2-packing set. The execution time of this algorithm is $O(n)$ steps and it uses $O(n \log n)$ messages. This algorithm does not require knowledge of the size of the input graph. To the best of our knowledge, this is the first deterministic distributed algorithm that solves such a problem for a geometric outerplanar graph in a linear number of steps.

1. Introduction

A *distributed system* is a collection of independent processors that communicate with each other using a computer network. Such processors cooperate to reach a common global goal. Graphs can naturally model distributed systems and researchers have studied and solved many graph problems related to these systems. This paper deals with the design of a deterministic distributed algorithm that solves the maximal 2-packing set problem (M2PS) in geometric outerplanar graphs.

A very well-known concept in graph theory is the independent set. Let $G = (V, E)$ be an undirected connected graph, where V and E are the set of vertices and edges, respectively. A subset $I \subseteq V$ is an *independent set* of G, if any arbitrary pair of vertices $u, v \in I$ are not neighbors in G. A 2-packing set is an independent set with further restrictions; in particular, a 2-*packing set* [14,10] is a subset $S \subseteq V$, such that the length of the shortest path between any pair of vertices $u, v \in S$ is at least three (some authors refer S as the *strong stable set* [16,7]). More generally, a k-*packing set* is a subset $L \subseteq V$ such that the length of the shortest path between any pair of vertices in L is at least $k + 1$ (in this sense, the independent set is also known as the 1-packing set). Most of the research in this area focuses on the maximal independent set (MIS). Some relevant distributed algorithms for finding a MIS are [25,5,21,6,27,19].

The fastest distributed algorithms to solve the MIS problem are the ones presented by Panconesi and Srinivasan [25] (for general graphs) that run in $O(2^{O(\sqrt{\log n})})$ rounds; by Barenboim and Elkin [5] and Kuhn [21] (for low degree graphs) that run in $O(\Delta + \log^* n)$ rounds, where Δ is the maximum degree in G; and Barenboim and by Elkin [6] (for bounded arboricity graphs) that run in $O(\log n / \log \log n)$ rounds. All these algorithms have sublinear running time and most of them use the network decomposition

* Corresponding author at: Department of Basic Sciences, Universidad del Caribe, Cancun 77528, Quintana Roo, Mexico.
E-mail addresses: jtrejo@ucaribe.edu.mx, jtrejo@cicese.mx (J.A. Trejo-Sánchez), fernan@cicese.mx (J.A. Fernández-Zepeda).

0743-7315/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jpdc.2013.12.002

technique [4]. (This technique generates a partition of the vertices of the input graph and generates a set of clusters of specific characteristics. Certain graph problems can be partially solved, independently and in parallel, in each cluster. Then, by combining the partial solutions of the clusters, it is possible to generate a solution for the original problem. This is the approach used by many sublinear running time algorithms for the MIS.) All these algorithms require that vertices know the number n of vertices in the input graph or at least an upper bound on this parameter. This is a strong assumption for certain types of networks (e.g. dynamic networks), since computing the number of vertices in a network requires $\Omega(n)$ time units [3]. In our paper, we do not assume that vertices know any of these parameters.

Finding a 2-packing set in a graph is useful in applications that require mutual exclusion in the vertices at a distance two. One example is the frequency assignment problem [13], in which the assignment should avoid co-channel interference. Finding a M2PS is also useful as a subroutine in algorithms that solve more complex problems and that require ensuring mutual exclusion among vertices with overlapping neighborhoods [11,15]. A 2-packing set S is *maximum* when S is the largest cardinality 2-packing set in G. Finding a maximum 2-packing set is an NP-hard problem [16]. A simpler problem is finding the M2PS. A 2-packing set S is *maximal* if there does not exist a 2-packing set S' such that $S \subset S'$.

There are many similarities between the MIS (maximal 1-packing set) and the M2PS problems. Although there exist some distributed algorithms that run in sublinear time for the first problem for general graphs, there does not exist, as far as we know, any sublinear algorithm for the second problem for general graphs. One issue that hinders this implementation is that for the M2PS problem, vertices need to have information of other vertices at a distance 2. Gathering this information can be expensive for graphs that do not have constant degree. For this reason, at least from the theoretical point of view, computing a M2PS seems to be more difficult that computing a MIS.

One easy way to compute a M2PS for a graph G by using the MIS algorithm of [6] is the following. First, compute the graph G^2; second, execute the MIS algorithm of [6] on G^2. The resulting MIS is a M2PS of G. Notice that this algorithm runs in sublinear time only if G has constant degree. For high degree graphs (or even some outerplanar graphs), G^2 can be a dense graph and computing all its edges requires $\Omega(n^2)$ work. For this reason, the execution time of this procedure is no longer sublinear when using n processors.

When the value of n is unknown, many of the current MIS algorithms cannot directly be used to compute a M2PS. However, there exist some methods that make it possible to execute these algorithms without the requirement of knowing n. Korman et al. [19] provided a method that eliminates, in distributed algorithms, the requirement of knowing n (with no overhead in time complexity). The main disadvantage of this method is that it assumes that the size of any message is unbounded (which is a very unrealistic assumption). The algorithms of [25,5,6] can use this method to eliminate the requirement of knowing n or a polynomial estimate of n. However, this implies an overhead of $\Omega(\Delta^2)$ to transform the size of messages to $O(\log n)$ bits (since the method of [19] requires that some vertices read information of other vertices at a distance 2). Remember that Δ can be as big as $O(n)$ for outerplanar graphs. Schneider and Wattenhofer [27] designed an optimal distributed algorithm for computing a MIS in $O(\log^* n)$ rounds for bounded independence graphs, without the requirement that vertices know the value of n. For general graphs, this algorithm can be as slow as $O(n)$ rounds. Note that outerplanar graphs are not necessarily bounded independence graphs.

Even if the value of n is known, a procedure similar to the one proposed by [6] to compute a M2PS would fail during the process of combining the partial solutions of the clusters (since some vertices with high degree would need to read information at a distance 2 in other clusters).

There exist relevant results to compute a M2PS in the self-stabilizing context (self-stabilization is a property of some distributed systems in which the system guarantees to converge to a legal state in a finite number of steps, regardless of the initial state and will remain in a legal state in the absence of faults [9]). Gairing et al. [10] presented a self-stabilizing algorithm for finding the M2PS in an arbitrary graph; this algorithm converges to a legitimate state in an exponential number of steps. More recently, Shi [28] presented a self-stabilizing algorithm to compute a M2PS in an arbitrary graph in $O(n^2)$ rounds. Turau [32] presented a generalization of the distance-2 model in self-stabilizing systems, which makes it possible to find a M2PS in $\Omega(n^2)$ rounds. Recently, Trejo-Sánchez and Fernández-Zepeda [30] presented a self-stabilizing algorithm that computes a M2PS in a cactus graph in $O(D)$ rounds, where D is the diameter of the cactus.

Very few algorithms focus on finding the maximum 2-packing set. Mjelde [23] designed a self-stabilizing dynamic programming algorithm to compute a maximum 2-packing set in a tree that converges in $O(n^3)$ steps. The algorithm in [30] computes the maximum 2-packing set when the input cactus is a ring.

There exist other works that focus on finding the maximal k-packing set. Manne and Mjelde [22] proposed a self-stabilizing algorithm that finds a maximal k-packing for an arbitrary graph in an exponential number of steps. Goddard et al. [12] defined a framework to access vertex information at a distance k which, if applied to an outerplanar graph, might find a M2PS in $\Omega(n^3)$ steps. Note that all these results are for the self-stabilizing paradigm.

A graph G is a *plane graph* if we can draw it in such a way that two edges only meet in the vertices. Given a plane graph G, their *faces* are the regions bounded by the edges of G. Let G be a plane graph, G is also *outerplanar* if all of its vertices lie on the boundary of the graph. We refer to the region outside the graph as the *outerface*. Outerplanar graphs have attracted a lot of attention, since some computational problems that are NP-hard for arbitrary graphs can be solved in polynomial time in outerplanar graphs [17,33]. Note that trees, rings and cacti are subclasses of the outerplanar graphs.

The algorithm that we propose in this paper receives as input an outerplanar geometric graph. *Geometric graphs* assume that their vertices know their coordinates in the plane. Edges are straight lines connecting any pair of vertices that can communicate with each other. Kranakis et al. [20] use geometric graphs to discover routes in an undirected graph. Chávez et al. [8] presented an algorithm for discovering routes for directed geometric outerplanar graphs. Some of the ideas behind the exploration of the geometric outerplanar graph in our algorithm are also present in these papers. We use the location information of geometric graphs to simplify the exploration of the input graph.

The results obtained by using geometric graphs have proved to be helpful for the solution of various combinatorial and computation geometry problems [24]. Knowing the location of each vertex in the plane is a realistic assumption because this information is now easily available by the use of Geographic Positioning Systems (GPS).

Although our algorithm solves a problem that is defined for an undirected graph, we model the distributed system where it runs as a directed graph $D = (V, A)$, where V and A are the set of vertices and directed edges, respectively, and $|V| = n$. Each vertex of the graph represents a processor and each directed edge a communication link between processors. We assume that processors communicate with each other by using the *message passing model* [2] in which each processor communicates with its neighbors by sending messages over communication channels. We assume that each vertex u has a unique identifier of $O(\log n)$ bits and it knows its own location and the locations of all its neighbors. The message length