Algorithmica (2011) 60: 593–608
DOI 10.1007/s00453-010-9476-z

On the Planarization of Wireless Sensor Networks

Fenghui Zhang · Anxiao (Andrew) Jiang · Jianer Chen

Received: 3 December 2008 / Accepted: 22 November 2010 / Published online: 14 December 2010

Abstract Network planarization has been an important technique in numerous sensornet protocols—such as Greedy Perimeter Stateless Routing (GPSR), topology discovery, data-centric storage, etc.—however the planarization process itself has been difficult. Known efficient planarization algorithms exist only for restrictive wireless network models: unit-disk graphs with accurately known location information. In this paper, we study efficient planarization of wireless sensor networks, and present a novel planarization method for a more general network model, where sensors can have non-uniform transmission ranges and no location information is needed. Our planarization algorithms also include a $(2+\epsilon)$-approximation algorithm and an FPT algorithm for the BIPARTITE PLANARIZATION problem.

Keywords Sensor network · Network planarization · Graph drawing · Bipartite planarization · Approximation algorithm · Parameterized algorithm

The work by F. Zhang was supported in part by the Texas Advanced Research Program under Grant No. 003581-0006-2006, the work by A. Jiang was supported in part by the National Science Foundation under the Grant CCF-0747415, and the work by J. Chen was supported in part by the National Science Foundation under the Grant CCF-0917288 and by the Texas Advanced Research Program under Grant No. 003581-0006-2006. A preliminary version of the paper (under the title *Robust Planarization of Unlocalized Wireless Sensor Networks*) has appeared in Proc. 27th IEEE INFOCOM, Phoenix, AZ, April 2008.

F. Zhang (✉) · A.(A.) Jiang · J. Chen
Department of Computer Science & Engineering, Texas A&M University, College Station, TX 77843, USA
e-mail: fhzhang@cs.tamu.edu

A.(A.) Jiang
e-mail: ajiang@cs.tamu.edu

J. Chen
e-mail: chen@cs.tamu.edu

1 Introduction

Wireless sensor networks usually need very efficient network protocols due to the limited communication and computation capabilities of small sensors. Therefore, it is important to exploit the special topological properties of sensornets for network functions. A common observation is that the topology of a wireless sensornet usually has a strong correlation with the geometry of the sensor field. This observation has been used in notable applications like geographic routing [12, 13], etc. In these applications, network planarization has become a very important technique, because a well planarized network not only exhibits the geometric properties of the sensor field, but can also be efficiently utilized in network protocols.

The objective of network planarization is to get a connected planar subgraph of the network that contains all the nodes of the network. To well reflect the geometry of the sensor field, the planar subgraph should contain many links, so that the hop distance (or communication distance) between nodes does not change a lot after planarization. Network planarization first became the foundation of several well known geographic routing protocols, including GPSR [12, 13], Compass Routing [14], GOAFR [15], etc. Such protocols use the faces in the planar subgraph to perform perimeter routing, which guarantees packet delivery. Since then, network planarization has also become a fundamental tool in numerous other applications, including data-centric storage [18], network localization [16] and topology discovery [9, 10, 19]. Here the data-centric storage schemes use the planar graph to help determine on which set of nodes to store each datum, as well as for routing; the network localization schemes can use the properties of planar graphs to facilitate localization; and the topology discovery schemes can use the faces of the planar graph to recognize and locate boundaries and holes in the sensor field. Discovering boundaries and holes in a sensor field is useful for understanding the collected sensor data (because the meaning of sensor data often depends on the type of physical environment where they are collected), for understanding the sensing environment (e.g., building floor plan, transportation network, lakes) and detecting events (e.g., fire in a forest), and for load-balanced routing.

Although network planarization has been proven to be an excellent technique for sensor network protocols, an important problem remains: network planarization itself is difficult. So far, efficient planarization algorithms exist only for very restrictive models: unit-disk graphs with accurately known measurements related to the nodes' physical locations. The measurements are the nodes' positions, the angles between all adjacent links, or the lengths of all links. A unit-disk graph (UDG) is a graph where two nodes have a link between them if and only if their physical distance is at most one. So it corresponds to a network where the transmission ranges are the same for all nodes and in all directions. For unit-disk graphs with accurate location measurements, network planarization can be performed efficiently in a distributed fashion based on Gabriel-graph, Relative-Neighborhood graph or Delaunay graph [3]. When the network knows the accurate node positions and is very similar to the unit-disk graph,—specifically, when it is the so called $\sqrt{2}$-quasi unit disk graph,—although no network planarization algorithm is known, the problem can be circumvented to some extent by using "virtual links" [2]. However, the virtual links may need to be realized by long paths in the network, which makes the approach not so useful for many

applications. For more general networks, no efficient planarization method is known. Practical sensor networks often deviate significantly from the unit-disk graph model. The transmission ranges of sensors usually vary substantially in different directions due to reasons including multi-path fading, antenna design, etc., and it is common to observe a variation ratio up to five or more [11]. Also, it is often hard for sensors to obtain accurate location measurements via expensive localization devices (e.g., GPS) or localization algorithms [3]. No efficient network planarization algorithm is currently available for such practical wireless sensor networks.

In this paper, we present a novel method that leads to a new planarization algorithm that robustly planarizes sensor networks of realistic models: networks with non-uniform transmission ranges and unlocalized sensors (that is, sensors whose location information is unknown). The method starts with a simple shortest path between two faraway nodes in the network, and progressively planarizes the whole network.

The key for our approach is to solve the so called BIPARTITE PLANARIZATION problem. It has been proved to be NP-hard [8]. We present two planarization algorithms for different settings. We first present a $(2+\epsilon)$-approximation algorithm for this problem where ϵ is an arbitrarily small positive number. The algorithm is applicable to general networks, and achieves the best known approximation ratio. It outperforms the known results in the graph drawing research field [7].

We then present a fixed parameter tractable (FPT) algorithm that solves the problem exactly (namely, it finds the optimal solution) running in time $O((2+\epsilon)^k n^{O(1)})$. The algorithm uses the key observation that when a certain parameter is small, the problem can be solved in polynomial time. We show the usefulness of the algorithm to practical networks by simulations.

Since no information on node locations is known, the planar subgraph output by our method is not embedded. It is already sufficient for some applications, such as topology discovery (boundary recognition) [9]. If embedding is needed, the planar graph can be embedded efficiently as a plane graph by using existing planar embedding algorithms in graph drawing [5, 17] or in [9, 10]. The embedded graph can then be used for many applications, including geographic routing [12]. We demonstrate the performance of our planarization method and its application to topology discovery by extensive simulations. We show that the planar subgraphs maintain the distance between nodes with small stretches, detect holes and boundaries with a much higher precision than existing methods, and our method is robust to the network models with different configurations.

The rest of the paper is organized as follows. In Sect. 2, we present an overview of the new planarization scheme. In Sect. 3, we present an efficient approximation algorithm for planarization. In Sect. 4, we present a fixed parameter tractable algorithm for optimal planarization. In Sect. 5, we describe the implementation of the planarization scheme, and demonstrate its performance by simulations. In Sect. 6, we present the conclusions.

2 Overview of the Planarization Scheme

In this section, we present an overview of the new planarization scheme. The scheme is for planarizing a sensor network deployed in a two-dimensional sensor field. The

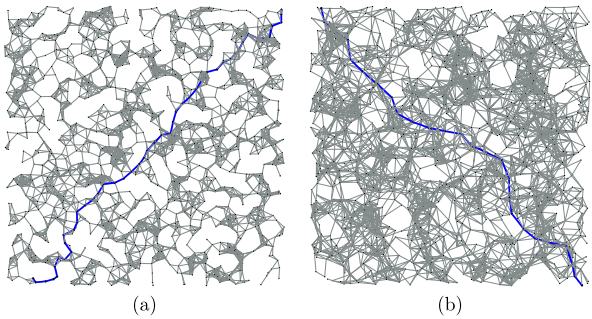

Fig. 1 The shortest path between two faraway nodes, in two sensor networks with drastically different transmission ranges. The average degree is 7 in both cases. (**a**) A sensornet with uniform transmission ranges. (**b**) A sensornet where the transmission ranges in different directions vary by up to 10 times

network is not required to be a unit-disk graph, nor is any location information needed. The only known information is the bi-directional connectivity between sensors. The planarization scheme consists of five steps, described as follows.

2.1 Finding a Shortest Path Between Two Faraway Nodes

The first step is to find a shortest path between two faraway nodes. The two faraway nodes can be found with the following common approach: first, randomly choose a node a, use one flooding to build a shortest path tree rooted at a, and find the node b that is the furthest (in hops) from a in the network; then, use a similar method to find the node c that is the furthest (in hops) from b. b and c are the two faraway nodes we need, and the unique path between b and c in the shortest path tree rooted at b is the shortest path between b and c. The advantage of a shortest path between two faraway nodes is that such a path usually does not twist, regardless of the uniformity of the transmission ranges. Thus, such a path is very likely to be a plane graph. We illustrate the property in Fig. 1. The two networks shown in Fig. 1 have drastically different features in the uniformity of transmission ranges, and the path is similar to a straight line in both cases. (When there are holes in the sensor field, the path may not be straight but still spreads out well.) This observation is validated by extensive simulations.

In the following, we will call the shortest path between b and c the *base path*.

2.2 Building a Shortest Path Tree

The second step is to build a shortest path tree. First, we find the node r_1 that is the furthest away (in hops) from the base path. This can be easily done by viewing the nodes in the base path as a super node, and build a shortest path tree rooted at this super node. Then, we build a shortest path tree rooted at node r_1. See Fig. 2(a) for an illustration.

2.3 Planarizing the Network Layer by Layer

In this third step, we planarize part of the network layer by layer. The nodes in the base path are in Layer 1. Recursively, in the shortest path tree rooted at r_1, if a node is the parent of a node in Layer i and is not included in any of the first i layers, then it

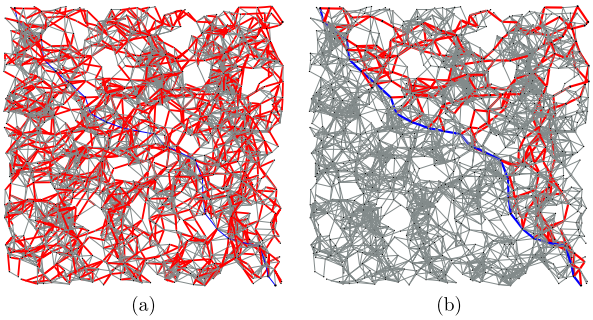

Fig. 2 *Red* (*Darker*) edges indicate (**a**) the shortest path tree rooted at node r_1. Here r_1 is near the up right corner of the network. (**b**) The nodes in the layers, and the edges between layers

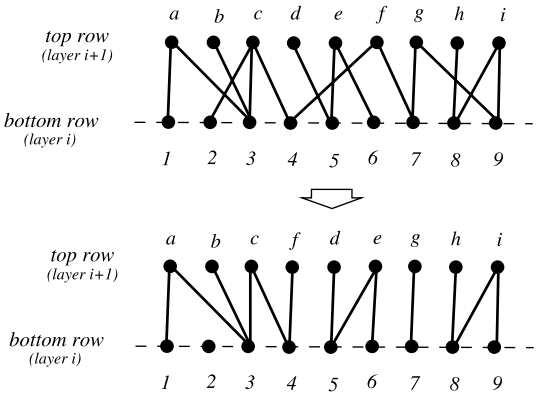

Fig. 3 Planarize two adjacent layers. The *upper graph* is before planarization, and the *lower graph* is after planarization. (1) In the *upper graph*, e, f are *cover nodes* of the virtual edge $\{5, 6\}$ and make its *cover number* be 2. (2) If c is the only cover node for virtual edge $\{3, 4\}$ in an optimal solution, neither of the walls $\{c, 3\}$ and $\{c, 4\}$ is removed by that solution

is in Layer $i + 1$. A node that is not the ancestor (in the tree) of any node in the base path is not included in any layer. Let us say that the maximum layer found this way is Layer M. See Fig. 2(b) for an illustration.

We progressively build a planar graph that includes the nodes in the M layers. First, we process Layer 1 and Layer 2. Let $G = (V_1 \cup V_2, E)$ denote the bipartite graph where the nodes in Layer 1 are in one row and the nodes in Layer 2 are in the other row. The edges in G are all those network links between Layer 1 and Layer 2. See Fig. 3 for an illustration. In the bipartite graph G, the nodes in the bottom row—which are the nodes in Layer 1—are placed following their order in the base path. The nodes in the top row—which are the nodes in Layer 2—are not ordered yet. We then use a planarization algorithm to remove some edges from G, and order the nodes in the top row, so that the remaining edges do not cross each other. (All the edges are straight.) Thus we obtain a planar subgraph between Layer 1 and Layer 2. Then, we process Layer 2 and Layer 3 in the same way, then Layer 3 and Layer 4, ..., and finally Layer $M - 1$ and Layer M. Note that when we are processing Layer i and $i + 1$, the nodes in Layer i (the bottom row) have been ordered. So the same algorithm can be used $M - 1$ times. All the edges we keep in these $M - 1$ steps form a planar graph.

The general idea is that the base path and the shortest path tree rooted at r_1 both act as good references for planarization. By processing the nodes layer by layer, we *comb through* the network and obtain a planar subgraph. The algorithm for planarizing the

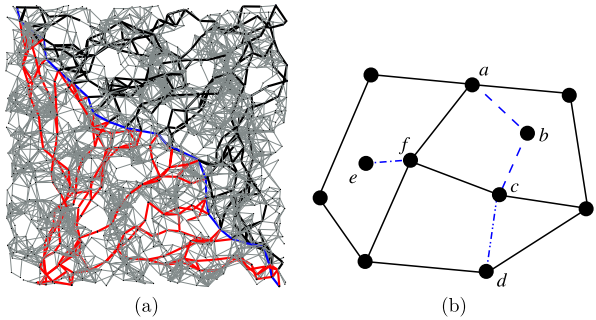

Fig. 4 (**a**) After finishing the first tree, we build the second and planarize it similarly. Edges of the second tree are *red* (*darker*); (**b**) the path $a \to b \to c$, the edge $\{c, d\}$, and the edge $\{e, f\}$ can be added into the planar graph

edges between two adjacent layers is the *key operation* in our method. We present the details of the algorithm in the following sections.

2.4 Building a Second Shortest Path Tree and Planarizing the Network

The planar graph built so far is a skeleton of the network covering part (often about half) of the sensor field. In this step, we build a second shortest path tree and planarize more of the network. This second tree is rooted at the node r_2 that is the furthest (in hops) from the node r_1. The planarization process is exactly the same as the process in the previous step (namely, the third step), except that here node r_2 replaces node r_1, and we do not include in the layers here those nodes that have been included in Layer 2 through Layer $M - 1$ in the previous step. See Fig. 4(a) for an illustration. The planar graphs built in this step and the previous step together form a large planar subgraph that covers most of the sensor field.

2.5 Refining the Planar Graph

The planar graph built so far is a skeleton of the network, which usually covers the whole sensor field. Those nodes outside it are usually within a few hops from it. To include all nodes into the planar graph and to include more edges, three simple steps are performed. First, if a node in the planar graph—which we shall call G_{planar}—finds that it has many 1- or 2-hop neighbors outside G_{planar}, it uses a 4-hop localized flooding to add one or more paths to G_{planar}, as long as the new path connects nodes in the same face and therefore preserves planarity of G_{planar}. Note that the previous planarization steps already tell us what the faces in the planar graph are, so this operation is easily doable. Second, if there are still nodes outside G_{planar}, they connect themselves to G_{planar} via small trees, which is a simple operation. The trees preserve the planarity of the graph. At this moment, the planar graph contains all the nodes. Then, to add more edges to G_{planar}, the nodes add their incident edges to it if the new edge connects two nodes in the same face (and therefore preserves the planarity). Now we get the final planar subgraph of the network. See Fig. 4(b) for an illustration on how the refining is done.

3 An Approximation Planarization Algorithm

In the previous section, the planarization scheme was presented. The *key* operation is how to planarize two adjacent layers. That is also the only part of the scheme whose detail has not been specified. Naturally, our objective is to remove as few edges as possible during planarization, because a dense planar graph is desirable. In this section, we present an approximation algorithm to this NP-hard problem.

We formally define the problem as follows:

> BIPARTITE PLANARIZATION problem (BiPP): In a bipartite graph $G = (V_1 \cup V_2, E)$, the nodes in the bottom row V_1 are already linearly ordered, and the nodes in the top row V_2 are not (see Fig. 3 for an example). How to remove some edges from G, and linearly order the nodes in the top row V_2, so that no two edges cross each other? The objective is to minimize the number of removed edges.

In the graph drawing research field, this problem is also called the ONE SIDED TWO-LAYER PLANARIZATION problem. It is known to be NP-hard, even when the nodes in V_2 all have degrees at most two and the nodes in V_1 all have degrees at most one [8]. The best existing solution that runs in polynomial time is a 3-approximation algorithm [7]. In this section, we present a new algorithm whose approximation ratio can be arbitrarily close to 2, thus improving the best known result.

Let us define a few terms. For any two nodes v_1 and v_2 in the bottom row V_1, if they are adjacent (in the sense of the given ordering of the nodes in V_1), then we imagine there is a *virtual edge* between v_1 and v_2. We say that a node $u \in V_2$ *covers* a virtual edge e if u has neighbors in the graph G that are on both the left side and the right side of e in the bottom row. Such a node u is called a *cover node* of e. The number of nodes in V_2 covering e is called the *cover number* of e. All the edges incident to a cover node of e are called the *walls* of e. Note that every cover node of e is incident to at least two walls of e (see the upper figure in Fig. 3).

Our approximation algorithm is based on the following observations.

Observation 3.1 (Lemma 10 of [7]) In any solution to the BIPARTITE PLANARIZATION problem, the cover number of any virtual edge e is at most 1. Therefore, let y be the cover number of e in the graph G. Then any solution must remove at least $y - 1$ walls of e.

Therefore, when the cover number of a virtual edge is large, if we remove two walls around e for each of the cover nodes, we would have removed no more than twice the number of edges removed by any solution plus one. This technique is used in our algorithm.

On the other hand, if the cover numbers of all the virtual edges are relatively small, we can solve the problem efficiently using a divide and conquer technique. The following observation is the basis for the divide and conquer technique.

Observation 3.2 Let y be the cover number of a virtual edge e in the graph G, in any solution at most one of e's cover nodes can keep all its corresponding walls around e.

For all the other cover nodes of e, each of them must remove all the walls on at least one side of e. Therefore, if we enumerate all the possible ways to solve the conflicts at e in any solution, there are at most $y2^{y-1} + 2^y$ cases to consider. (Specifically, for each cover node, we first consider if its walls should be removed; if yes, we consider which side of the walls to remove. So there are $y2^{y-1} + 2^y$ cases.)

If a virtual edge e has no cover node in a solution, the nodes in the bottom row can be separated into two parts: the left of e and the right of e. Then every node in the top row is adjacent to nodes in only one of the two parts. So in that case, the problem can be solved for the two subgraphs separately. If in an optimal solution, e has one cover node, then the following observation tells us that we can still divide the problem into two parts.

Observation 3.3 Suppose that in an optimal solution, a virtual edge e has exactly one cover node u. If the wall w_l (respectively, w_r) incident to u is the closest edge to e from the left (respectively, right) side, then w_l, w_r must have not been removed by that optimal solution (as shown in Fig. 3). In that case (assuming that we have guessed this case to be true), when we search for the optimal solution, we can mark the two walls closest to e from each side (namely, w_l and w_r) to be *irremovable* (namely, we do not remove them in the algorithm).

The approximation algorithm is shown in Fig. 5.

Let us first look at the subroutine *Exact-BiPP*. Within each recursion we pick the edge e and branch on all possible cases to reduce the cover number of e to at most 1. After that there are two cases to consider: (1) e has 1 cover node; (2) e has no cover node. In the first case, suppose v covers e with the edges $\{v, u_1\}$ and $\{v, u_2\}$, i.e., $\{v, u_1\}$ and $\{v, u_2\}$ are both kept in the final solution and marked irremovable. Hence we have divided the problem to two isolated subproblems in line 6 of *Exact-BiPP*. In the case e has no cover node in the solution, it is clear that we have split the problem into two independent subproblems. Since we enumerate all possible cases at e in the recursive stage, the optimal solution is guaranteed.

Let $c \leq |V_2|$ be the maximum cover number of all virtual edges in V_1. Let $T(n)$ be the running time of the algorithm where $n = |V_1|$ for each recursion. We have the recurrence relationship $T(n) < 2 \times c2^c T(\frac{n}{2})$. This recursion has a solution $T(n) < (2c2^c)^{\log_2 n} = n^{c+\log_2 c+O(1)}$. Hence we obtain the following lemma immediately.

Lemma 3.4 *The algorithm Exact-BiPP finds the optimal solution for the* BIPARTITE PLANARIZATION *problem and runs in time* $n^{c+\log_2 c+O(1)}$ *where* c *is the maximum cover number for all virtual edges in* V_1 *and* n *is the number of nodes in* V_1.

In the following theorem, we prove the algorithm *APX-BiPP*'s approximation ratio, $2 + 3\epsilon$, and its polynomial time complexity. The parameter ϵ can be a arbitrarily small positive number, so we assume that $\epsilon \leq 1/3$.

Theorem 3.5 *The planarization algorithm APX-BiPP is a* $(2 + 3\epsilon)$*-approximation for the* BIPARTITE PLANARIZATION *problem. It runs in time* $O(|V_1|^{\frac{1}{\epsilon}+\log_2 \frac{1}{\epsilon}+O(1)} + |E||V_1|)$.

Algorithm 1. *APX-BiPP*
Require: $G = (V_1 \cup V_2, E), \epsilon$
Ensure: G_{planar}: A solution to BiPP

1. **repeat**
2. $e \leftarrow$ a virtual edge in V_1 with cover number $> 1/\epsilon$
3. for each cover node of e, remove a wall incident to that node from each side of e
4. **until** no virtual edge in V_1 has cover number larger than $1/\epsilon$
5. call the procedure *Exact-BiPP*(G) and return the result

Algorithm 2. *Exact-BiPP*
Require: $G = (V_1 \cup V_2, E)$
Ensure: G_{planar}: A solution to BiPP

1. Let n be the number of nodes in V_1. If $n < 5$, solve the problem in the brute force way, and return the solution
2. $e \leftarrow$ the $\lfloor n/2 \rfloor$th virtual edge in V_1
3. $y \leftarrow$ the cover number of e
4. **for all** $y2^{y-1} + 2^y$ cases at e (see Observation 3.2) **do**
5. skip this iteration if in this case, an irremovable edge is to be removed
6. the problem is now split into two disjoint subproblems of roughly the same size,
   recursively call *Exact-BiPP* on them
7. record the solution of this case if it removes the least number of edges among all cases considered so far
8. **end for**
9. remove edges according to the best solution got above
10. **return** G

Fig. 5 The approximation algorithm

Proof Since *APX-BiPP* returns a result returned by *Exact-BiPP*, by Lemma 3.4, it produces a solution for the problem.

Let us look at the approximation ratio. In the recursive stage (subroutine *Exact-BiPP*) of the algorithm, by Lemma 3.4 we get the exact solution.

While in the preprocessing stage (lines 1 through 4 in *APX-BiPP*), let $a_e > 1/\epsilon$ be the cover number of e. Among the a_e nodes in V_2 covering e, at most one of them can be the cover node of e in any solution. We removed $2a_e$ edges over e. That is at most $a_e + 1$ more than the number of edges removed by an optimal solution S.

Suppose that in the preprocessing stage (lines 1 through 4 in *APX-BiPP*), we have run the loop m times. Let M be the total number of edges removed, since in each loop we remove at least $2/\epsilon$ edges, we have $M > 2m/\epsilon$. Let $2a_i$ be the number of edges we removed at the i'th loop for an edge $e_i (1 \leq i \leq m)$. Then by the argument in the previous paragraph, any optimal solution would have removed at least $a_i - 1$ edges for the edge e_i. Hence the total number of edges removed by any optimal solution S

would be at least

$$\sum_{i=1}^{m}(a_i - 1) = M/2 - m.$$

Let the residual graph (we obtained at line 5) be B'. Therefore the number of edges removed by S, denoted by $R(B)$, is at least $M/2 - m + R(B'')$ where B'' is the residual graph after we removed these $M/2 - m$ edges according to S. Since in the approximation stage we have removed all walls at those virtual edges, B' is a subgraph of B''. Hence we have $R(B') \leq R(B'')$ and $R(B) \geq M/2 - m + R(B')$. While in our approximated solution the number of edges removed, denoted by $A(B)$, is exactly $M + R(B')$. Hence we have

$$\begin{aligned}\frac{A(B)}{R(B)} &\leq \frac{M + R(B')}{M/2 - m + R(B')} = 2 + \frac{4m - 2R(B')}{M - 2m + 2R(B')} \\ &\leq 2 + \frac{4m}{2m/\epsilon - 2m}. \end{aligned} \tag{1}$$

Now we show that the algorithm runs in polynomial time. The preprocessing stage takes time $O(|E||V_1|)$. In the stage when we call *Exact-BiPP*, since the cover numbers of all virtual edges in V_1 are bounded by $\frac{1}{\epsilon}$, by Lemma 3.4 the running time is upper bounded by $n^{\frac{1}{\epsilon}+\log_2 \frac{1}{\epsilon}+O(1)}$. Therefore the total running time is then $O(|V_1|^{\frac{1}{\epsilon}+\log_2 \frac{1}{\epsilon}+O(1)} + |E||V_1|)$. □

4 A Parameterized Algorithm for Optimal Planarization

In our extensive simulations, we observed that while planarizing the subgraph induced by two adjacent layers, the number of edges that need to be removed is usually much smaller than the number of nodes in those two layers, for a network of moderate density (average degree roughly between 6 and 12). If the network is very dense, we can use a simple preprocessing operation to reduce the edge density to the moderate level. (Details of the operation will be discussed later.) Therefore a question remains: can we use that observation to practically improve the planarization algorithm?

In recent years, a new approach called parameterized computation has been proposed to solve NP-hard problems by exploiting small parameters [6]. Let k be a parameter in a parameterized problem. We say that the problem is *fixed parameter tractable* (FPT) if it can be solved optimally in time $O(f(k)n^{O(1)})$, where n is the input size and $f(k)$ is a function of k. When k is bounded, not only is the time complexity polynomial, but it usually also grows much slower than $O(n^k)$ when n increases. Quite a few NP-hard problems have been proved to be FPT with effective algorithms. For example, the VERTEX COVER problem with parameter k as the cover size can be solved in time $O(1.286^k + n^3)$ [4].

In [7], the authors showed that the BIPARTITE PLANARIZATION problem (also called the ONE SIDED TWO-LAYER PLANARIZATION problem) is fixed parameter tractable when the parameter k is an upper bound for the number of edges to be

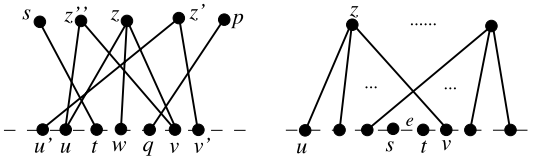

Fig. 6 (**a**) $z(u, v)$ is an arc; the edges $\{p, q\}, \{s, t\}, \{z'', u\}$ conflict with $z(u, v)$; the conflict number of $z(u, v)$ is then 3; $z'(u', v')$ strictly covers $z(u, v)$ and has a larger conflict number 7. (**b**) Cover number and conflict number at virtual edge $e(s, t)$: let f be the cover number of the virtual edge $e(s, t)$, there are at least $f - 1$ edges incident to nodes between u and s but not z

removed. They developed an FPT algorithm for the problem running in time $3^k n^{O(1)}$. In this section, we present an improved FPT algorithm running in time $(2+\gamma)^k n^{O(1)}$, where γ can be an arbitrarily small positive number.

Formally, the parameterized version of the BIPARTITE PLANARIZATION problem can be stated as follows:

> Given an instance of the BIPARTITE PLANARIZATION problem and a parameter k, either find a solution to the instance that removes at most k edges, or report that no such solution exists.

Let us first define a few terms. Let $u \in V_1$ be the leftmost neighbor of a node $z \in V_2$, and $v \in V_1$ be the rightmost. We call the pair of edges $\{u, z\}, \{z, v\}$ an *arc*, and denote it by $z(u, v)$. We say that the arc $z(u, v)$ *conflicts* with an edge $\{w_1, w_2\}$ if it is necessary to remove $\{w_1, w_2\}$ in order to keep both edges $\{z, u\}, \{z, v\}$ in a solution. To help simplify the following discussion, for each pair of arcs $z(u, v), z'(u, v)$—i.e., they share the same leftmost and rightmost neighbors u, v in V_1—we also consider $\{z, u\}$ a conflict edge of $z'(u, v)$. Similarly $\{z', u\}$ is also a conflict edge of $z(u, v)$.

The *conflict number* of an arc $z(u, v)$ is the number of edges that conflict with the arc. (See Fig. 6(a) for an illustration of the terms.)

The conflict number of a given arc $z(u, v)$ is simply the summation of the following two numbers: (1) the number of edges incident to nodes (exclusively) between u, v in V_1 but not z; (2) the number of arcs of the form $z'(u, v)$ (where $z' \neq z$), i.e. they share the same leftmost (rightmost) node in V_1. The set of edges that are conflicting with a given arc can also be decided easily by definition.

The following lemma shows the relationship between cover numbers and conflict numbers.

Lemma 4.1 *If there is a virtual edge e in V_1 with cover number f, there must exist an arc that has conflict number at least $f - 1$. In other words, if no arc has conflict number larger than $f - 1$, no virtual edge in V_1 will have cover number larger than f.*

Proof Suppose the left-most walls of e are incident to u in V_1. Let $z(u, v)$ be the arc such that v in V_1 is the furthest away to the right of u. Hence $\{z, u\}$ is a leftmost wall of e and v is the right-most neighbor of the cover nodes adjacent to u. Each of the $f - 1$ cover nodes (other than z) of e either is adjacent to u and has no neighbor exclusively to the right of v, or has its left-most wall exclusively to the right of u.

Algorithm 3. *FPT-BiPP* (where $\beta \geq 1$ is any given number)
Require: $G(V_1 \cup V_2, E), k$
Ensure: G_{planar}: A solution removing no more than k edges

1. $z(u, v) \leftarrow$ the arc with the maximum conflict number $c(z)$
2. **if** $c(z) = 0$ **then return** true
3. **if** $k < c(z)$ **then return** false
4. **if** $c(z) \leq \beta$ **then**
5. planarize G using the procedure *Exact-BiPP*. (That is, run *Exact-BiPP*(G).)
6. **if** *Exact-BiPP*(G) removes at most k edges, **return** true; **else return** false
7. **end if**
8. $E(z) \leftarrow$ the set of edges conflicting with $z(u, v)$
9. **if** *FPT-BiPP*$(G(V_1 \cup V_2, E \setminus E(z)), k - c(z))$ **return** true
10. **if** *FPT-BiPP*$(G(V_1 \cup V_2, E \setminus \{\{z, u\}\}), k - 1)$ **return** true
11. **if** *FPT-BiPP*$(G(V_1 \cup V_2, E \setminus \{\{z, v\}\}), k - 1)$ **return** true
12. **return** false

Fig. 7 The FPT algorithm

By definition, each of these $f - 1$ cover nodes (other than z) of e will contribute at least 1 to $z(u, v)$'s conflict number. Either their rightmost walls (if they are adjacent to u but not v) or their leftmost walls will be conflict edges of $z(u, v)$. The conflict number of $z(u, v)$ is, therefore, at least $f - 1$. □

We present the FPT algorithm in Fig. 7. The constant β is any predefined positive number that is no smaller than 1. The algorithm returns a solution if there exists a solution that removes at most k edges, and returns *false* otherwise. (With a little abuse of notations, when the algorithm returns *true*, it returns the solution as well.) Note that when a solution is found, the conflict number of every arc becomes 0.

The following theorem proves the correctness and complexity of the algorithm.

Theorem 4.2 *The algorithm FPT-BiPP will either find a solution for the* BIPARTITE PLANARIZATION *problem by removing at most k edges, or correctly report that there is no such solution. The running time of the algorithm is upper bounded by* $(2 + \frac{1}{\beta})^k n^{\beta + \log_2 \beta + O(1)}$.

Proof During the calling of the routine *Exact-BiPP* in line 5 (that is, when $c(z) \leq \beta$), since the conflict numbers of the nodes in V_1 are upper bounded by β, by Lemma 4.1, the cover number of the virtual edges in V_1 will be bounded by $\beta + 1$. Again by Lemma 3.4, the running time of *Exact-BiPP* in this algorithm will be bounded by $T_1 = n^{\beta + \log_2 \beta + O(1)}$.

As for the recursive calls in lines 9, 10 and 11, we branch into three cases where k decreases by 1, 1 and $c(z)$, respectively. Since we enter this stage only if $c(z) > \beta$, if we use $T(k)$ to denote the running time of the algorithm, the following recurrence relationship holds for $T(k)$:

$$T(k) \leq 2T(k - 1) + T(k - \beta).$$

We use induction to show that $T_1(2+\frac{1}{\beta})^k$ is an upper bound for $T(k)$. In the case when $k < c(z)$, we return false right away. If $c(z) \le k \le \beta$, we already have $T(k) \le T_1$. Hence when $k \le \beta$, we always have $T(k) \le T_1(2+\frac{1}{\beta})^k$.

Suppose for $\beta \le k < t$ we have $T(k) \le T_1(2+\frac{1}{\beta})^k$. Using basic calculus, it is easy to verify that when $\beta \ge 1$, $2(2+\frac{1}{\beta})^{\beta-1}+1 \le (2+\frac{1}{\beta})^{\beta}$. Thus we have

$$T(t) \le 2T(t-1)+T(t-\beta) \le 2T_1\left(2+\frac{1}{\beta}\right)^{t-1}+T_1\left(2+\frac{1}{\beta}\right)^{t-\beta}$$
$$\le T_1\left(2+\frac{1}{\beta}\right)^{t-\beta}\left[2\left(2+\frac{1}{\beta}\right)^{\beta-1}+1\right] \le T_1\left(2+\frac{1}{\beta}\right)^{t}.$$

This proves that the running time of the algorithm is bounded by $(2+\frac{1}{\beta})^k n^{\beta+\log_2\beta+O(1)}$. □

The exponential part of the running time of the algorithm *FPT-BiPP* can be arbitrarily close to 2^k by choosing a large enough value for β.

The difference between the approximation approach and a parameterized approach is that the latter gives the optimal solution. Although the problem is NP-hard, the FPT algorithm solves the problem in polynomial time when the parameter is small, i.e., the exponential part of the running time is independent of the number of the nodes in the input graph. As we will show later in simulations, the number of edges to be removed is actually very small (less than 15 in most cases for networks of up to 2500 nodes) for networks with practical models.

5 Implementing the Planarization Algorithms

We conducted extensive simulations to test the performance of the planarization method. The performance has been very stable for different network models, sensor deployment methods, network sizes and sensor densities. In the following, we present some typical simulation results. A planarized network has numerous applications, including topology discovery, localization, geographic routing, etc. (For geographic routing, embedding is needed and such embedding algorithms are available [5, 9, 10, 17].) We illustrate the performance of our result by showing its application to topology discovery. We observe that our planarized network can locate holes and outer boundaries of the sensor fields accurately.

We will only present the simulation results of the centralized implementation of our algorithm. For interested readers, we outline the distributed implementation of our planarization algorithm in the following. The centralized *FPT-BiPP* algorithm will be used for solving the BIPARTITE PLANARIZATION problem during the planarization process. Our planarization method utilizes a few shortest path trees, and a practically limited amount of localized flooding to refine the final result. (The total cost of the localized flooding is about the same as flooding the network once or twice.) Both building shortest path trees and localized flooding are very mature techniques in networking. Both algorithms presented in Sects. 3 and 4 planarize the network

layer by layer, which naturally corresponds to a distributed implementation. While planarizing two adjacent layers, we take the simple approach of letting one node in the two layers act as a proxy and run the algorithm in a centralized way. (The nodes in the two layers can easily send their information to the proxy node via the corresponding shortest path tree.) Note that our algorithms take the divide and conquer approach, so it is simple to decentralize its implementation.

The faces in the planarized network are always very clear throughout the planarization process. That is because nodes are planarized layer by layer, and the planarization algorithm sorts the edges incident to each node by ordering the nodes in the two adjacent layers. That makes the refinement step and the topology discovery application of using faces to recognize holes/boundaries very simple.

Our method works well for networks of moderate or sparse edge densities. For dense networks (e.g., average degree 15 or more. We picked 15 as our threshold based on our experiences. Minor changes on the threshold may slightly affect the results of the algorithm), we apply the following simple preprocessing that can effectively reduce the edge density and the number of edge crossings: for maximal cliques in each node's neighborhood, we remove some edges from it so that the remaining edges form a star. Detecting maximal cliques in graphs itself is a very interesting topic, readers are referred to [1] for an in-depth discussion on this topic. In this paper, we suggest the following simple greedy approach to detect maximal cliques: starting from a node u and the induced subgraph C of G containing u. We repeatedly find a neighbor v of u such that $C \cup \{v\}$ is a clique and add v to C. We stop when there is no such v existing. At the end C is one of the maximal cliques in u's neighborhood. Apparently this process can be done in polynomial time. Note that the average degree of a planar graph is always less than 6. So such an edge-reduction preprocessing step goes along well with planarization.

We have presented two planarization algorithms: an approximation algorithm and an FPT algorithm. They have very similar performance in practice. We present the simulation results for the FPT algorithm in the following (the results for the approximation algorithm are very similar to that of the FPT algorithm so we decided not to include them here). In most simulations, the optimal planarization solution removes less than 15 edges for all the layers. So by setting the parameter k to the *FPT-BiPP* to 15 or a little above for networks of N (ranged from 1000 to 2500) nodes, the FPT algorithm finds the optimal solution and maintains a low time complexity at the same time.

We randomly deploy N sensors uniformly in a 15000×15000 square area. The network follows the quasi unit disk graph (quasi-UDG) model: two nodes do not have a link if their distance is greater than R, have a link if their distance is less than r, and have a link with probability $1/3$ if their distance is between r and R. (Here $r \leq R$.) Let $\alpha = R/r$. When N and α are given, we adjust r and R to obtain the desired average node degree. To make the sensor field non-trivial, we also randomly place holes in the network. Our reported results are for two holes of radius about $2R$. For each parameter configuration, 1000 networks are generated and measured.

A typical planarization result is shown in Fig. 8. To quantitatively analyze the performance of planarization, we measure the stretch (*str.*) of hop distance and its standard deviation σ. Let u, v be two nodes, whose hop distance is $h(u, v)$ before planarization and is $h'(u, v)$ after planarization. The *multiplicative stretch* is defined as $h'(u, v)/h(u, v)$, and the *additive stretch* is $h'(u, v) - h(u, v)$. To better charac-

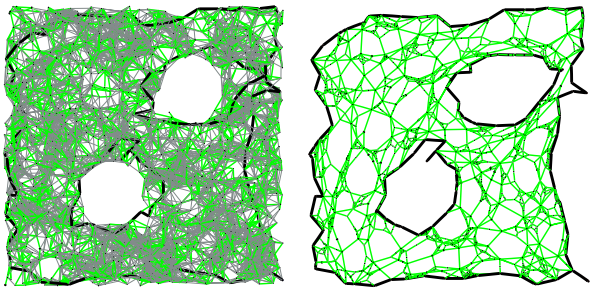

Fig. 8 Wireless sensor network and its planarization. (**a**) The *black* (*darkest*) edges and the *green* (*darker*) edges are the planarized network. The *light* (*grey*) edges are the remaining edges in the original network. The original network is a quasi-UDG with $N = 2000$ nodes and average degree 12, where transmission ranges vary by as much as $R/r = 5$. (**b**) A plane embedding of the planarized network

Table 1 Planarization results for $N = 1500$

Additive stretch for nodes within *Diameter*/4 hops									
	$d = 7$			$d = 9$			$d = 11$		
α	D	*str.*	σ	D	*str.*	σ	D	*str.*	σ
1	48.3	4.03	9.80	40.1	3.56	8.04	36.1	3.36	7.27
2	36.2	3.45	8.11	30.6	3.28	7.19	27.4	3.24	6.66
5	28.9	3.23	6.76	24.0	3.05	6.25	21.5	2.98	6.00
10	27.0	3.11	6.82	23.7	3.01	6.13	21.4	2.96	5.83
Multiplicative stretch for nodes more than *Diameter*/4 hops apart									
	$d = 7$			$d = 9$			$d = 11$		
α	d_p	*str.*	σ	d_p	*str.*	σ	d_p	*str.*	σ
1	3.5	1.27	0.57	3.8	1.28	0.56	3.8	1.30	0.57
2	3.7	1.31	0.54	3.8	1.35	0.56	3.9	1.38	0.58
5	3.7	1.36	0.55	3.8	1.40	0.55	3.8	1.43	0.56
10	3.7	1.36	0.54	3.8	1.40	0.55	3.8	1.43	0.57

terize the stretch of different node pairs, we measure the multiplicative stretch for nodes whose hop distance is greater than $1/4$ of the network diameter, and measure the additive stretch for the others. The results are shown in Table 1 where d is the average degree of the input networks, d_p is the average degree of the planar spanning subgraph found and D is the average diameter of the input networks.

We see that for networks of very different connectivity models and densities, the stretch is constantly small. This shows the good performance of the planarized networks.

6 Conclusions

In this paper, we present a new planarization scheme for wireless sensor networks. The scheme is efficient and works for sensor networks of a general model, where

sensors can have non-uniform transmission ranges and no location information is needed. The planarization scheme provides a tool for many applications. A natural question is how to further decentralize the scheme for higher efficiency, and to extend the scheme for sensor networks of more specialized topological features. This remains as our future research.

References

1. Abello, J., Resende, M., Sudarsky, S.: Massive quasi_clique detection. In: Rajabaum, S. (ed.) Lecture Notes in Computer Science, vol. 2286, pp. 598–612 (2002)
2. Barrière, L., Fraigniaud, P., Narayanan, L.: Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In: Proc. 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, DIALM 2001, pp. 19–27 (2001)
3. Bruck, J., Gao, J., Jiang, A.: Localization and routing in sensor networks by local angle information. In: Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc, pp. 181–192 (2005)
4. Chen, J., Kanj, I., Jia, W.: Vertex cover: further observations and further improvements. J. Algorithms **41**, 280–301 (2001)
5. Davidson, R., Harel, D.: Drawing graphs nicely using simulated annealing. ACM Trans. Graph. **15**, 301–331 (1996)
6. Downey, R., Fellows, M.: Parameterized Complexity. Springer, New York (1999)
7. Dujmovic, V., Fellows, M., Hallett, M., Kitching, M., Liotta, G., McCartin, C., Nishimura, N., Ragde, P., Rosamond, F., Suderman, M.: A fixed-parameter approach to two-layer planarization. In: Proc. 9th International Symposium on Graph Drawing, GD 2001. Lecture Notes in Computer Science, vol. 2265, pp. 1–15 (2002)
8. Eades, P., Whitesides, S.: Drawing graphs in two layers. Theor. Comput. Sci. **131**, 361–374 (1994)
9. Funke, S., Klein, C.: Hole detection or: "How much geometry hides in connectivity?". In: Proc. 22nd ACM Symposium on Computational Geometry, SoCG, pp. 377–385 (2006)
10. Funke, S., Milosavljevic, N.: Network sketching or: "How much geometry hides in connectivity?—Part II". In: Proc. 18th ACM-SIAM Symposium on Discrete Algorithms, SODA, pp. 958–967 (2007)
11. Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., Wicker, S.: Complex behavior at scale: an experimental study of low-power wireless sensor networks. Technical Report UCLA/CSD-TR 02-0013, UCLA (2002)
12. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: Proc. 6th Annual International Conference on Mobile Computing and Networking, MobiCom, pp. 243–254 (2000)
13. Kim, Y., Govindan, R., Karp, B., Shenker, S.: Geographic routing made practical. In: Proc. 2nd Symposium on Networked System Design and Implementation, NSDI, pp. 217–230 (2005)
14. Kranakis, E., Singh, H., Urrutia, J.: Compass routing on geometric networks. In: Proc. 11th Canadian Conference on Computational Geometry, pp. 51–54 (1999)
15. Kuhn, F., Wattenhofer, R., Zollinger, A.: Worst-case optimal and average-case efficient geometric ad-hoc routing. In: Proc. 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc, pp. 267–278 (2003)
16. Moore, D., Leonard, J., Rus, D., Teller, S.: Robust distributed network localization with noisy range measurements. In: Proc. 2nd International Conference on Embedded Networked Sensor Systems, SenSys, pp. 50–61 (2004)
17. Nakano, S.: Planar drawings of plane graphs. IEICE Trans. Inf. Syst. **3**, 384–391 (2000)
18. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., Shenker, S.: GHT: a geographic hash table for data-centric storage. In: Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA, pp. 78–87 (2002)
19. Wang, Y., Gao, J., Mitchell, J.S.: Boundary recognition in sensor networks by topological methods. In: Proc. 12th Annual International Conference on Mobile Computing and Networking, MobiCom, pp. 122–133 (2006)