# In Defense of Stint for Dense Breach-Free Sensor Barriers

Jorge A. Cobb

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080-3021
U.S.A.
Email: cobb@utdallas.edu

***Abstract*—A sensor barrier consists of a subset of sensors that divide an area of interest into two regions so that no intruder can move from one region to another without being detected by at least one sensor. The length of time over which the sensors protect the area can be maximized if the sensors are divided into disjoint barriers, and only one barrier is active at a time. Dividing the sensors into a maximum number of disjoint barriers can be done with the well-known Stint algorithm. However, recently, a new security problem was discovered, known as a barrier-breach, that allows an intruder to cross the area undetected by taking advantage of the time when one barrier is replaced by the next. This is dependent not on the structure of an individual sensor barrier, but on the relative shape of two consecutive sensor barriers. There have been several heuristics proposed in the literature that attempt to maximize the number of breach-free barriers. In recent work, we proposed a heuristic that outperforms earlier heuristics. In this paper, we refine our previous heuristic to deliver even better performance than before by the careful elimination of redundant nodes. In addition, me present a simple modification of the Stint algorithm that results in a method that outperforms all others when the density of sensors nodes is very high.**

***Keywords–Sensor networks; Barrier coverage; Security breaches.***

## I. INTRODUCTION

A wireless sensor network (WSN) consists of an area of interest in which a large number of sensor nodes have been deployed. We assume each sensor has a limited battery lifetime, and thus it is crucial for the network to use the limited sensor lifetime wisely. Information that the sensors collect could be relayed to a wireless base station [1].

In general, there are two types of coverage provided by a sensor network: full coverage and partial coverage. In full coverage, the entire area is covered at all times by the sensors. Therefore, any event occurring anywhere in the area is detected immediately [2] [3] [4] [5]. In partial coverage, only certain regions at a time are covered by the sensors. Thus, any event occurring outside of the current region being monitored is not detected. [6] [7] [8].

A sensor barrier consists of a subset of sensors that divide an area of interest into two regions so that no intruder can move from one region to another without begin detected by at least one sensor. Barrier coverage is thus a special case of partial-coverage. There have been extensive studies of sensor barriers due to their many applications, in particular intrusion detection [9] [10] [11] [12] [13] [14] [15] [16]. Figure 1(a) highlights a subset of sensors that provide barrier coverage to the area.

Maintaining full coverage is often counter-productive for applications such as intrusion detection, because the protection only lasts for the duration of a single lifetime of the sensors. However, if $n$ disjoint barriers are constructed, then only one barrier needs to be active at a time. When the currently active barrier is running out of power, the next barrier is activated. In this manner, the protection lasts $n$ times the lifetime of a sensor. Figure 1(b) shows the sensors divided into four barriers.

The problem of dividing the sensors into the maximum number of disjoint barriers has been solved in polynomial time in [11]. Here, two optimal algorithms, Stint and Prahari, are presented. Stint considers the case when the remaining battery level of each sensor is equal, while Prahari addresses the harder case in which each sensor may have different remaining battery level. Their approach is based on transforming the sensor connectivity graph into a maximum flow problem.

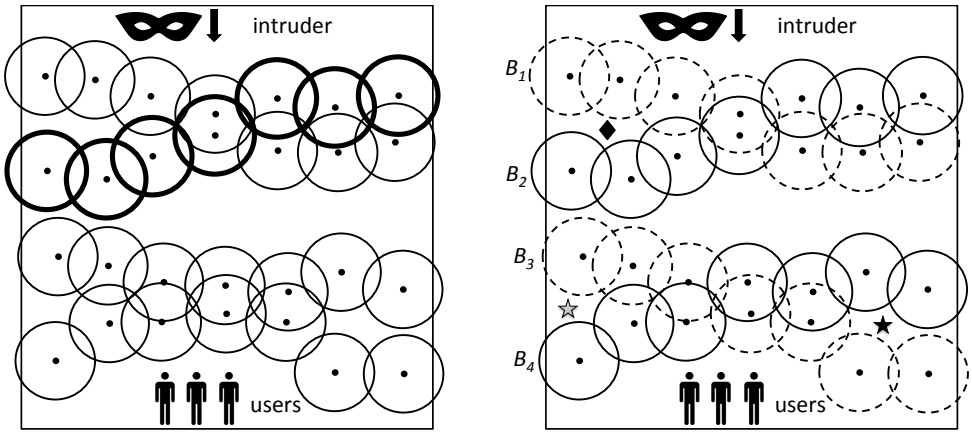
In recent years, [17] [18], a vulnerability of sensor barriers, known as a *barrier breach*, was discovered. For some barriers, it is possible for an intruder to cross the area of interest after activating one barrier and deactivating the previous one. This is true even though each of these two sensor barriers correctly divides the area into two disjoint regions, and thus provides appropriate coverage.

Although methods have been devised for dividing a group of sensors into breach-free barriers [17] [18], these are heuristics and are not guaranteed to be optimal. To our knowledge, the computational complexity of determining the largest number of breach-free sensor barriers remains an open problem.

In earlier work [19], we presented a heuristic which outperforms those in [17] [18]. In this paper, we refine our previous heuristic to deliver even better performance than before by the careful elimination of redundant nodes. In addition, me present a simple modification of the Stint algorithm that results in a method that outperforms all others when the density of sensors nodes is very high.

The rest of this paper is organized as follows. Section II reviews earlier work, such as the definition of a barrier breach and the heuristics from [17] [18] [19]. In Section III, we present our improved heuristic. In Section IV, we present a modification of Stint whose output is guaranteed to be breach-free. Finally, simulation results and concluding remarks are presented in Sections V and VI respectively.

 ISBN: 978-1-61208-499-2




(a) Sensor Barrier (b) Barrier Breaches

Figure 1. Sensor Barriers

## II. BACKGROUND

In this section, we overview the concept of a barrier breach. In addition, we discuss Stint [11], and how its solution is vulnerable to barrier breaches. We then overview the heuristics presented in [17] [18] [19].

### A. Barrier Breaches

We first illustrate the concept of a barrier breach with the example of Figure 1(b). The figure shows four different sensor barriers, with each barrier displayed with different line types.

Assume the intruder penetrates the area in the direction of the users. Assume also that all sensor nodes have the same lifetime, which we normalize to one unit. If all sensors are operational at the same time, then we form a single barrier that protects the users for a duration of one unit.

Instead, we can divide the sensors in four barriers, $B_1$ through $B_4$. If we use the barriers in a sequential wakeup-sleep cycle ($B_1$, $B_2$, $B_3$, and finally $B_4$), the users are protected for a total of four time units.

While this strategy is appealing, it suffers from the following drawback.

(a) The order in which $B_1$ and $B_2$ are scheduled affects the effectiveness of the barriers. Consider scheduling $B_2$, followed by $B_1$. In this case, an intruder could move to the point highlighted by a diamond, and after $B_2$ is turned off, the intruder is free to cross the entire area.
(b) Only one of $B_3$ and $B_4$ is of use. To see this, suppose that we activate $B_3$ first. In this case, the intruder can move to the location marked by the black star. Then, when $B_4$ is activated and $B_3$ deactivated, the intruder can reach the users undetected. The situation is similar if $B_4$ is activated first, and the intruder moves to the location of the grey star.

The above drawback was originally presented in [17] and given the name *barrier breach*. A barrier breach for an ordered pair $(B_1, B_2)$ of barriers is a point $p$ such that $p$ is outside the sensing range of $B_1$ and $B_2$, and furthermore, $B_1$ cannot detect an intruder moving from the top of the area towards $p$, and $B_2$ cannot detect the intruder moving from $p$ towards the bottom of the area.

Note that this definition can be easily extended to a sequence of barriers, and thus, to the schedule used to activate the barriers.

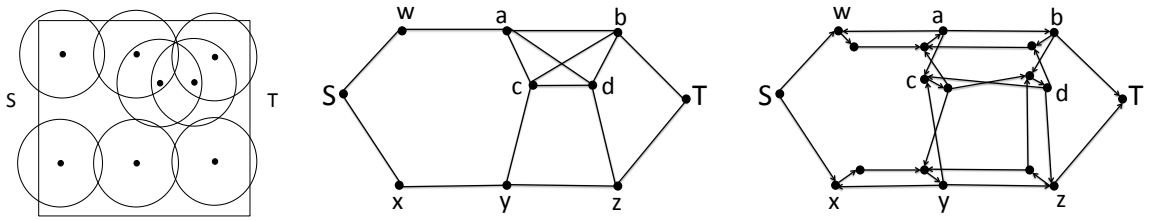
### B. Upper Bound on Breach-Free Barriers

Under the assumption that all sensor nodes have equal lifetime, finding the largest number of disjoint sensor barriers has been solved in polynomial time by Kumar et. al. [11] with their algorithm known as Stint. This provides obviously an upper bound on the maximum number of breach-free barriers, because Stint does not take barrier breaches into account. Given that the complexity of obtaining the largest number of breach-free barriers remains an open problem, several heuristics have been presented in the literature [17] [18] [19]. Some of these heuristics are based on Stint, so we briefly overview Stint below.

Consider Figure 2(a), with a sample sensor network. The first step consists of adding a virtual source node $S$ and a virtual destination node $T$. Then, a connectivity graph is built containing all the sensor nodes as vertices in the graph, including the virtual nodes $S$ and $T$. Two nodes are then connected via an edge iff their sensing ranges overlap. Also, an edge is added between $S$ and any sensor whose range overlaps the left border of the area. Similarly, an edge is added between $T$ and any sensor whose range overlaps the right border of the area. The resulting graph is shown in Figure 2(b). Finally, the maximum number of node-disjoint paths are found between $S$ and $T$.

Computing the maximum number of node-disjoint paths is done with a small variation of the above graph and applying a maximum-flow algorithm. Each sensor node $x$ in the graph is replaced by two nodes, $x_{in}$ and $x_{out}$, with a directed edge from $x_{in}$ to $x_{out}$. For every sensor node $y$ that is a neighbor of $x$, the directed edges $(x_{out}, y_{in})$ and $(y_{out}, x_{in})$ are added to the network. Finally, an edge $(S, x_{in})$ is added for every neighbor $x$ of $S$, and also an edge $(y_{out}, T)$ for every neighbor $y$ of $T$. All of these directed edges have a capacity one. The

 ISBN: 978-1-61208-499-2 




(a) Sensor Area (b) Connectivity Graph (c) Enhanced Connectivity Graph

Figure 2. Stint Sensor Barriers

resulting enhanced connectivity graph is shown in Figure 2(c). The node-disjoint paths are obtained by running a maximum-flow algorithm on this enhanced connectivity graph.

### C. Barrier Crossings

Several heuristics for maximizing the number of non-penetrable barriers have been proposed in [17] [18]. These heuristics are based on variations of Stint that restrict the connectivity graph of Figure 2(b) in a manner that ensures the output is breach-free. In [18], it was shown via simulation that one of these heuristics outperforms the others. Next, we briefly describe this heuristic, which will be used for comparison against our heuristic that we present in Section III.

The heuristic begins by constructing the connectivity graph of Stint, as in Figure 2(b), and removing edges that cross each other. For example, in Figure 2(b), edges $(c, b)$ and $(a, d)$ cross each other. One of these edges must be removed, leaving in place a graph where no edges cross. Then, as in Stint, the maximum number of node-disjoint paths are found, resulting in the desired sensor barriers.

What remains to be decided is, of the set of edges that cross each other, which ones should be removed? Note that in the example only one of the edges needs to be removed, not both. The method proposed in [18] deletes the edge that has the least impact on the number of node-disjoint paths from $S$ to $T$.

That is, for every edge $(x, y)$ that crosses other edges, the edge is removed and Stint is run on the remaining graph. The edge whose removal produces the largest number of disjoint paths is chosen for permanent removal. The process is repeated until no two edges cross each other. A more detailed description may be found in [18].

Note that this is a computationally intensive algorithm in the case of a dense sensor network where many edges cross each other. This is because if there are $m$ number of edges that cross other edges, then Stint has to be run $m$ times, which results in a graph with $m - 1$ crossing edges. This in turn requires Stint to be run $m - 1$ times, and so on.

### D. Ordered Ceilings

In [19], we presented a heuristic which outperforms those presented in [17] [18]. We briefly overview it here, as it is the foundation for our improved heuristic in Section III.

Consider Figure 3(a), and note that a sensor barrier $B$ divides the area of interest into an *upper region* and a *lower* region. The *ceiling* of $B$ consists of all points $p$ along the border of the sensing radius of each sensor in $B$ such that one can travel from $p$ to any point in the upper region without crossing the sensing area of any sensor. The *floor* of $B$ is defined similarly but with respect to the lower region. Consider again the sensor barrier depicted in Figure 3(a). Its ceiling and floor are depicted in Figure 3(b).

Our heuristic finds each barrier iteratively as follows. Consider the set of all sensor nodes as a single barrier, and obtain its ceiling. The first barrier consists of all sensor nodes that take part of this ceiling. These nodes are then removed from the network. The remaining sensor nodes are again considered as a single barrier, and their ceiling is found, which in turn yields the second barrier, etc..

For example, consider Figure 3(c). The first barrier, $B_1$, is obtained from the ceiling of all the sensor nodes. The next barrier, $B_2$, is obtained by removing $B_1$ and obtaining the ceiling of the remaining nodes. $B_3$ is found the same way.

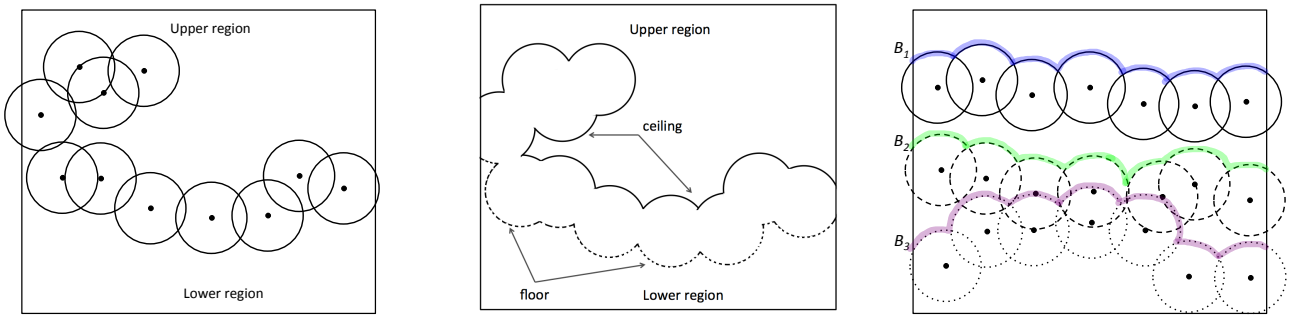
To obtain the ceiling of a set of nodes, one simply begins with the top-most node that intersects the left border of the area. Let $b_0$ be this node, and $p_0$ the top-most point where $b_0$ intersects the left border. Point $p_1$ is the *first point, clockwise from* $p_0$, where the sensing area of $b_0$ intersects the sensing area of another node $b_1$. This point becomes $p_1$. This process continues until either the right border is reached, and thus the barrier is complete, or it returns to the left border, in which case the nodes are discarded since they do not take part of any barrier.

We refer to our heuristic as the *ordered-ceilings* heuristic, and is presented in more detail in [19].

## III. Compressed Ceilings Heuristic

We next present our enhancement to the above heuristic, which we refer to as the *compressed ceilings* heuristic. Consider the barrier in Figure 4(a). Assume that it was obtained using the ordered ceilings heuristic. As the barrier is constructed from left to right, the algorithm runs into node $i$. The next clockwise node after $i$ is $j$. Similarly, $k$ follows $j$ clockwise. Note that the next clockwise node after $k$ is $i$ again. In this case, all three nodes $j$, $k$, and $l$, serve no practical purpose. We call these nodes a *detour* because they simply return to the original node. We can remove these nodes and still maintain a sensor barrier.

Consider now nodes $x$, $y$, and $z$. Again, the next clockwise node after $x$ is $y$, followed clockwise by $z$. Note, however, that




(a) Sensor Set (b) Ceiling and Floor (c) Barriers Obtained

Figure 3. Ceiling-First Method

$y$ is not needed for the integrity of the barrier. Nonetheless, our earlier heuristic will have included it in the barrier. Thus, $y$ can be removed. We refer to $y$ as a *spurious* node, because it connects two nodes which are already connected.

Spurious nodes and detours could be removed as the barrier is constructed, or doing a pass over the barrier once its construction finishes. We choose the latter for its simplicity. The nodes that we remove as described above are not deleted permanently from the network. Simply deleting them would serve no purpose at all. Instead, the removed nodes remain available in the pool of potential sensors for the construction of the next barrier. By doing so, we increase the likelihood of being able to construct more barriers in the iterations that remain in the algorithm.

Note that the savings can be significant, in particular, if we consider networks that are very dense, i.e., with many sensors per unit of area. Consider a sequence of sensors that are very close to each other. The location of each sensor in the sequence is only a slight amount to the right of the previous one in the sequence. In this case, a large number of spurious nodes will exist, which can be removed and then reclaimed for the next barrier created by the heuristic.

However, removing nodes to be later reclaimed by other barriers does have its perils: barrier breaches may occur. Consider Figure 4(b). Two barriers are shown. The top barrier (i.e., scheduled first) has circles with solid lines. The lower barrier (i.e. scheduled second) has circles with dashed lines. The top barrier has a detour, and the sensors involved in the detour are marked with thicker lines. Once the detour nodes are removed, the results are shown in Figure 4(c). Note that the lower barrier *does not* have a detour because, although close, the same node is not visited twice. Because of this, the location marked by the star is actually a barrier breach. That is, an intruder can move to this location while the first barrier is active, and once the switch to the second barrier occurs, the intruder is free to move to the lower part of the area.

One method to avoid this barrier breach is to check after the completion of barrier number $m$ whether there is a barrier breach with the previous barrier $m-1$. If so, all the nodes of barrier number $m$ could be removed from the network. This, however, is somewhat drastic because perhaps only a few nodes of the barrier are involved in the breach, and removing the whole barrier is unnecessary.

Instead, as each node is added to the barrier, we can check if adding this node will cause a barrier breach, and if so, this individual node is removed from the network. Again, the node is not permanently removed, it is just set aside and possibly reused in the construction of subsequent barriers.

To detect if a node is causing a barrier breach, we use the following result which we proved in [19].

> *An ordered pair of barriers $(B_1, B_2)$ has a breach iff the floor of $B_2$ intersects the ceiling of $B_1$.*

Figure 5(a) shows the first three nodes of a barrier, and also the left border of the area of interest. Note that the contribution of node 2 to the ceiling or floor of the barrier is not defined until node 3 is chosen. Thus, in our heuristic, when node $i+1$ is chosen, then the arc that node $i$ contributes to its floor can be checked against the ceiling of the previous barrier, and see if they intersect, as shown in Figure 5(b). If so, then node $i$ is removed from the current barrier and no longer considered a candidate for this barrier, while node $i+1$ is returned to the set of candidate nodes for this barrier.

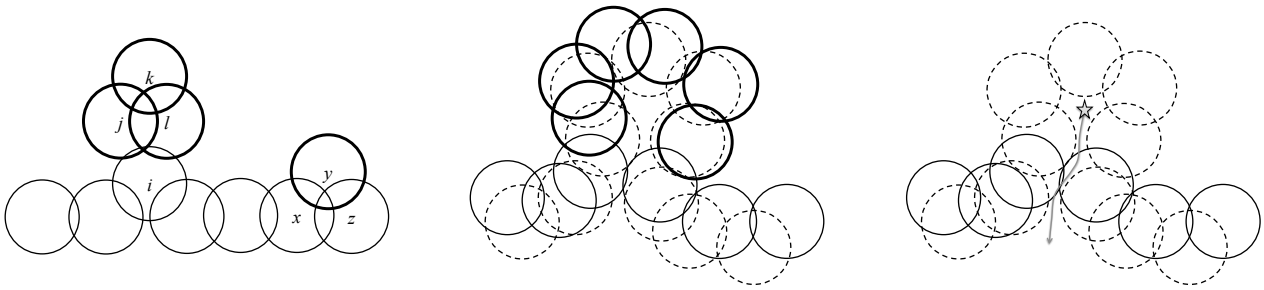
Figure 5(a) also shows the *ceiling points* and the *floor points*. These are the points in the ceiling (respectively floor) where the sensing areas of two consecutive sensors intersect plus the points where the ceiling (respectively floor) intersects the left or right border of the area of interest. We make use of these definitions in the pseudocode of our heuristic, which is presented in Algorithm 1.

## IV. Breach-Free Stint

In addition to the heuristics of [18] [19], we consider a simple modification to the Stint protocol [11]. The heuristics of [18] [19] pre-process the sensor graph before running the Stint algorithm in a manner such that breaches are avoided. We take the opposite approach of first using Stint to obtain the maximum number of barriers, which of course contain a significant number of barrier breaches, and then we eliminate the barrier breaches using the intersection of the floor and ceiling of consecutive barriers.

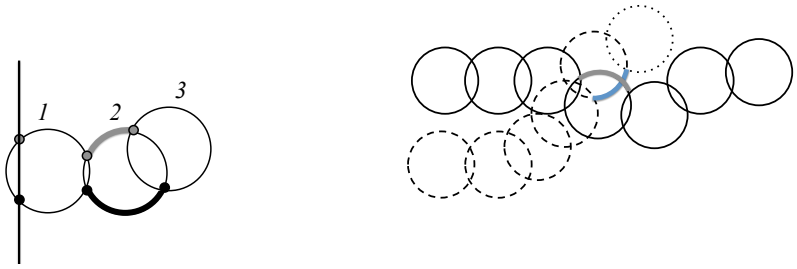
Thus, after we run Stint, we sort the barriers in order of their intersection with the left wall of the area of interest. If there were no barrier breaches, then the barriers would be scheduled from top to bottom. To check for breaches, we

 ISBN: 978-1-61208-499-2 




(a) Sensors to be removed. (b) Potential breach after removal. (c) Breach created.

Figure 4. Compressed Ceilings Method




(a) Node contribution to ceiling and floor. (b) Detecting a barrier breach.

Figure 5. Preventing Barrier Breaches

consider each pair $(B_i, B_{i+1})$ of consecutive barriers. If the floor of $B_{i+1}$ intersects the ceiling of $B_i$, then $B_{i+1}$ is removed from the network. We keep repeating this until all barrier breaches are removed.

## V. Simulation Results

In this section, we compare the performance of our Compressed Ceilings (CC) heuristic presented in Section III against several other heuristics: Ordered Ceilings (OC) that we presented in [19] and overviewed in Section II-D, Max-Flow-Edge-Eraser (MFEE) [18] that we overviewed in Section II-C, and Breach-Free Stint (BFS) that we presented above. We use pure Stint as an upper bound on these heuristics, since it is oblivious to barrier breaches. We only compare against the MFEE heuristic and not against other heuristics in [17] [18] as it was shown in [18] that MFEE is clearly superior to the others.

The area of interest is a square of size $100 \times 100$ meters. We also simulated rectangles of dimensions $60 \times 100$, $80 \times 100$, $100 \times 60$, and $100 \times 80$ meters, but the results were similar. Also, $n$ sensor nodes are randomly deployed in each area, where $n$ ranges from 30 to 80, and the sensing radius ranges from 15 meters to 25 meters. Every point in our plots corresponds to the average of 100 simulations.

Figure 6 shows our first comparison, where the number of sensor ranges from 30 to 80. It shows the results for three different sensing ranges $r$: 15, 20, and 25. Our CC heuristic clearly outperforms all others, being more noticeable as the number of sensor nodes increases or as the radius increases. I.e., as the density of sensor nodes increases, the CC heuristic improves its margin over the other heuristics.

Our second group of simulations are shown in Figure 7. The transmission range varies from 15 to 20, and the number of nodes is 40, 50, or 60. Again CC outperforms all other heuristics. The difference becomes more significant as the number of sensors increases from $n = 40$ in Figure 7(a) to $n = 60$ in Figure 7(c).

A curious observation in both of these figures is that BFS begins to outperform some of the heuristics as the sensor density increases. For example, in Figure 6, BFS begins to outperform MFEE when $r = 20$, and significantly outperforms it when $r = 25$. A similar pattern can be observed in Figure 7 as the number of nodes increases.

In order to explore this phenomenon, we took Figure 6 and extended the number of sensors dramatically, up to 600 sensors. The results are shown in Figure 8.

First note that our CC heuristic significantly outperforms our previous OC heuristic. In addition, in Figure 8(a), where $r = 15$, BFS begins to approach the performance of our earlier OC heuristic. In Figure 8(b), where $r = 20$, BFS outperforms OC staring around 300 nodes which yields about 20 barriers. Finally, in Figure 8(c), where $r = 25$, BFS outperforms even our current heuristic CC, again starting around 300 nodes which yields about 45 barriers. Thus, in very dense networks, a simple variation of the Stint method can actually produce more breach-free barriers than any other current heuristic.

 ISBN: 978-1-61208-499-2 

**Algorithm 1** ***Compressed-Ceilings*** $(N)$
Inputs: sensor node set $N$.
Output: set $O$ of node-disjoint breach-free sensor barriers.

$O \leftarrow \emptyset$;
$N' \leftarrow N$;
**while** exists a sensor in $N'$ whose range crosses the left edge of the area **do**
  $Barrier \leftarrow$ top sensor overlapping left edge of area;
  $lastBarrier \leftarrow \emptyset$;
  $done \leftarrow$ **false**;
  $success \leftarrow$ **false**;
  **while** $\neg done$ **do**
    $s \leftarrow lastSensor(Barrier)$;
    $p \leftarrow$ last point in the ceiling of $Barrier$;
    $Q \leftarrow \{q \mid [\exists\, t \in N', q \in (\text{range}(s) \bigcap \text{range}(t))]\}$;
    $p' \leftarrow$ first point in $Q$ clockwise from $p$ around $s$;
    $s' \leftarrow$ sensor whose range overlaps that of $s$ at $p'$;
    **if** range of $s'$ overlaps the area's left edge **then**
      $success \leftarrow$ **false**;
      $done \leftarrow$ **true**;
    **end if**
    $Barrier \leftarrow append(Barrier, s')$;
    **if** $lastBarrier \neq \emptyset \wedge |Barrier| \geq 2 \wedge intersect(floor(Barrier), ceiling(lastBarrier))$ **then**
      $q \leftarrow next\text{-}to\text{-}last(Barrier)$;
      $N' \leftarrow N' - q$;
      $Barrier \leftarrow Barrier - \{s', q\}$;
    **else**
      **if** range of $s'$ overlaps the area's right edge **then**
        $success \leftarrow$ **true**;
        $done \leftarrow$ **true**;
      **end if**
    **end if**
  **end while**
  **if** $success$ **then**
    $O \leftarrow O \bigcup Barrier$;
  **end if**
  $N' \leftarrow N' - Barrier$;
**end while**
**return** $O$

## VI. Concluding Remarks and Future Work

In this paper, we have refined our previous heuristic to deliver a higher level of performance by carefully eliminating redundant nodes. We have also presented a simple modification of the Stint algorithm that results in a method that outperforms all others when the density of sensors nodes is very high. This points to the possibility that that there is still room for improving our earlier heuristics, because the Stint protocol is oblivious to breaches, and our modification to Stint is straightforward by eliminating any barrier in its output that causes a breach with a previous barrier.

## References


[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, vol. 38, no. 4, Mar 2002, pp. 393–422.

[2] C. Huang and Y. Tseng, "The coverage problem in a wireless sensor network," in ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA), 2003.

[3] H. Zhang and J. Hou, "On deriving the upper bound of -lifetime for large sensor networks," in Proc. of The 5th ACM Int'l Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc), 2004.

[4] Cardei, M., Thai, M.T., Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in INFOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies., vol. 3, March 2005.

[5] M. Thai, Y. Li, and F. Wang, "O(log n)-localized algorithms on the coverage problem in heterogeneous sensor networks," in IEEE Int'l Performance, Computing, and Communications Conference, 2007. IPCCC 2007., April 2007, pp. 85–92.

[6] S. Gao, X. Wang, and Y. Li, "p-percent coverage schedule in wireless sensor networks," in Proc. of 17th Int'l Conference on Computer Communications and Networks, 2008. ICCCN '08., Aug 2008, pp. 1–6.

[7] C. Vu, G. Chen, Y. Zhao, and Y. Li, "A universal framework for partial coverage in wireless sensor networks," in Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th Int'l, Dec 2009, pp. 1–8.

[8] Y. Li, C. Vu, C. Ai, G. Chen, and Y. Zhao, "Transforming complete coverage algorithms to partial coverage algorithms for wireless sensor networks," Parallel and Distributed Systems, IEEE Transactions on, vol. 22, no. 4, April 2011.

[9] S. Kumar, T. Lai, and A. Arora, "Barrier coverage with wireless sensors," in Proc. of the 11th Annual Int'l Conference on Mobile Computing and Networking (MobiCom), 2005.

[10] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage of line-based deployed wireless sensor networks," in INFOCOM 2009, IEEE, April 2009, pp. 127–135.

[11] S. Kumar, T. Lai, M. Posner, and P. Sinha, "Maximizing the lifetime of a barrier of wireless sensors," Mobile Computing, IEEE Transactions on, vol. 9, no. 8, Aug 2010.

[12] H. Yang, D. Li, Q. Zhu, W. Chen, and Y. Hong, "Minimum energy cost k-barrier coverage in wireless sensor networks," in Proc. of the 5th Int'l Conf. on Wireless Algorithms, Systems, and Ap- plications (WASA), 2010.

[13] H. Luo, H. Du, D. Kim, Q. Ye, R. Zhu, and J. Zhang, "Imperfection better than perfection: Beyond optimal lifetime barrier coverage in wireless sensor networks," in Proc. of The IEEE 10th Int'l Conference on Mobile Ad-hoc and Sensor Networks (MSN 2014), Dec 2014.

[14] D. Li, B. Xu, Y. Zhu, D. Kim, and W. Wu, "Minimum (k,w)-angle barrier coverage in wireless camera sensor networks," Int'l Journal of Sensor Networks (IJSNET), 2014.

[15] L. Guo, D. Kim, D. Li, W. Chen, and A. Tokuta, "Constructing belt-barrier providing quality of monitoring with minimum camera sensors," in Computer Communication and Networks (ICCCN), 2014 23rd Int'l Conf. on, Aug 2014.

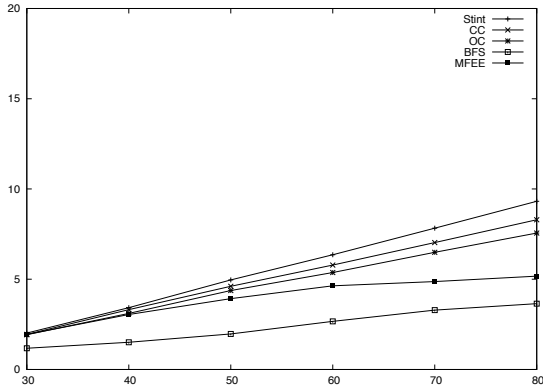
[16] B. Xu, D. Kim, D. Li, J. Lee, H. Jiang, and A. Tokuta, "Fortifying barrier-coverage of wireless sensor network with mobile sensor nodes," in Proc. of the 9th Int'l Conference on Wireless Algorithms, Systems, and Applications (WASA 2014), Jun 2014.

[17] D. Kim, J. Kim, D. L. abd S. S. Kwon, and A. Tokuta, "On sleep-wakeup scheduling of non-penetrable barrier-coverage of wireless sensors," in Proc. of the IEEE Global Communications Conference (GLOBECOM 2012), Dec 2012, pp. 321–327.

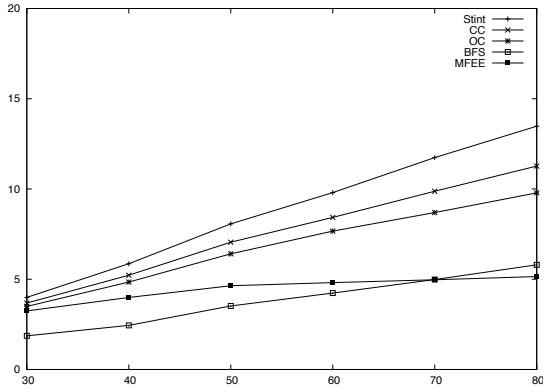
[18] D. Kim, H. Kim, D. Li, S.-S. Kwon, A. O. Tokuta, and J. A. Cobb, "Maximum lifetime dependable barrier-coverage in wireless sensor networks," Ad Hoc Networks, vol. 36, no. 1, Jan 2016.

[19] J. A. Cobb, "Improving the lifetime of non-penetrable barrier coverage in sensor networks," in IEEE $14^{\text{th}}$ International Workshop on Assurance in Distributed Systems and Networks (ADSN), Jun 2015, pp. pp. 1–10.

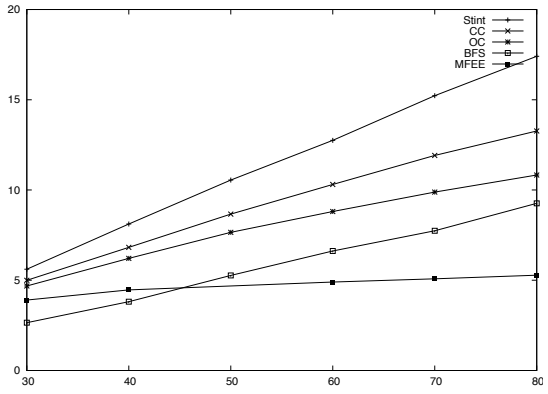

 ISBN: 978-1-61208-499-2




(a) $r$ = 15

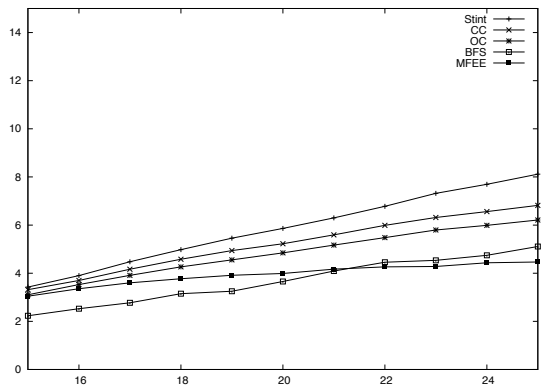



(b) $r$ = 20

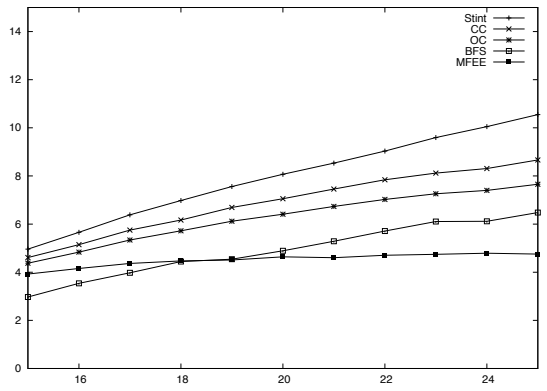



(c) $r$ = 25

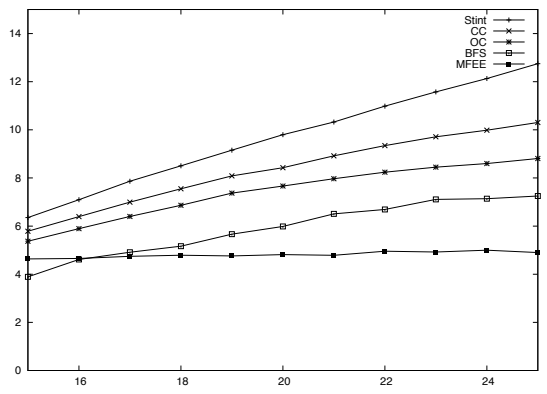
Figure 6. Number of sensors (horizontal) vs. number of barriers (vertical) in $100 \times 100$ region




(a) $n$ = 40

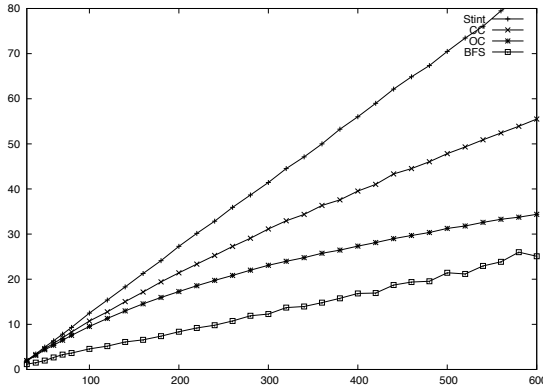



(b) $n$ = 50

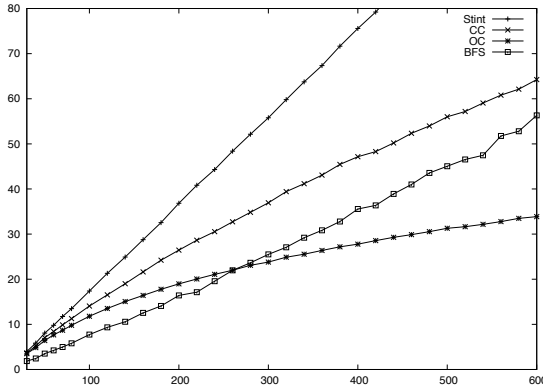



(c) $n$ = 60

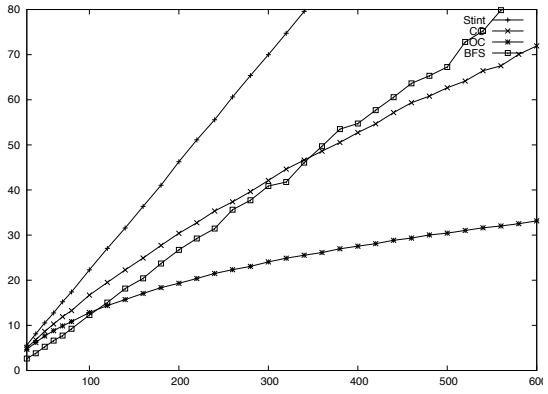
Figure 7. Transmission range (horizontal) vs. number of barriers (vertical) in $100 \times 100$ region




(a) $r = 15$




(b) $r = 20$




(c) $r = 25$

Figure 8. Number of sensors (horizontal) vs. number of barriers (vertical) in $100 \times 100$ region

 ISBN: 978-1-61208-499-2