Incremental Construction of k-Dominating Sets in Wireless Sensor Networks

Mathieu Couture, Michel Barbeau,
Prosenjit Bose and Evangelos Kranakis

School of Computer Science, Carleton University, Herzberg Building
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6 Canada
{mathieu,jit}@cg.scs.carleton.ca
{barbeau,kranakis}@scs.carleton.ca
http://www.scs.carleton.ca
Fax: 1-613-520-4334

Abstract. Given a graph G, a k-dominating set of G is a subset S of its vertices with the property that every vertex of G is either in S or has at least k neighbors in S. We present a new incremental distributed algorithm to construct a k-dominating set. The algorithm constructs a monotone family of dominating sets $D_1 \subseteq D_2 \ldots \subseteq D_i \ldots \subseteq D_k$ such that each D_i is an i-dominating set. For unit disk graphs, the size of each of the resulting i-dominating sets is at most six times the optimal.

Key words: unit disk graph, dominating set, maximal independent set, approximation algorithms, distributed algorithms, fault-tolerance

1 Introduction

An *ad hoc network* is a special type of wireless network where no node has a priori knowledge about the other nodes. Constructing and maintaining a structure allowing nodes to communicate with each other is one of the main challenges of ad hoc networks. Sensor networks are a specific type of ad hoc networks dedicated to a specific task: sensing (light, temperature, humidity, etc.). The dominating set structure helps ad hoc and sensor networks to perform routing. In sensor networks, dominating sets also help the sensing task itself. Since nodes located close to each other sense similar values, only a dominating set of the nodes is needed to monitor an area. This helps prolonging the network's lifetime by turning off the nodes that are not in the dominating set, thereby extending the battery life of these nodes.

Sensor networks typically contain more nodes and each node has less memory than in general ad hoc networks. Therefore, it is important to design algorithms with low memory complexity. An algorithm is *distributed* if the information needed by a node to perform its computation only concerns its direct neighbors. The amount of memory needed by each node to execute a distributed algorithm only depends on the number of its neighbors and not on the network size.

Sensor nodes are more error-prone than nodes in general ad hoc networks. They have limited energy resources and need to be periodically redeployed by adding new nodes to the network. The fact that they are error-prone calls for *fault-tolerance* in the design of algorithms for such networks.

We address the problem of distributively constructing k-dominating sets on unit disk graphs. Unit disk graphs are the standard structure used to model ad hoc and sensor networks. A k-dominating set is a dominating set where each node is either in the dominating set or has at least k neighbors in the dominating set.

We generalize dominating set algorithms based on the idea of maximal independent sets [1, 9, 13] to obtain k-dominating sets. A subset S of the nodes of a graph G is said to be *independent* if it does not contain two adjacent nodes. It is *maximal* if it does not have a proper independent superset. It is straightforward to show that a maximal independent set is also a dominating set. Our algorithm is distributed and, on unit-disk graphs, has a deterministic performance ratio of six. The *performance ratio* of a k-dominating set algorithm is defined as the ratio of the size of the k-dominating set it produces over the size of an optimal (minimum) k-dominating set. It is not position-aware, which means that nodes do not need to know their coordinate in the plane. It also constructs the k-dominating set incrementally. More specifically, it constructs a monotone family of dominating sets $D_1 \subseteq D_2 \ldots \subseteq D_i \ldots \subseteq D_k$ such that each D_i is an i-dominating set. Incremental construction of k-dominating sets is helpful when redeploying sensor networks. When senor nodes in the k-dominating set run out of batteries or experiment failure for diverse reasons, the k-dominating set has to be reconstructed. With an incremental algorithm, reconstruction of a k-dominating set can be done by keeping the current dominators.

The k-dominating set problem has been addressed by Dai and Wu [3] and Kuhn *et al.* [8]. However, our algorithm is the only one which has a constant deterministic performance ratio. It is also the only one to provide an explicit incremental construction. The rest of this paper is organized as follows: in Section 2, we present our algorithm. In Section 3, we analyze its performance ratio. In Section 4, we present some simulation results. We draw conclusions in Section 5.

2 Algorithm

Alzoubi *et al.* [1] and Wan *et al.* [13] addressed construction of a *connected* dominating set. Their algorithm consists of two phases. The first phase constructs a maximal independent set. A maximal independent set is also a dominating set. In this section, we generalize that first phase of the algorithm presented in [1, 13] to obtain a k-dominating set. Our generalization augments a $(k-1)$-dominating set in order to obtain a k-dominating set. More specifically, we want to construct a monotone k-dominating family.

Definition 1. *A* k-dominating family *is a sequence* $D_1, D_2, \ldots, D_k$ *of subsets of vertices of the unit disk graph such that for all* $i = 1, 2, \ldots, k$, D_i *is an* i*-dominating set. A* monotone k-dominating family *is a* k*-dominating family with the additional property that the sequence of dominating sets is monotonically increasing under inclusion, i.e.* $D_1 \subseteq D_2 \subseteq \cdots \subseteq D_k$.

The key idea of our algorithm is that we first construct a 1-dominating set by constructing a maximal independent set. Then, we construct a maximal independent set of the nodes that are not 2-dominated, which gives a 2-dominating set. We repeat the procedure until we have a k-dominating set. The construction of each dominating set is similar to the approach in [1, 13].

We now present an overview of our algorithm. Every node has a unique identifier. In initialization phase, each node sends its identifier to its immediate neighbors. After initialization, two types of messages are used: JOIN(id, i) and GIVE-UP(id, i), where id is the identifier of the sending node and $i = 1 \ldots k$ identifies a round. These messages are only sent to immediate neighbors. The JOIN(id, i) message means that the sender joins the j-dominating sets for $j = i \ldots k$. Such a node is said to be *marked* in round i. The GIVE-UP(id, i) message means that the sender is excluded of the i-dominating set. After transmitting a JOIN message, the sender remains silent. A node is said to be a *candidate* for round i if it is not part of the $(i-1)$-dominating set and it has never sent the GIVE-UP(id, i) message. Following the completion of the initialization phase, every node that has an identifier lower than the ones of all its immediate neighbors sends the JOIN$(id, 1)$ message. The rest of the algorithm is message driven. Algorithm 1 specifies how each node should behave. Note that different nodes may execute simultaneously different rounds.

Figure 1, illustrates the marking process for $k = 1$, 2 and 3. Nodes in black are dominators. Nodes in grey are k-dominated. Nodes in white are

Algorithm 1 DOMINATING SET(id, N, k)

Input: id, the node identifier
N, the list of the neighbors identifiers
k, the required number of dominators for a non-dominating node
Output: $dominator$, a boolean indicating whether the node is a dominator
Local Variables: $round$, the current round
$candidate$, a lookup table indicating whether or not a node n is a candidate to be a dominator in round r (all initial values are **true**)

```
1:  dominator $\leftarrow$ false
2:  $round \leftarrow 1$
3:  if $id < \min(N)$ then
4:      dominator $\leftarrow$ true
5:      send JOIN$(id, 1)$
6:      exit
7:  end if
8:  while $round \leq k$ do
9:      receive $message$
10:     if $message$ is JOIN$(n, r)$ then
11:         send GIVE-UP$(id, round)$
12:         $round \leftarrow round + 1$
13:         for $i = r$ to $k$ do
14:             $candidate[n, i] \leftarrow$ false
15:         end for
16:     end if
17:     if $message$ is GIVE-UP$(n, r)$ then
18:         $candidate[n, r] \leftarrow$ false
19:     end if
20:     if $id < \min\{n \in N | candidate[n, round]\}$ then
21:         $dominator \leftarrow$ true
22:         $round \leftarrow k + 1$
23:         send JOIN$(id, round)$
24:         exit
25:     end if
26: end while
```

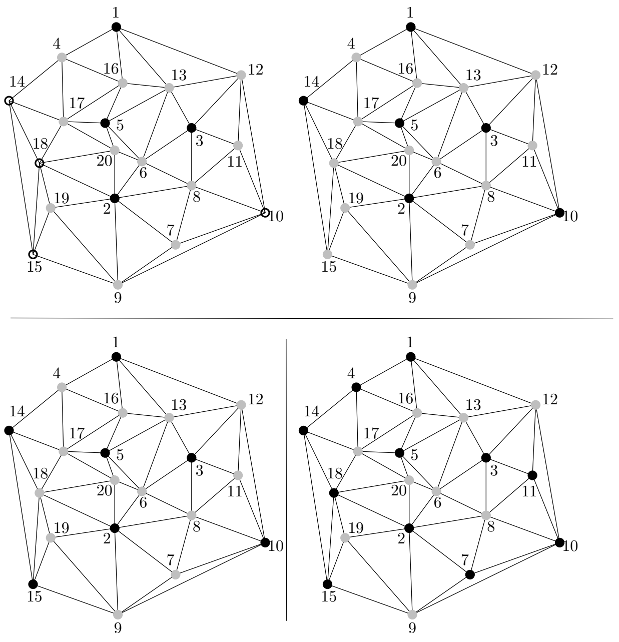

Fig. 1. Marking Process Example for $k = 1$ (above) and $k = 2$ and 3 (below).

not k-dominated. For $k = 1$, nodes 1, 2, 3 and 5 have the smallest identifier among their (0-dominated) neighbors and thus declare themselves dominators. Initially, node 10 can not declare itself a dominator because of nodes 7, 8 and 9. However, after node 2 has declared itself a dominator, nodes 7, 8 and 9 become 1-dominated. Node 10 is then allowed to declare itself a dominator. The same reasoning applies to nodes 1 and 14. The 1-dominating set is then $\{1, 2, 3, 5, 10, 14\}$. For $k = 2$, there is only one new dominator, node 15. For $k = 3$, the new dominators are nodes 4, 7, 11 and 18.

3 Theoretical Properties

In this section, we give an overview of the theoretical properties of our algorithm. We first show that our algorithm computes a valid k-dominating set and a monotone k-dominating family. Then, we analyze the worst case performance ratio of our algorithm. In the latter part, we follow the general idea of Kuhn *et al.* [8]. More precisely, we first show that no unit disk can contain more than a given number of dominators (i.e. $5k$). Then, we use properties of k-dominating sets to show that this leads to a constant performance ratio.

Proposition 1. *Let S_i be the set of nodes that are marked in rounds $j = 0 \dots i$ of Algorithm 1. Then S_i is an i-dominating set.*

Proof: We proceed by induction on i. To make the things simpler, we say that i ranges from 0 to k. The round zero chooses the empty set as a 0-dominating set. The base case is trivial, since all nodes of the graph have at least zero neighbors in the empty set. For the induction case, we have to show that if S_i is a valid i-dominating set, then S_{i+1} is also valid $(i+1)$-dominating set. In order to do this, we proceed by contradiction. Let n_1 be a node that is not $(i+1)$-dominated by S_{i+1}. This means that it has not sent a $\textsc{join}(id, i+1)$ message. Consequently, it must have a neighbor n_2 with a lower identifier that is still a candidate for round $i+1$ (line 20), meaning that it is not $(i+1)$-dominated either (line 11 in n_2, and 17 in n_1). Since n_2 is not $(i+1)$-dominated, by the same reasoning, there must have a node n_3 that is not $(i+1)$-dominated and has a lower identifier than the one of n_2. This process allows to construct a path $n_1, n_2, \dots, n_j, \dots, n_k$ such that none of the n_j is $(i+1)$-dominated, $id(n_1) > id(n_2) > \dots > id(n_j) > \dots > id(n_k)$, and n_k does not have any neighbor with a lower identifier that is not $(i+1)$-dominated. Then, n_k should have elected himself as a dominator, contradicting the fact that it is not $(i+1)$-dominated. Therefore, every node is $(i+1)$-dominated, which completes the inductive case. $\square$

Proposition 2. *For $i = 1 \dots k$, let S_i be defined as above. Then for all $i = 0 \dots k-1$, $S_i \subseteq S_{i+1}$.*

Proof: The monotonicity property claimed in the statement of the proposition is true by construction. That is, let $n \in S_i$. Then, it has been marked in some round $j \leq i < i+1$ and by definition of S_{i+1}, we have $n \in S_{i+1}$. $\square$

Proposition 3. *In any given round, the nodes marked by Algorithm 1 form an independent set.*

Proof: Suppose that in the same round, two adjacent nodes n_1 and n_2 declare themselves dominators. Without loss of generality, suppose n_1 has a lower identifier than n_2. This means that as long as n_1 did not send a give-up message, n_2 can not elect itself a dominator. But since n_1 never sends such a message (no node sends both a give-up and a join message),

n_2 can never declare itself a dominator. This means that no two adjacent nodes can declare themselves dominators. $\square$

Proposition 4. *Let $G = (V, E)$ be a unit disk graph, C be a unit disk and $S \subseteq V$ be the set of nodes marked by Algorithm 1. Then $|S \cap C| \leq 5k$.*

Proof: By proposition 3, S is the union of k independent sets. Since no unit disk can contain more than 5 independent nodes [9], $S \cap C$ can not contain more than $5k$ nodes. $\square$

Proposition 5. *Let $G = (V, E)$ be a graph, S a subset of V, t an integer and $OPT_k = \{v_1, \ldots, v_{|OPT_k|}\}$ an optimal k-dominating set of G. If $|S| > t|OPT_k|$, then there is at least one node $v \in OPT_k$ such that $|N(v) \cap S| > k(t-1)$, where $N(v)$ is the set formed by v and its neighbors.*

Proof: Let S' be $S \setminus OPT_k$. Since $|S'| \geq |S| - |OPT_k| > t|OPT_k| - |OPT_k|$, we have $|S'| > (t-1)|OPT_k|$. For each $v_i \in OPT_k$, define S_i as $N(v_i) \cap S'$. Since each node in S' is adjacent to at least k nodes in OPT_k, we have that

$$\sum_{i=1}^{opt_k} |S_i| \geq k|S'| > k(t-1)|OPT_k|$$

Therefore, by the pigeonhole principle, one of the S_i contains more than $k(t-1)$ nodes. The result follows from the fact that $S_i \subseteq N(v_i) \cap S$. $\square$

Theorem 1. *Let $G = (V, E)$ be a unit disk graph, $S \subseteq V$ the set of nodes marked by Algorithm 1 and OPT_k an optimal k-dominating set. Then $|S| \leq 6|OPT_k|$. In other words, the performance ratio is not greater than six.*

Proof: Suppose $|S| > 6|OPT_k|$. By proposition 5, there is at least one node $v \in V$ such that $|N(v) \cap S| > 5k$. But this contradicts proposition 4, and therefore $|S| \leq 6|OPT_k|$. $\square$

We now show that for any k, there exists graphs for which our algorithm has a performance ratio of five. It is an open question whether or not it is possible to close the gap between five and six. First, we need the following lemma:

Lemma 1. *Let $\triangle ABC$ be an isosceles triangle such that $\angle BAC = \angle ACB = \phi$, p be a point located on the line AB such that A is between p and B, and q be a point located on the line CB such that C is between q and B. Then $|pq| > |AC|$.*

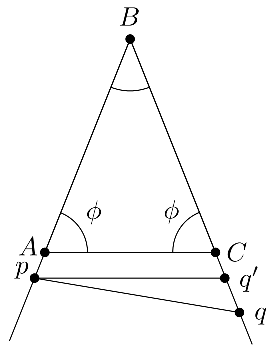

Fig. 2. Lemma 1.

Proof: If $|pB| = |qB|$, then $\triangle pBq$ is similar to $\triangle ABC$, and $|pB| > |AB|$ implies $|pq| > |AC|$. Suppose now that $|pB| < |qB|$, and let q' be the point located on the line CB such that C is between q' and B and $|q'B| = |pB|$. By the first case, $|pq'| > |AC|$. Now, since $\triangle ABC$ is isosceles, $\phi < \frac{\pi}{2}$, and since $\angle pq'q = \pi - \phi$, we have that $\angle pq'q > \frac{\pi}{2}$. Therefore, $\angle pq'q$ is the largest angle of $\triangle pq'q$, meaning that its opposite side, pq, is the largest side. In particular, we have $|pq| > |pq'| > |AC|$. The case where $|pB| > |qB|$ is identical, which completes the proof of the lemma. $\square$

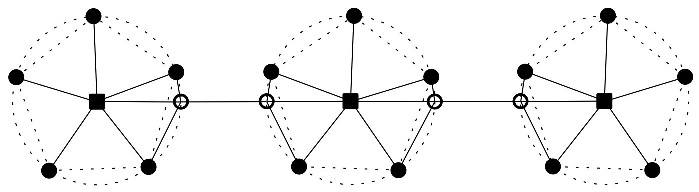

Fig. 3. Lower bound of five for Algorithm 1.

Proposition 6. *The worst case performance ratio of Algorithm 1 is at least five.*

Proof: For $k = 1$ and $n = 6$, place five nodes equally spaced on the boundary of a circle of radius 1, and place one other node in the center of that circle. Since the circle has radius 1, the center node shares an edge with all the other nodes. Also, since the distance between every pair of nodes on the circle is at least $2\sin\frac{\pi}{5} > 1$, there is no other edge in the unit disk graph. In the remainder of the proof, this basic structure will be referred to as a *star*, the node placed in the center of the circle will be referred to as *the center* of the star and the five nodes on the boundary of the circle will be referred to as the *branches* of the star. The center of a star forms a dominating set of the whole star. However, if the center happens to be given a higher identifier than one of the branches, all branches would be marked as dominators, leading to a performance ratio of five.

Figure 3 depicts how to connect several stars to build cases with n as large as desired. More precisely, we show how to construct examples of size $14 + 8m$, for any given m (in Figure 3, $m = 1$). The construction goes as follows: place $m + 2$ stars on a horizontal line such that their centers are placed at x-coordinates $0, 3, 6, \ldots, 3(m+1)$ and no branch lies on the horizontal line. Since the circles in which the stars are inscribed are at distance at least 1 from each other, the only edges of the graph so far are the ones linking the branches of the stars to their centers. All that remains is to connect the graph. In order to do so, add nodes on the intersection of the inscribing circles with the horizontal line. These nodes will be referred to as *bridging nodes*. Since the centers of two consecutive stars are at distance 3 from each other, the two bridging nodes between them are at distance 1 from each other. Therefore, there is an edge between two bridging nodes which are between the centers of two consecutive stars, making the whole network connected. To see how the performance ratio of five can be reached, notice that the star centers form a dominating set of size $m + 2$. However, since the set of all branches form an independent set, it could be that those nodes would be marked as dominators, leading to a dominating set of size $5(m + 2)$, which gives a performance ratio of at least five.

For $k > 1$, Figure 4 shows how to generalize the star structure. The goal is to map to each node of the star a set of k nodes such that:

1. nodes mapped to the center share an edge with every other node and
2. nodes mapped to the branches only share edges with nodes mapped to the center and nodes mapped to the same branch.

In order to achieve this, draw a regular pentagon having side length of 1. Let C be the inscribing circle of that pentagon and $r = \frac{1}{2\sin(\frac{\pi}{5})}$ be the radius of C. Now, let C_1 and C_2 respectively be the circles having the same center as C and radii $r_1 = \frac{1-r}{2}$ and $r_2 = r + r_1$. For each vertex v_i of the pentagon (i from 1 to 5), let s_i be the half-line from the center of

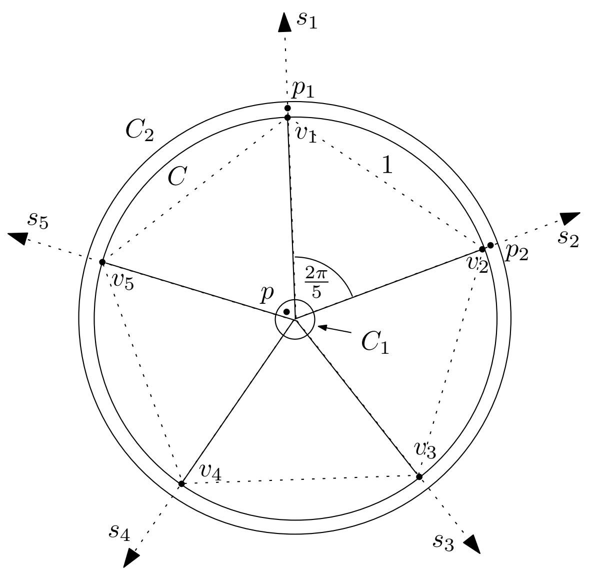

Fig. 4. Widget for $k > 1$.

C through v_i. Now, let p be a point located inside C_1, and p_1 and p_2 be two points located on some s_i and s_j ($i \neq j$), between C_2 and C. Then, Lemma 1 tells us that

$$|p_1, p_2| > |v_1, v_2| \geq 1$$

and from the triangle inequality, we have

$$|p, p_1| \leq r_1 + r_2 = 2(\frac{1-r}{2}) + r = 1.$$

Similarly, $|p, p_2| \leq 1$. The construction we need is then the following: place k points inside C_1 and k points on each of the s_i between C and C_2. We call the result of that construction a *generalized star*. The points located inside C_1 form a k-dominating set, but the algorithm may mark all nodes located on the s_i. Since there are $5k$ such points, the performance ratio is five in that case. To construct a lower bound example with $k > 1$ for large n, we link the generalized stars in a similar fashion as for the case $k = 1$. $\square$

4 Simulation Results

We have generalized an existing independent set-based algorithm [1, 9, 13] in order to incrementally construct a k-dominating set. We have chosen to generalize this specific algorithm because it is distributed and has constant performance ratio. By simulation, we compare our algorithm with k-generalized versions of other available algorithms. Stojmenovic *et al.* [12] suggested the following heuristic to improve the independent set algorithm of [1, 9, 13]: instead of ordering the nodes according to their identifier, order them according to their degree first and then their identifier. Higher priority is granted to nodes having higher degree. The performance ratio is still at most five, but it has not been proven it is actually better than that. For the k-dominating set problem, it is not desirable to favor higher degree nodes. The reason is that nodes having degree less than k cannot have k dominating neighbors, so they must necessarily be in the k-dominating set.

Selecting nodes of higher degree is the same idea that is behind the greedy set-cover algorithm [4]. The greedy set-cover algorithm first favors nodes that dominate the largest number of nodes not yet dominated. Although this is a global selection criterion, it still has to be examined. At first sight, since it does not have constant performance ratio (its performance ratio is $H(\Delta)$, where Δ is the maximum degree of a node in the network and H is the harmonic function), one would believe that it would not perform as well as our algorithm. However, it turns out that in order to have $H(\Delta) > 5$, we need Δ to be at least 83, and to reach six, we need Δ to be at least 226. Since it is not likely to have nodes having that many neighbors in real situations, this algorithm still deserves attention.

In this section, we discuss simulation results comparing Algorithm 1 with k-generalized versions of both the algorithm presented in [12] and the greedy algorithm. We also compare it with the greedy construction of a maximal independent set. The k-generalized versions of those algorithms work the same way we generalized the maximal independent set algorithm: for $k = 1$, we run the standard algorithm on all nodes. For $k \geq 2$, we run the standard algorithm on nodes that are not yet k-dominated. We ran our simulations 200 times for networks of 200 nodes. We have chosen a communication range such that with high probability, the network is connected. According to Penrose [10, 11], for any integer $k \geq 0$ and real constant c, if the nodes have identical radius r given by the formula:

$$r = \sqrt{\frac{\ln n + k \ln\ln n + \ln(k!) + c}{n\pi}}$$

then the network is $(k+1)$-connected with probability $e^{-e^{-c}}$ as n goes to infinity. For $n = 200$, choosing $k = 1$ and $c = 5$, we then obtain that for a radius of $r \approx 0.138$, the network is 2-connected with probability 0.99.

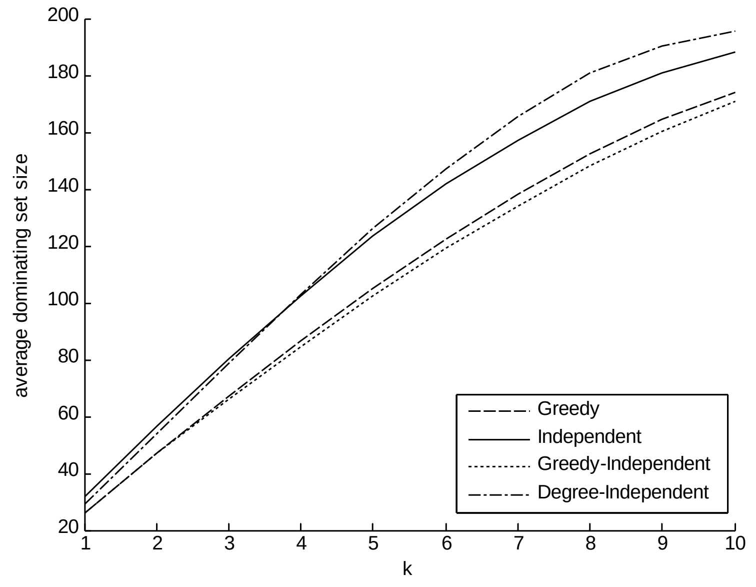

Fig. 5. Average dominating set size for 200 nodes.

Figure 5 shows the simulation results we have obtained. The algorithm which performed the best is the one in which we greedily constructed an independent set. Not far behind is the greedy algorithm. It is worth noting that even if those algorithms perform slightly better, neither of the two are distributed. This is because the greedy choice of the next node to be marked is based on global criteria. For the two distributed algorithms, it is interesting to note that the one using the ordered pair degree-id only performs better for small values of k (5 and less). After that, it is the one simply based on identifiers which performs better. With a 95% certainty, the expected values of the size of the dominating sets was at most ± 0.67 node.

Unfortunately, Figure 5 does not show the optimal solution. Since the dominating set problem is NP-complete, only exponential time algorithms are known to solve the problem. This is why only small instances of the problem can be addressed by simulation. Figure 6 compares the same algorithms with the optimal solution for a network of 35 nodes. We ran over 200 simulation cases. In that case, with a 95% certainty, the actual expected values of the dominating sets size was at most ± 0.28 nodes. Figure 7 shows the average performance ratio we obtained for each algorithm. For small values of k, the two global greedy algorithms are the best, followed by the distributed algorithm granting priority to

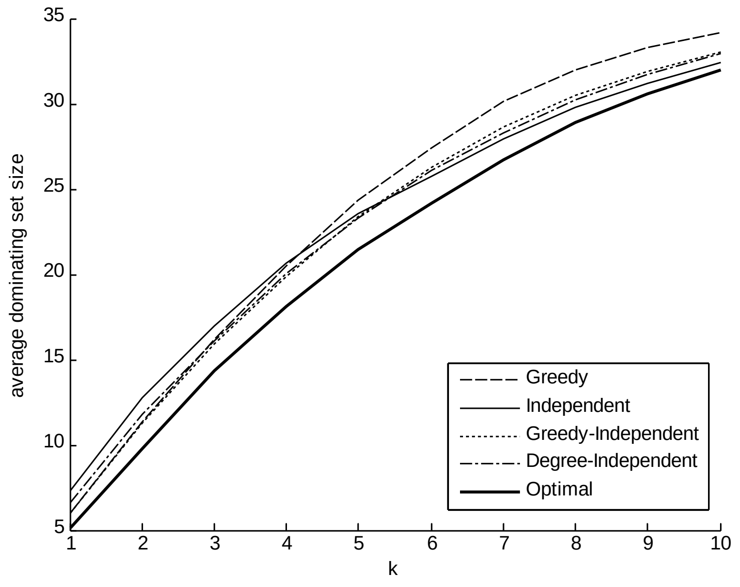

Fig. 6. Average dominating set size for 35 nodes.

high degree nodes. The algorithm simply based on identifiers performs the worst. However, as k grows, the results change completely. The algorithm simply based on identifiers becomes the best, and the basic greedy algorithm becomes the worst. The algorithm constructing independent sets by granting priority to high degree nodes performs slightly better than the greedy construction of an independent set.

Exact simulation values can be found in the technical report version of this paper [2].

5 Conclusion

In this paper, we have introduced a new algorithm to address the k-dominating set problem. Our algorithm has a deterministic performance ratio of six. The previously best algorithm had an expected performance ratio of $O(k)$ for an unspecified constant [8]. We have shown that the size of the k-dominating set our algorithm produces may be five times bigger than the optimal one. However, it is an open issue whether or not the gap between five and six can be closed. The expected performance ratio is also unknown.

Simulation results have shown that in some cases, the k-generalized version of the greedy dominating set algorithm performs better than ours.

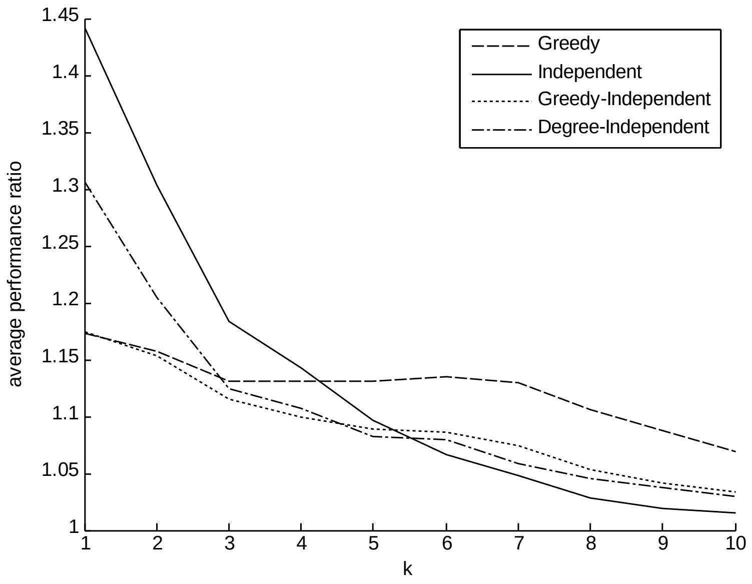

Fig. 7. Average performance ratio for 35 nodes.

Besides their worst-case performance ratio, an other important difference between the greedy dominating set algorithm and ours is that one is global while the other is distributed. We believe that differences between the performance of global, distributed and local algorithms is an interesting research avenue. Important work in that field has been done by Kuhn *et al.* [6, 7] and Kuhn [5].

Acknowledgment

The authors graciously acknowledge the financial support received from the following organizations: Natural Sciences and Engineering Research Council of Canada (NSERC) and Mathematics of Information Technology and Complex Systems (MITACS).

References

[1] K. M. Alzoubi, P.-J. Wan, and O. Frieder, Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 157–164, ACM Press, New York, NY, USA, 2002.

[2] M. Couture, M. Barbeau, P. Bose, and E. Kranakis, Incremental construction of k-dominating sets in wireless sensor networks. Tech. Rep. TR-06-11, School of Computer Science, Carleton University, Ottawa, Ontario, Canada, 2006.

[3] F. Dai and J. Wu, On constructing k-connected k-dominating set in wireless networks. In *IPDPS*, IEEE Computer Society, 2005.

[4] D. S. Johnson, Approximation algorithms for combinatorial problems. In *STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing*, pp. 38–49, ACM Press, New York, NY, USA, 1973.

[5] F. Kuhn, The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives. In *PhD Thesis, ETH Zurich, Diss. ETH No. 16213*, 2005.

[6] F. Kuhn, T. Moscibroda, and R. Wattenhofer, What cannot be computed locally! In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pp. 300–309, ACM Press, New York, NY, USA, 2004.

[7] F. Kuhn, T. Moscibroda, and R. Wattenhofer, On the locality of bounded growth. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pp. 60–68, ACM Press, New York, NY, USA, 2005.

[8] F. Kuhn, T. Moscibroda, and R. Wattenhofer, Fault-Tolerant Clustering in Ad Hoc and Sensor Networks. In *26th International Conference on Distributed Computing Systems (ICDCS), Lisboa, Portugal*, 2006.

[9] M. Marathe, H. Breu, S. Ravi, and D. Rosenkrantz, Simple heuristics for unit disk graphs. *Networks*, **25**:59–68, 1995.

[10] M. D. Penrose, The longest edge of the random minimal spanning tree. *The Annals of Applied Probability*, **7(2)**:340–361, 1997.

[11] M. D. Penrose, On k-connectivity for a geometric random graph. *Random Struct. Algorithms*, **15(2)**:145–164, 1999.

[12] I. Stojmenovic, M. Seddigh, and J. Zunic, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, **13(1)**:14–25, 2002.

[13] P.-J. Wan, K. M. Alzoubi, and O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks. *Mob. Netw. Appl.*, **9(2)**:141–149, 2004.