Blackbone2, an Efficient Deterministic Algorithm for Creating 2-Connected m-dominating Set-Based Backbones in Ad Hoc Networks

Julien Schleich
University of Luxembourg
FSTC - CSC
Luxembourg
julien.schleich@uni.lu

Grégoire Danoy
University of Luxembourg
FSTC - CSC
Luxembourg
gregoire.danoy@uni.lu

Pascal Bouvry
University of Luxembourg
FSTC - CSC
Luxembourg
pascal.bouvry@uni.lu

Le Thi Hoai An
University Paul Verlaine - Metz
LITA
France
lethi@univ-metz.fr

ABSTRACT

This paper introduces Blackbone2, a novel fully decentralized algorithm that aims at creating a robust backbone in ad hoc networks. Backbone robustness is supported by a 2-Connected m-dominating Set, $2, m$-CDS, and decentralization relies on the usage of two rules that only require two-hop knowledge in order to reduce the use of bandwidth. Blackbone2 deterministic approach guarantees a density-independent valid solution and is proved correct. The algorithm is also characterized by its efficient theoretical computation time, $\mathcal{O}(\Delta^2)$ with Δ the average number of neighbors, which outperforms known solutions. The domination parameter, m, can be increased without changing the theoretical computation time. Efficiency of the Blackbone2 algorithm compared to the equivalent literature solutions is illustrated through simulations of a large panel of networks with a wide density range.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology, Wireless communication; G.2.2 [**Graph Theory**]: Network problems, Graph algorithms

General Terms

Algorithms, Design, Performance, Theory

Keywords

k-connected m-dominating Set, Wireless Sensor Networks, Localized algorithm

1. INTRODUCTION

Ad hoc networks distinguished themselves clearly from other communication networks with many features such as absence of a fixed infrastructure, wireless multihop communication, and strict resource limitations (e.g., limited bandwidth and energy resources).

In this kind of networks, the shared medium and the lack of global coordinator implies that the node throughput declines rapidly to zero as the number of nodes in the network increases [6]. In order to deal with large-sized networks, the creation of a virtual backbone is a commonly proposed solution [3, 7, 12, 14–16]. Backbone formation leverages the scalability limits of traditional routing for ad hoc networks by selecting a subset of the network nodes, the so-called backbone, that is responsible for performing and managing multipoint communication (routing, multicast, broadcast).

One of the most studied solution to create a virtual backbone is the computation of a Connected Dominating Set (CDS) in the communication graph representing the network. Given a simple undirected graph $G = (V, E)$, where V is a set of vertices representing the hosts and E is a set of undirected edges representing the links. A subset $V' \subset V$ is a dominating set of a graph $G = (V, E)$ if every vertex not in V' is connected to at least one member of V' by some edge. A subset $V'' \subset V$ is a connected dominating set of graph $G = (V, E)$ if V'' is a dominating set of G and the subgraph induced by V'' is connected.

The quality of a CDS depends on its size corresponding to the number of nodes in V''. Indeed, if less nodes are in charge of the routing process, the overhead due to routing will be less important. The problem of finding the CDS of the smallest size is called *Minimum CDS* and many approximation algorithms have been proposed [1, 3, 12, 15, 16] to solve this problem in a decentralized way with local information.

The major drawback of the CDS approach in an ad hoc context is its poor reliability due to node failure. Indeed, most of the time, loosing one node that is part of the virtual backbone will induce its breakage. In order to ensure the robustness of the structure, we add the following constraint to the generated CDS: when the topology graph permits it, every couple of node of the backbone should be able to communicate through two independent paths.Two paths are independent if they do not have any internal vertex in common. Computing such a CDS is equivalent to generating a 2-connected dominating set.

To solve this problem we propose *Blackbone2*, a fully decentralized algorithm for computing CDS or 2-Connected m-Dominating Set. To compute a global solution, all the nodes of the network require a 2-hop knowledge, which can easily be gathered via beaconing packets. For bandwidth consideration no other message has to be exchanged.

The remainder of this paper is organized as follows. In Section 2, we review some existing CDS and k, m-CDS construction algorithms. In Section 3, we describe an algorithm for constructing a 2-Connected m-Dominating Set in an efficient and a decentralized way. and the Section 4 provides an analysis of its theoretical performances and in Section 5, we evaluate the performances of our proposed algorithm through simulation. Finally, we conclude this paper and discuss some future research directions in the last section.

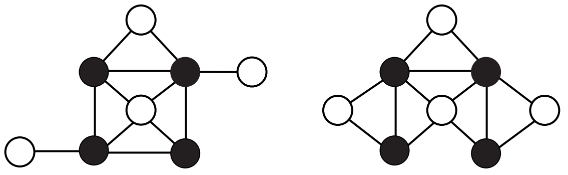
2. RELATED WORK

The most common approach to create a backbone is to partition the network into clusters, composed of a set of ordinary nodes and one clusterhead. After this step, connections are established between clusterheads in order to obtain a connected structure. As stated in [2], backbone formation corresponds to computing a Connected Dominating Set (CDS) of the nodes in the network topology graph. In this graph, $G = (V, E)$, nodes represent the network devices and an edge between two nodes exists if and only if they are neighbors, i.e. they are in each other's communication range.

Many different algorithms for computing CDSs can be found in the literature [1, 3, 8, 12, 16]. Wu *et al.* [16] proposed a localized connected dominating set approach, based on the nodes marking rule (*true* means *in the backbone*) and two pruning rules. A generalization of the pruning rules has been proposed by Dai *et al.* [3]. This rule checks if a set of k nodes covers the neighbor set. This modification achieves better results concerning the size of the CDS.

Approximation algorithms have also been proposed to create CDS. These algorithms are generally compared using their approximation factor n, which means that in the worst case, the size of the solution produced by the algorithm is n times the size of the optimal solution. Alzoubi *et al.* [1, 12] proposed an algorithm with an approximation factor of 8 and in [8], Li *et al.* introduced a completely localized one-phase distributed algorithm, r-CDS, with a approximation factor of 172. In [2], the authors proposed a simulation-based comparison of the most representative solutions to create CDS. In this work, the studied algorithms are sorted based on their degree of localization, which is a measure related to the locality of the information required by the algorithm (the higher, the more local information nodes need to gather).

As we are in an ad hoc context, the network may change during the simulation, consequently the CDS should evolve in order to reflect these changes. Indeed, we may have links breaks between two nodes of the CDS (mobility), or a CDS node might disappear (switch off, out of battery). As these nodes are the backbone of the network, these events may break some paths which will negatively affect the quality of service. To enhance the robustness of the backbone, k-Connected m-Dominating Sets (k, m-CDS) have been recently studied for ad hoc networks. The requirement of k-connectivity guarantees that between any pair of backbone nodes there exists at least k independent paths. The requirement of m-domination takes care of fault tolerance for dominatees, which ensures that every dominatee has at least m adjacent dominator neighbors. These two properties are displayed in Figure 2.



(a) 2-connected dominating set (b) connected 2-dominating set

Figure 1: Robustness properties

To solve this difficult problem, several approximation algorithms have been proposed. However most of them are centralized, or have a high message complexity.

In [10, 11, 13, 18], the authors proposed centralized algorithms. In [13], Wang *et al.* proposed CDSA, a 64-approximation centralized algorithm that is only applicable in the case where $k = 2$ and $m = 1$. Shang *et al.* [10] proposed three centralized algorithms to construct 1-connected m-dominating set, 2-connected m-dominating set, and $k \geq 3$-connected m-dominating set. In [11], the centralized algorithm requires that the input graph is at least $\max(k, m)$ connected. Wu *et al.* [18] proposed CGA, a centralized algorithm, that in a first time creates a m-dominating set, and then augments it until it becomes k-connected. In [17], the authors proposed a centralized algorithm, ICGA, that has a constant performance ratio and can construct k, m-CDS for general k and m.

Centralized algorithms will generally obtain better theoretical results, because they benefit from the network global knowledge. However gathering such knowledge is not realistic in an ad hoc network. As a matter of fact, only decentralized solutions can be implemented in such networks. Decentralized algorithms have been proposed in [4, 17, 18]. Dai *et al.* proposed four localized k-CDS algorithms, where $k = m$. The first one is a probabilistic approach which requires the network size and the node density to compute a probability p_k of a node k to be in the backbone. The second one maintains a fixed node degree in the backbone and also requires network size and density. The third one is a deterministic approach which is an extension from the coverage condition introduced by the same authors to construct 1-CDS [4]. This approach checks if there are k independent paths composed of high priority nodes between each pair of neighbors. The last one is an hybridization of probabilistic and deterministic approaches. In [18], the authors proposed a localized version of CGA called DDA. However, DDA requires a huge number of messages, which is a major drawback for a real application in ad hoc networks. In [17], a distributed algorithm, LDA, is proposed to overcome the problems of DDA. However, it requires the maximal node degree to be constant and it uses CDS-BD-D [7], which requires the election of a root node in the network.

In summary, most of the proposed solutions to create a k, m-CDS for managing ad-hoc networks are not realistic enough (central approach), or not efficient because of the huge number of message that they require. In this paper, in order to overcome current limitations of existing algorithms, we propose, *Blackbone2*, a decentralized algorithm that requires very few messages and has a very efficient computation time. However, *Blackbone2* is not designed to create k, m-*CDS* for any k and m. We restricted ourselves to $k = \{1, 2\}$ and $m \in \mathcal{N}^+$.

3. BLACKBONE2 ALGORITHM

This section presents the Blackbone2 algorithm and two variations. In a first time some notations are introduced and the general characteristics of the algorithm are described. Then the two main properties, biconnectivity and m-domination are developped and solutions are proposed to create a backbone that fits their requirements. Finally the main algorithm and its variations are presented.

3.1 Preliminaries

Let $N_1(v)$ be the 1-hop neighborhood of a node v. Let $N_2(v)$ be the 2-hop neighborhood of a node v. Let $N_{1\cup 2}(v) = N_1(v) \cup N_2(v)$ be the complete neighborhood of node v. Let $CN(x, y)$ be the common neighbors of x and y.

3.2 Decentralized algorithm with 2-hop knowledge

The main goal of this work is to create a robust topology for ad hoc networks. As mobility may induce changes in the network topology graph (loss of nodes and/or loss of edges), node-disjoint alternative paths are required to reduce the impact of these unpredictable events on the quality of service of the real communication. Consequently we choose to develop an algorithm that creates a 2-connected m-dominating set. Finding a minimum 2-connected m-dominating set is NP-hard [5] and mobiles nodes (e.g. sensors) are usually lightweight devices, therefore a heuristic approach is proposed.

Blackbone2 is decentralized and only requires 2-hop knowledge to take decisions. To gather these pieces of information, all the nodes of the network broadcast beacons containing the list of their direct (1-hop) neighbors. As nodes are moving, topology may change, therefore every node has to regularly update its stored information. From a node point of view, if one of its neighbors does not update its neighbors list during some pre-defined time, then the information about this neighbor is deleted.

3.3 Blackbone2 color scheme

Two colors are used to represent the state of a given node. Black means "backbone member" and white "not backbone member". Black nodes regularly check if they are still needed to guarantee the connectivity of the backbone, and white nodes periodically check if they are needed to fulfill the minimum requirements.

Let $N_1^B(v)$ be the black nodes in the 1-hop neighborhood of node v. Let $N_2^B(v)$ be the black nodes in the 2-hop neighborhood of node v. Let $N_{1\cup 2}^B(v) = N_1^B(v) \cup N_2^B(v)$ be the complete black neighborhood of node v. Let $CN^B(x, y)$ be the common black neighbors of x and y. The equivalent sets are defined for white nodes: $N_1^W(v)$, $N_2^W(v)$, $N_{1\cup 2}^W(v)$ and $CN^W(x, y)$.

3.4 2-Connected structure

The main idea of our algorithm is to create a 2-connected structure. To achieve this property, another graph-related notion will be used: articulation points, also called cut-vertices.

3.4.1 *Articulation point and biconnectivity*

Definition 1. *An articulation point or cut vertex is a vertex that if removed (along with all edges incident with it) produces a graph with more connected components than the original graph.*

Definition 2. *A biconnected graph is a connected graph with no articulation vertices.*

Figure 2(a) is a classical example for simple biconnected graph. Indeed, if one node is removed, the remaining graph will still be connected. On the contrary, Figure 2(b) shows a graph with one articulation point, the grey node. To construct 2-connected backbones, we used definition 2. Let us see how to detect articulation vertices.

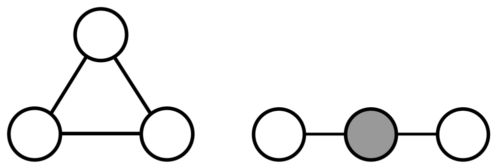

(a) No cut vertex (b) One cut vertex

Figure 2: Illustration of articulation point

3.4.2 *Algorithm to compute articulation points*

An efficient algorithm to compute the cut-vertices has a complexity of $\mathcal{O}(|V| + |E|)$. In order to do the processing, some extra data are required for every node. Let v be a node of $G = (V, E)$.

- Num(v): the visit number obtained from a depth-first search (from any node of the graph)
- Low(v): lowest-numbered vertex reachable from v using 0 or more spanning tree edges and then at most one back edge

Finally Low(v) is the minimum of:

- Num(v)
- Lowest Num(w) among all back edges (v,w)
- Lowest Low(w) among all tree edges (v,w)

Num(v) and Low(v) can be computed in $\mathcal{O}(|V|+|E|)$ with depth-first search for example. Algorithm 1 computes all articulation points. However, in our case, we only require to know if an articulation point exists, that is why we can even stop the computation when an articulation point is found. This optimization reduces computation in simulations but does not change the theoretical computation time.

3.4.3 *Global property*

Let $G(v) = (V(v), E(v))$ be the local graph of the node v. This graph represents what v knows about its neighborhood. Blackbone2 only requires two-hop neighborhood to

Algorithm 1: findArt: an efficient algorithm for articulation point detection

```
Input: Vertex v
Visited(v) = true;
Low(v) = Num(v) = counter++;
for each w adjacent to v do
    if not(Visited(w)) then
        Parent(w) = v;
        findArt(w);
        if Low(w) >= Num(v) then
            v is an articulation point
        Low(v) = min(Low(v), Low(w));
    else
        if Parent(v) != w then
            Low(v) = min(Low(v),Num(w));
```

take a decision, that is why $V(v) = N_{1\cup 2}(v)$. $E(v)$ contains a restricted set of edges defined by:

$$E(v) = \Big\{(u, w) \in E / u \in N_1(v) \wedge w \in N_{1\cup 2}(v)\Big\}$$

Let $G^B(v) = (V^B(v), E^B(v))$ be the local graph of the node v induced by black nodes only. $V^B = N^B_{1\cup 2}(v)$ and $E^B(v)$ represents the edges between two nodes of V^B. In a more formal way:

$$E^B(v) = \Big\{(u, w) \in E / u \in N^B_1(v) \wedge w \in N^B_{1\cup 2}(v)\Big\}$$

Let $G^B = (V^B, E^B)$ be the global black graph obtained by the union of all local black graph.

$$G^B = \cup_{v\in V^B} G^B(v)$$

PROPERTY 1. *If $\forall v \in V$, $G^B(v)$ is biconnected, then G^B is biconnected.*

3.5 m-domination

Having at least 2 node-disjoint paths between any pair of nodes in a backbone increases its tolerance to faults. However, it does not improve the fault-tolerance for nodes outside the backbone. Blackbone2 is able to change the domination parameter m, i.e. the nodes outside the backbone will have at least m direct neighbors inside the backbone.

3.5.1 m-domination property

DEFINITION 3. *A subset $V' \subset V$ is m-dominating if $\forall u \in V - V'$ there are $v_1...v_m \in V'$ for which $(u, v_1)...(u, v_m) \in E$.*

As we are dealing with decentralized algorithm, our solution only considers the local point of view of a node. For a given node v, the following property has to be checked:

$$\forall w_i \in N^W_1(v),$$

$$\exists z_1, ...z_m \in N^B_{1\cup 2}(v) \Big/ (z_1, w_i), ..., (z_m, w_i) \in E(v)$$

In a less formal way, every white 1-hop neighbor should have m black neighbors (1 or 2 hop).

Algorithm 2: m-domination checking algorithm

```
Input: m, the domination parameter
Output: true if the subgraph is m-dominating,
        false otherwise
dominating = true;
for each $w \in N^W_1(v)$ do
    if $|N^B_1(w)| >= m$ then
        dominating = false;
        break;
return dominating;
```

3.5.2 m-domination checking algorithm

The proposed algorithm, is the direct translation of definition 3. Algorithm 2 checks if for a given node v all its 1-hop white neighbors are covered by at least m black nodes, without considering itself. With an appropriate data structure, this algorithm is processed in $\mathcal{O}(|\Delta|)$.

3.5.3 Global property

Let DS(G) be a dominating set of a graph $G = (V, E)$.

PROPERTY 2. *If for all node $v \in V$, $G(v)$ is m-dominated by $DS_{G(v)}$, then the $\cup_{v\in V}(DS_{G(v_i)})$ is a m-dominating set of G.*

3.6 Main algorithm

Blackbone2 algorithm is based on two rules, but only one will be executed depending on the current node state, i.e. its current color. Indeed, when a node v is outside the backbone a construction rule will be triggered to determine if v can stay ouside the backbone or has to be part of it. Based on the same idea, backbone nodes regularly check with a pruning rule if they are still required to fulfill the backbone properties.

3.6.1 Subgraph for biconnectivity checking

For the connectivity property we will not consider the complete local graph $G(v)$, but a subgraph based on the black nodes set, $N^B_{1\cup 2}(v)$. More precisely we will only consider 1-hop black nodes and the useful 2-hop black nodes, i.e. black nodes that help for the 2-connectivity. To do so, 2-hop black nodes with less than 2 black neighbors will be removed from this set. Indeed, even if node v enters the backbone it will not change the fact that the subgraph is not biconnected. Moreover the considered subgraph does not contain the node v.

Let $V_{ul}(v)$ be the set of useless 2-hop black nodes:

$$V_{ul}(v) = \{u \in N^B_2(v) \,/\, |N^B_1(u)| > 1\}$$

In a more formal way, the set of considered nodes $V_c(v)$ is:

$$V_c(v) = N^B_{1\cup 2}(v) \setminus V_{ul}(v)$$

Let $G_c(v)$ be the graph induced by the set of nodes $V_c(v)$.

3.6.2 Pruning rule

The pruning rule, algorithm 3, checks if the black induced subgraph is a m-dominating set and if it is the case, it checks if the graph $G_c(v)$ is biconnected. When these two properties are met, a node can get out of the backbone.

Algorithm 3: Pruning rule

```
Input: $k \in \{1, 2\}$, the connectivity
       $m \in \mathcal{N}^+$, the domination parameter
Output: true if the node can get out of the backbone
        false if it is still required
changeState = false;
if $N^B_{1\cup 2}(v)$ is a m-dominating set of $G(v)$ then
    if $k = 1$ and $G(v)$ is connected then
        changeState = true;
    if $k = 2$ and $G_c(v)$ is biconnected then
        changeState = true;
return changeState;
```

Algorithm 4: Construction rule

```
Input: $k \in \{1, 2\}$, the connectivity
       $m \in \mathcal{N}^+$, the domination parameter
Output: true if the node can get out of the backbone
        false if it is still required
changeState = true;
if $N^B_{1\cup 2}(v)$ is a m-dominating set of $G(v)$ then
    if $k = 1$ and $G(v)$ is connected then
        changeState = false;
    if $k = 2$ and $G_c(v)$ is biconnected then
        changeState = false;
return changeState;
```

3.6.3 Construction rule

The construction rule is the opposite of the pruning rule. A node has to enter the backbone if its 1-hop nodes are not m-dominated or if the graph $G_c(v)$ is not biconnected.

3.6.4 Main algorithm

The main algorithm repeats the same operation during the whole simulation. Depending on the current node color, it selects the rule to be applied and changes the color when it is required.

3.6.5 Proof of correctness

Proof. The domination property (prop 2) creates a m-dominating set. The connectivity property (prop 1) ensure to create a 2-connected set. Then, the resulting backbone is a 2-connected m-dominating set. □

3.6.6 Variation of the Blackbone2 algorithm

Two variations of Blackbone2 are proposed. The first one is designed to create good quality $1, m$-CDS and the second one proposes to enhance the convergence process in order to get a steady solution in less time.

1-connected version.
A simple variation of the algorithm consists in creating 1-connected m-dominating set. The only modification resides in checking the connectivity of the black induced subgraph. For this variation we do not remove the previously defined useless 2-hop nodes. The algorithms are unchanged, only the parameter k has to be set to 1. The connectivity checking is performed by a simple depth-first search based algorithm and can be processed in $\mathcal{O}(|V| + |E|)$.

Algorithm 5: Main algorithm of Blackbone2

```
Input: $k \in \{1, 2\}$, the connectivity
       $m \in \mathcal{N}^+$, the domination parameter
for every time step do
    if $BLACK$ and pruningRule($k, m$) then
        color = WHITE;
    if $WHITE$ and constructionRule($k, m$) then
        color = BLACK;
```

Algorithm 6: Construction rule for fast convergence

```
Input: m, the domination parameter
changeState = true;
if $N^B_{1\cup 2}(v)$ is a m-dominating set of $G(v)$ then
    if $G^B(v)$ is connected then
        if $G_c(v)$ biconnected then
            changeState = false;
        else
            if $G_c(v)$ with v is biconnected then
                changeState = true;
            else
                changeState = false;
return dominating;
```

1-connected fast convergence version.
The basic idea too achieve a faster convergence would be to reduce the possibilities for a node to change its color. One solution we found consists in adding another decision in the construction rule. First, a node v checks if the black nodes are a m-dominating set. Second, the connectivity of the black node subgraph is tested. If these two first conditions are met, then v will check if the graph $G_c(v)$ is biconnected. If not, it checks if $V_c(v) \cup \{v\}$ is biconnected. If not, the node v does not enter the backbone, because it supposes it cannot help achieving biconnectivity. This rule can be found in algorithm 6. Of course, in some graphs, obtaining a biconnected structure will require the cooperation of a group of nodes. This case is not taken into account, that is why this version does not guarantee the biconnectivity.

4. THEORETICAL PERFORMANCES

In this section we briefly discuss the theoretical performances of the Blackbone2 algorithm in terms of time, bandwidth and message complexity.

4.1 Time complexity

Let Δ be the average node degree. In the worst case, $|V(u)| > \Delta^2$ and $|E(u)| > \Delta^2$. Based on this, checking the existence of an articulation point requires $\mathcal{O}(\Delta^2)$. As the domination checking algorithm only requires $\mathcal{O}(\Delta)$, the two rules (pruning and construction) have a complexity of $\mathcal{O}(\Delta^2)$. Finally, Blackbone2 has a time complexity of $\mathcal{O}(\Delta^2)$.

4.2 Bandwidth and message complexity

Except the beacons, that are periodically broadcasted in wireless networks, the *Blackbone*2 algorithm does not require additional messages to compute a solution. For this reason, we say that the message complexity is $\mathcal{O}(1)$. However, if we consider the bandwidth usage, we have to take into account the fact that the size of the beacons increases with the number of 1-hop neighbors. Let Δ be the mean density of the network. Then the bandwidth usage for each node is $\mathcal{O}(\Delta)$.

5. NUMERICAL RESULTS

In this section, we will first briefly discuss the setting of the algorithm parameters. In a second subsection, a performance comparison will be achieved for regular CDS and 2-connected dominating sets. In a last subsection, the performances of the algorithm different heuristics will be compared.

5.1 Parametrization

The simulator used for all the experimentation is OMNeT++ version 3.3p1 with the Mobility Framework version 2.0p3. The first series of tests were used to fine tune the algorithm parameters:

- The beaconing period
- The checking period that updates the state of a node

The speed of the algorithm, i.e. the required time to converge to a steady solution (in static context), depends on these two parameters. In order to compute the best ratio between these two periods, we fixed the beaconing period to 500 ms and we tried different values for the checking period. Based on these first experiments, we noticed that good values for checking state period are multiples of the beaconing period (500, 1000 and 1500). Finally the beaconing period has been set to 500 ms and the checking period to 1000 ms for the remainder of the experimentations.

In order to initialize the simulation all the nodes will start their beaconing process at a time defined by a random number. This randomness also reduces collisions at the beginning of the simulation.

The simulation space is a 100 meters square. The transmission range has been fixed to 25 meters. These experiments have been repeated for different numbers of nodes (20, 30, 40 up to 140), which permits to deal with a wide range of average node degree or density [2, 22]. In table 1 a summary of the mean densities by number of nodes in the network is shown.

Table 1: Average density per number of node in the network

Number of nodes	Average density
20	2,87
30	4.52
40	6.18
50	7.73
60	6.37
70	10.93
80	12.47
90	14.06
100	15.6
120	18.68
140	21.76

The OMNeT++ default random number generator (Mersenne Twister RNG by M. Matsumoto and T. Nishimura) has been used to distribute the nodes in the simulation space. The seed of the simulation is equal to the number of its run.

5.2 Performance comparison

We first compare results for biconnected backbones and then for connected dominating sets.

5.2.1 Biconnected backbone

The main goal of this article is to propose a fully decentralized 2-connected m-dominating set algorithm. As far as we know, no equivalent algorithm has been proposed. Therefore, our results cannot be fairly compared to the solutions we found in the literature because of important differences. First, a lot of algorithms are composed of distinct phases, such as DDA and LDA, which is not realistic for ad hoc networks where a synchronization of the whole networks is hardly achievable. Moreover LDA is based on CDS-BD-D which requires a node to be the root of the network. Second, some algorithms may have access to some piece of information that is very difficult to gather in a decentralized way, such as network size and node density for the two first approaches in [4], or even constant maximal node degree for LDA.

The only similar algorithm existing in the literature is the deterministic approach of Dai *et al.*, that proposes to create k, m-CDS with $k = m$. This algorithm uses the k-coverage condition [4] that checks if there are k independent paths for every pair of neighbors. This computation is done using a variation of the Edmonds and Karp maximum flow algorithm. We have compared our results with this approach, even if there is one main difference: our algorithm does not require to have a synchronized network and can be easily used in a dynamic context. We also compare our results with the previous version of Blackbone [9] in its 2-connected version, Blackbone1, that can only produce 1-dominating backbones.

As we can see in Figure 3, Blackbone1 gives good solutions only when the network density is not too high. This problem is overcome with Blackbone2 which always provides better solutions than the coverage condition of Dai *et al.* for $k = m = 2$ without any density restriction.

Fair comparisons can be made by comparing:

- BB1 $m = 1$ with BB2 $m = 1$ for $2, 1$-CDS
- BB2 $m = 2$ with DaiWu for $2, 2$-CDS

In both cases we can observe that Blackbone2 gives better results for all the simulated networks. Moreover, Blackbone2 can be computed in $\mathcal{O}(\Delta^2)$ when the first version of Blackbone required $\mathcal{O}(\Delta^3 \log(\Delta))$. The k-coverge condition can be computed in $\mathcal{O}(k\Delta^4)$. This high computation cost may be an important factor when dealing with mobile environments, where information may be incomplete and available computation time is short. All computation times can be found in table 2.

5.2.2 Connected backbone

The variation of the Blackbone2 algorithm does not guarantee biconnectivity that is the reason why we compared its results to Blackbone1 set to create $1, 1$-CDS. We have also compared these results to the well-known algorithm of Wu *et*

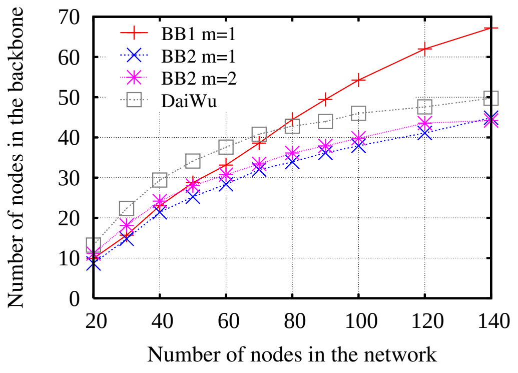


Figure 3: Quality of solution for biconnected backbone

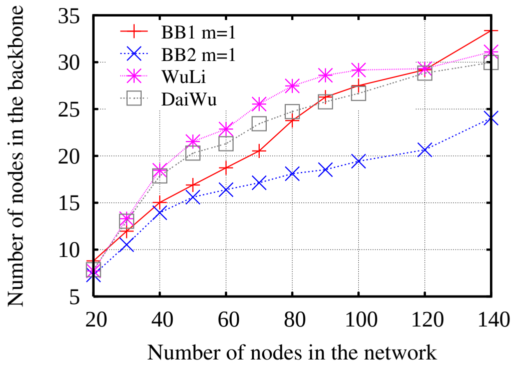


Figure 4: Quality of solution for connected backbone

al. in its most efficient version [16]. The pruning rule of this version considers every subset of nodes that could cover the set of marked nodes for a given node u. This algorithm will be named WuLi(*). Moreover, the k-coverage condition [4] with $k = 1$, more recently proposed by Kim *et al.* has been implemented for comparison purpose.

In Figure 4 we can observe that Blackbone1 gives the best results until the network is composed of 80 nodes. After 80 nodes and up to 100 nodes, DaiWu and Blackbone2 give the best results and are almost equal. However, in very dense networks Blackbone2 generates the best solutions.

All computation time can be found in table 2. Blackbone1 require much more time than Blackbone2 but can compute better solutions for low and moderatly dense networks. DaiWu algorithm gives good results but suffers from a very high computation cost.

In addition to the computation cost, another major improvement of the variation of Blackbone2 is the stability of the structure before convergence. In Figure 5 we can see that Blackbone2 greatly reduces the number of required state changes to achieve the final solution. This is particularly suitable in mobile environment in which topology changes induce recurrent backbone maintenance.

5.3 Impact of the domination parameter

In this subsection we present the impact of the domination parameter on the results of the Blackbone2 algorithm and its variation for 1-connected m-dominating sets.

Table 2: Computation time

Algorithm	Computation time
Blackbone1	$\mathcal{O}(\Delta^3 \log(\Delta))$
Blackbone2	$\mathcal{O}(\Delta^2)$
k Coverage Condition	$\mathcal{O}(k\Delta^4)$
Coverage Condition	$\mathcal{O}(\Delta^3)$
WuLi(*)	$\mathcal{O}(\Delta^2)$

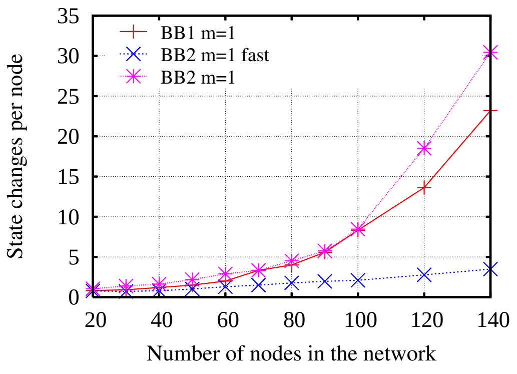


Figure 5: Stability of the backbone before convergence

In Figure 6, we can observe that the Blackbone2 algorithm gives good results even when the domination increases. Moreover, the impact of the domination parameter is negligible in very dense networks for $m \in \{1, 2, 3\}$. Indeed, when there are 140 nodes in the simulation space, the results are almost the same for the three first values of m. This is due to the increasing density, a single backbone node will cover more nodes when the density is higher.

In Figure 7 we can observe that the fast version gives worse results than the first variation, which is more quality-oriented. In high density networks the fast version outperforms the results of Blackbone1 [9]. However, if quality of the solution is the main criterion, the first variation of Blackbone2 obtains the best results for every network size.

We have seen that the first variation creates smaller backbones, however it requires much more time than the second version. In Figure 8, the stability results of the fast version outperform those of Blackbone1 and the first variation of Blackbone2.

6. CONCLUSION

In this paper we have investigated the problem of constructing 2-connected m-dominating set in ad hoc networks. As such we have proposed Blackbone2, a deterministic time-and-message efficient algorithm composed of simple rules based on node marking. This algorithm has been designed to adapt itself to mobile environments by relying on two computationally efficient rules. Moreover the algorithm improves existing results proposed in the literature for any network density and has a very low theoretical complexity. This algorithm also shows very encouraging results for 1-connected m-dominating set with the two variations of the algorithm, for both quality of the solutions and convergence time.

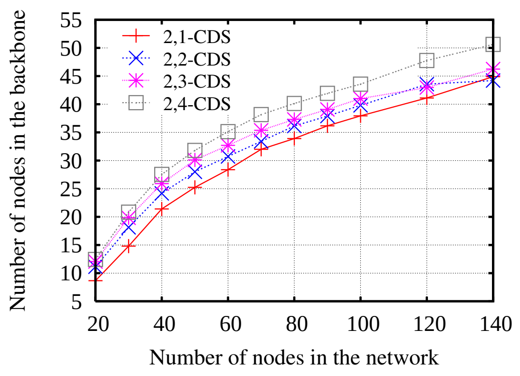


Figure 6: Impact of the domination parameter on quality of solution

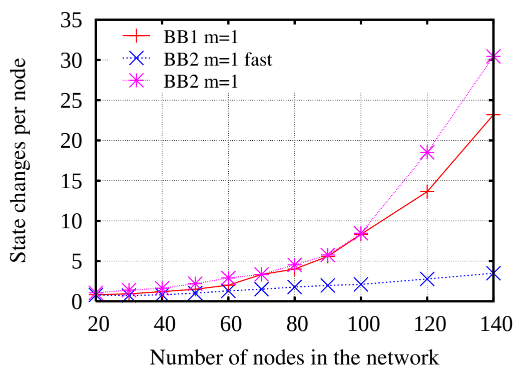


Figure 8: Impact of the domination parameter on the stability

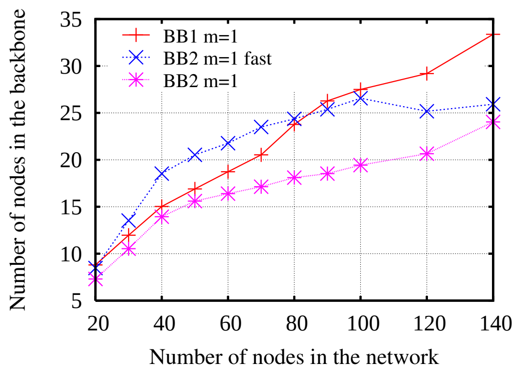


Figure 7: Impact of the domination parameter on the quality

Acknowledgement

The authors gratefully acknowledge the financial support from Fonds National de la Recherche, Grand-Duché de Luxembourg and Université Paul Verlaine - Metz, France, for J. Schleich throughout the period of this research.

7. REFERENCES

[1] K. M. Alzoubi, P.-J. Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. *Proc. IEEE Hawaii Intl. Conf. System Sciences*, 2002.

[2] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli. Localized protocols for ad hoc clustering and backbone formation: A performance comparison. *IEEE Transactions on Parallel and Distributed Systems*, 17(4):292–306, 4 2006.

[3] F. Dai and J. Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Trans. Parallel and Distributed Systems*, 15(10):908–920, 10 2004.

[4] F. Dai and J. Wu. On constructing k-connected k-dominating set in wireless network. *IEEE International Parallel and Distributed Processing Symposium*, 2005.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[6] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(3):388–404, 2000.

[7] D. Kim, Y. Wu, Y. Li, F. Zou, and D.-Z. Du. Constructing energy efficient connected dominating sets with bounded diameters in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 2007.

[8] Y. Li, S. Zhu, M. T. Thai, and D.-Z. Du. Localized construction of connected dominatinf set in wireless networks. *IEEE INFOCOM*, 2002.

[9] J. Schleich, P. Bouvry, and L. T. H. An. Decentralized fault-tolerant connected dominating set algorithm for mobile ad hoc networks. In *International Conference on Wireless Networks*, 2009.

[10] W. Shang, F. Yao, P. Wan, and X. Hu. Algorithms for minimum m-connected k-dominating set problem. *COCOA 2007, LNCS 4616*, pages 182–190, 2007.

[11] M. T. Thai, N. Zhang, R. Tiwari, and X. Xu. On approximation algorithms of k-connected m-dominating set in disk graphs. *Accepted by Theoretical Computer Science*, 2007.

[12] P.-J. Wan, K. M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. *NFS International Workshop on Theoretical Aspects of Wireless Ad hoc, Sensor and Peer-to-Peer Networks*, 6 2004.

[13] F. Wang, M. T. Thai, and D.-Z. Du. 2-connected virtual backbone in wireless network. *Accepted by IEEE Transactions on Wireless Communication*, 2007.

[14] Y. Wang, W. Wang, and W. yang Li. Efficient distributed low-cost backbone formation for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(7):681–693, 7 2006.

[15] J. Wu. An enhanced approach to determine a small forward node set based on multi-point relay. *Proc. 58th IEEE Semiann. Vehicular Technology Conf*, 4:2774–2777, 10 2003.

[16] J. Wu and H. Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. *Proc. Third ACM Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm.*, Dial M 1999:7–17, 8 1999.

[17] Y. Wu and Y. Li. Construction algorithms for k-connected m-dominating sets in wireless sensor networks. *Mobihoc'08*, pages 83–90, 5 2008.

[18] Y. Wu, F. Wang, M. T. Thai, and Y. Li. Constructing k-connected m-dominating sets in wireless sensor networks. *Military Communication Conference*, 10 2007.