Experimental comparison of algorithms for interference control in ad-hoc wireless networks

Stavros Athanassopoulos and Christos Kaklamanis and Evi Papaioannou and Constantinos Tsantilas
Research Academic Computer Technology Institute &
Department of Computer Engineering and Informatics
University of Patras, 26500 Rio, Greece
E-mail: {athanaso, kakl, papaioan, tsantilk}@ceid.upatras.gr

Abstract—Interference is an issue of outstanding importance for efficient communication in ad-hoc wireless networks. Interference arises due to overlapping transmission power levels of nodes and causes message collisions that require energy-consuming re-transmissions. In this work we study algorithms for interference control in ad-hoc wireless networks in the Euclidean space that aim at constructing low-interference network topologies with particular properties preserving network connectivity.

We survey relative research work on such algorithms and provide implementations for four such algorithms comparing them on random geometric instances of the problem in the Euclidean space in average-case networks. Our experiments show that our implementations obtain in practice improved performance compared to corresponding theoretical bounds. Our findings confirm that sparse topologies do not automatically imply low interference and, furthermore, imply that a unified model for decreasing simultaneously both edge- and node-interference remains hard to define.

Keywords: Energy consumption, interference, ad-hoc wireless networks, algorithms, communication efficiency, experimental evaluation.

I. INTRODUCTION

Wireless networks have received significant attention during the recent years. Especially, ad-hoc wireless networks (even composed of sensors) emerged due to their potential applications in battlefield, emergency disaster relief, etc. [16]. Unlike traditional wired networks or cellular wireless networks, no wired backbone infrastructure is installed for ad hoc wireless networks which consist of autonomous mobile nodes that are equipped with an omnidirectional antenna which is responsible for sending and receiving signals, thus having the ability to communicate over wireless links.

A node in these networks is equipped with an omnidirectional antenna which is responsible for sending and receiving signals. Communication is established by assigning to each station a transmitting power. In the most common power attenuation model, the signal power falls as \(1/r^\alpha\), where \(r\) is the distance from the transmitter and \(\alpha\) is a constant which depends on the wireless environment (typical values of \(\alpha\) are between 1 and 6). So, a transmitter \(s\) can successfully send a signal to a receiver \(t\) if \(P(s) \geq \gamma d(s,t)^\alpha\), where \(P(s)\) is the power of the signal transmitted, \(d(s,t)\) is the Euclidean distance between the transmitter and the receiver, and \(\gamma\) is the receiver power threshold for signal detection which is usually normalized to 1. In this case, we say that node \(s\) establishes a direct link to node \(t\). So, communication from a node \(s\) to another node \(t\) may be established either directly if the two nodes are close enough and \(s\) uses adequate transmitting power, or by using intermediate nodes (see Fig. 1).

An ad-hoc wireless network is usually modelled as a complete directed graph \(G = (V,E)\), with a non-negative edge cost function \(c : E \rightarrow R^+\). Intuitively, \(V\) is the set of stations or nodes, the edges in \(E\) correspond to potential direct links, and the function \(c\) denotes the minimum energy required for establishing a direct link between any possible transmitter-receiver pair. Usually, the edge cost function is symmetric (i.e., \(c(u,v) = c(v,u)\)). An important special case, which usually reflects the real-world situation, henceforth called geometric case, is when nodes of \(G\) are points in a Euclidean space and the cost of an edge \((u,v)\) is defined as the Euclidean distance between \(u\) and \(v\) raised to a fixed power \(\alpha\), i.e. \(c(u,v) = d(u,v)^\alpha\). Asymmetric edge cost functions can be used to model medium abnormalities or batteries with different energy levels [9]. A directed edge \((u,v)\) can only be used by \(u\) to reach \(v\) but not vice versa. Yet \(v\) could reach \(u\) by some other distinct path. An undirected graph is called connected if there is a path between any two nodes in the graph. If there is a path between any two nodes in a directed graph, the graph is called strongly connected. The transmission range of network nodes can be idealized as a circle or disk; nodes correspond to disk centers while transmission power of nodes correspond to disk radii. Unit Disk Graphs (UDGs) [3] have also been

Fig. 1. Communication among 3 nodes in ad-hoc wireless networks.
used for modelling wireless networks. In a Unit Disk Graph \( G = (V, E) \), there is an edge \((u, v) \in E\) iff the Euclidean distance between \( u \) and \( v \) is at most 1. This equivalently means that we assume all nodes to have the same limited transmission ranges.

Interference in ad-hoc wireless networks arises when messages fail to be delivered to destination-nodes either because they are sent simultaneously over the same communication link or because they simultaneously arrive at their destination-nodes. Interference may result in collisions and consequently data retransmissions which degrade network throughput and increase delays. Minimizing interference between network nodes has been an important objective. In previous works (e.g., [12], [3], [2]) interference in ad-hoc wireless networks has been naturally closely related to energy consumption problems: reducing energy consumption results from minimizing interference between network nodes and hence reducing message collisions and required retransmissions. Interference has also been studied from a topology control perspective: network nodes reduce their transmission power and drop long-range connections in a coordinated way so that network connectivity is preserved. Thus, topology control can be considered a trade-off between energy conservation and (i) interference reduction and (ii) connectivity. Recent research results address topology control and routing. Topology control aims at constructing sparse network topologies that guarantee particular properties, like short paths, low energy consumption, planarity, etc, preserving network connectivity. Routing algorithms then try to actually route information along optimal paths that are theoretically present due to topology control. The main reason for constructing sparse networks is to reduce interference. Such an implicit notion of interference can however lead to topology control algorithms that fail to reduce interference since message transmission can affect nodes even if they are not direct neighbors of the sending node in the resulting topology graph [1]. Besides demonstrating this weakness of implicit interference models, [1] introduces an explicit definition of interference, based on the number of nodes potentially disturbed by communication over a link. Another line of research [5] has considered topology control algorithms that explicitly minimize interference instead of node degree.

Several interference models have been proposed in the literature; see for example [10], [13], [1], [11], [12]. Here, we focus on two interference models: edge-interference model and node-interference model. Their main difference in the definition of interference can be stated as follows: in the former model, interference is defined based on how many nodes are affected by communication over a particular edge, while in the latter, interference is defined with a focus on the receiving node, i.e., interference is defined by explicitly counting the number of nodes potentially disturbing the reception of a message.

Edge Interference [1]: the interference of a link \( uv \) is defined as the number of nodes covered by two disks centered at \( u \) and \( v \) with radius \(|uv|\). Let \( D(u, r) \) denote the disk centered at node \( u \) with radius \( r \). Specifically, they defined the coverage of a link \( uv \) as \( cov(uv) = \{w|w\text{ is covered by } D(u, |uv|) \text{ or } D(v, |uv|)\} \). Thus, edge-interference, \( cov(uv) \), represents the set of all nodes that could be affected by node \( u \) or by node \( v \) when they communicate with each other using exactly the minimum power needed to reach each other. Alternatively, edge-interference can be considered as the number of nodes located in the overlapping area of two nodes.

The edge level interference is extended to a graph interference measure as the maximum coverage occurring in a graph: \( I(G) = \max_{e \in E} cov(e) \). This model is chosen since whenever a link \( uv \) is used for a send-receive all nodes at distance less than \(|uv|\) from node \( u \) or from node \( v \) will be affected. Thus, the network is represented by a geometric undirected weighted graph, \( G = (V, E, W) \), with vertices corresponding to wireless nodes, and edges representing communication links. The weight of each link \( uv \) is its interference \( IC(uv) \).

Node Interference [14]: the interference value of a node \( v \) is defined as \( I(v) = \|\{u|v \in D(u, r_u)\}\| \) where \( D(u, r) \) stands for the transmission circle with node \( u \) in its center and radius \( r_u \). The interference of a Graph \( G(V, E) \) is defined as \( I(G) = \max_{v \in V} I(v) \). Thus, the interference of a node is the number of transmission circles by which the node is covered. The interference of the whole network is defined as the maximum of all interference values in the graph. When directed edges (or arcs) are assumed, the node-interference model is selected for the problem of finding a sink tree of minimum interference for a given sensor network. More precisely, the Minimum Interference Sink Tree (MIST) problem is defined as the problem of finding a sink tree for a given node set with minimal interference. The assumption of bidirectional edges [14] makes the node-interference model to be also suitable for general ad-hoc networks.

In this work, we have studied four algorithms for interference control in ad-hoc wireless networks. EMST finds a minimum spanning tree of a set of points in the plane (using the Kruskal’s algorithm [6]) where the weight of the edge between each pair of points is the distance between those two points. GLIT is a simple greedy algorithm [1] that requires nodes to be spanned by a minimum interference tree. RNG is based on the construction of Relative Neighbouring Graphs [15] where and edge exists between two vertices if and only if they are relative neighbours, i.e., if they are at least as close to each other as they are to any other vertex. NCC Nearest Component Connector (NCC) algorithm [5], which builds valid topologies with interference \( O(\log n) \) in any ad-hoc wireless network by connecting components to their nearest neighbors; a component can be a single node or a group of previously connected nodes. The algorithm works in rounds and finally creates a sink tree. We have implemented several NCC heuristics which result from using different formulas for increasing node transmission range when necessary for preserving network connectivity. We have experimentally compared these algorithms in terms of interference efficiency of the solutions obtained on random instances on the Euclidean plane in terms of the induced edge and node interference. In
our study, optimizing running times of our implementations has not received significant attention, since interference and execution speed has been our focus. The rest of the paper is structured as follows. We devote Section 2 to a more detailed description of algorithms EMST, GLIT, RNG and NCC. Implementation details and experimental findings are presented in Section 3.

II. ALGORITHM DESCRIPTION

In this section, we present the algorithms considered in this paper, namely EMST, RNG, GLIT and BNCC. BNCC is a new algorithm we implemented based on algorithm NCC which has been proved to outperform in terms of node-interference in worst case instances. Our motivation has been to investigate whether an algorithm that minimizes node-interference can remain efficient in terms of edge-interference.

A. Greedy Low Interference Tree (GLIT)

GLIT [1] is a simple greedy algorithm that constructs an interference optimal spanning tree, i.e., when nodes are required to be spanned by a tree GLIT computes the minimum interference tree. GLIT can be considered as a variation of the Kruskal’s algorithm for constructing Minimum Spanning Trees (MST) in edge-weighted connected graphs where edge weights correspond to their interference.

GLIT implies the construction of a tree; however, this holds only for undirected edges; if directed edges are used, GLIT constructs an interference-optimal strongly-connected topology. GLIT starts with a fully connected graph and continually removes edges with high coverage as long as the desired property is maintained. As proved in [1], the tree constructed by GLIT is 1-interfering according to interference measure \( I_{\text{avg}} \) that consider interference on nodes. This is because nodes are static and thus the number of nodes covered by a disk can be computed once, regardless of edges added to the topology in next steps.

Different versions of GLIT can be obtained in the same way as Prim’s and Kruskal’s algorithms for minimum spanning tree; fringe edges could be added to an existing tree (like Prim’s algorithm), or the edge that minimizes the greedy criterion globally could be added (like Kruskal’s algorithm). Furthermore, if directed edges are used, \( NNG \subseteq GLIT \), NNG denotes the Nearest Neighbor Graph where each node has a directed edge to its nearest neighbor.

B. Relative Neighbourhood Graph (RNG) and Euclidean Minimal Spanning Tree (EMST)

\( MST \) (Minimal Spanning Tree) is a well-known graph proposed in the literature for the following problem: given a set of points on the plane find a structure/graph among the points in the form of edges connecting a subset of the pairs of points. In \( MST \), \( P \) edges are formed by joining pairs of points to form a tree that spans \( P \) so that the sum of the Euclidean edge lengths is less than the sum for any other spanning tree. The Euclidean minimum spanning tree or EMST is a minimum spanning tree of a set of points in the plane, where the weight of the edge between each pair of points is the distance between those two points. Kruskal’s algorithm [6] is a greedy algorithm for finding a minimum spanning tree for a connected weighted graph by finding a subset of edges that forms a tree and includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component).

The Relative Neighbourhood Graph (RNG) of a finite planar set [15] is based on the notion of “relatively close” neighbours initially defined by Lankford [8]. Given a set \( P = \{p_1, p_2, \ldots, p_n\} \) of \( n \) distinct points on the plane two points \( p_i \) and \( p_j \) are “relatively close” if \( d(p_i, p_j) < \max\{d(p_i, p_k), d(p_j, p_k)\}, \forall k = 1, \ldots, n, k \neq i, j \). The definition states that two points are relative neighbours if they are at least as close to each other as they are to any other point also implying that that three points lying equidistant from each other are not relative neighbours. The RNG is obtained by adding an edge between points \( p_i \) and \( p_j \) for all \( i, j = 1, \ldots, n \), with \( i \neq j \) if and only if \( p_i \) and \( p_j \) are relative neighbours. As proved in [15], \( MST \subset RNG \subset DT \) and therefore the number of edges \( N_e \) in the RNG of \( n \) points on the plane is bounded by \( n - 1 \leq N_e \leq 3n - 6 \), and is thus \( O(n) \).

C. Nearest Component Connector (NCC)

Nearest Component Connector (NCC) algorithm, as described in [5], builds valid topologies with interference \( O(\log n) \) in ad-hoc sensor networks, but can be generalized to apply to ad-hoc wireless networks by eliminating the directional-edges requirement.

The main idea that NCC implements is to connect components to their nearest neighbors in several rounds and finally construct a sink tree. Components can be single nodes or groups of previously connected nodes. Initially, each node including predefined global sink is a component. Two or more components connected in one round form a single component for subsequent rounds containing a node - called local sink - to which all other component members have a directed path. When a new arc (directed edge) is added, it connects the local sink of a component to the nearest node not in that component. Since all nodes are assumed to have unit maximum transmission range, the current sink of a component may not be able to connect to any external node; in such cases the node which is closest to the current sink and can reach any node outside the component becomes the new sink; finding such nodes is always feasible since only connectable networks are considered. Similarly, if the root of the final NCC tree is not the global sink, it is moved to the global sink.

As proved in [5], the NCC algorithm constructs a valid sink tree topology for a given (sensor) network consisting of \( n \) nodes and has asymptotically optimal worst-case behavior meaning that it produces interference not greater than \( O(\log n) \) for all possible node arrangements.
III. IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

We have implemented algorithms NCC, GLIT, RNG and EMST in networks where nodes are arbitrarily distributed in a square field with side length 100 units. We have experimentally compared algorithms EMST, GLIT, RNG and NCC in terms of edge interference. We have also provided an experimental study of algorithm NCC and several NCC-based heuristics.

Nodes are assumed to be randomly and uniformly distributed in the Euclidean space and simulations have been made for network instances of 10, 20, 30, ..., 500 nodes. Simulations for each such instance include 100 repetitions for obtaining an average case analysis. Simulations have been implemented in C++ using the Eclipse IDE platform for C/C++ Developers and MinGW (compiler and debugger). Below, we give brief implementation details. Initially, we define to total number of network nodes (10 - 500). Node coordinates are stored via class Node which contains coordinates x and y as well as the node id. All network nodes are stored in the NodeSet list of that class. Coordinates are assigned to nodes through GiveCoordinates(). Values are generated using ReturnRandom(), which returns a random integer values between 0 and 100. Then, the maximum node transmission range (Max_Transmit) is defined through function DisconnectednessCheck(). A basic property off all resulting graphs is that they maintain full connectivity. Therefore the maximum transmission range can not be less than the maximum minimum distance between every two nodes since in such a case there will be non-reachable nodes. However, this value still cannot guarantee network connectivity for networks of less than 100 nodes. Based on experimental observations, doubling the above value is sufficient for guaranteeing graph connectivity in our simulations. Edges are defined through class Edge which contains source and destination nodes, their distance and the induced interference. The valid edge set is then defined as a list of this class.

Next, we provide a short description of basic components of our implementation.

- CreateEdgeSet(): constructs the valid edge set; it checks distances between all nodes and creates an edge between two of them when they are at distance less or equal to the maximum transmission range. Each edge is inserted in the edge list so that the resulting valid edge set is sorted according to increasing Euclidian node distance. The final edge set is defined through class EmstEdge and is stored in a list of this class. Class EmstEdge contains the same elements as class Edge and an additional element: component. Two edges contain the same component as long as there is a path between them. Similar data structures are used for the rest of the algorithms (i.e., GlitEdgeSet, Glit_Set, RngEdge, Rng_Set).
- CreateEmstSet(): constructs a list of this class and uses Kruskal's algorithm for the MS generation (edge weights correspond to their Euclidian distance) and sequentially checks all sorted edges of list EdgeSet. Similar functions are used for the rest of the algorithms (i.e., CreateGlitSet(), Glit_Set, CreateRngSet()).
- nccnode: structure that contains information for node coordinates, nodes to which arcs are established, local sink property, component, nearest node outside the node component and node interference. All this info is used by algorithm NCC and is stored in dynamic arrays.
- GlobalSink(): randomly selects a network node as the global sink for algorithm NCC. LocalSinkArray() assigns values 1 or 0 to nodes depending on whether they are or not local sinks; initially, all nodes are assumed to be local sinks. in are local sinks.
- CreateNccSet(): constructs the final graph. Until a single sink remains, it locates the local sink and then FindNearest() finds the nearest reachable node outside the component. If no such node exists, FindNearestCandidateSink() finds such a node by gradually increasing nodes transmission range while MoveSink() is used for altering local sinks assignments.

- ReturnEmstInterference(), AverageEmstInterference(): calculate graph edge interference and average graph edge interference, respectively, for graphs generated by algorithm EMST.
- ReturnGlitInterference(), AverageGlitInterference(): calculate graph edge interference and average graph edge interference, respectively, for graphs generated by algorithm GLIT.
- ReturnRngInterference(), AverageRngInterference(): calculate graph edge interference and average graph edge interference, respectively, for graphs generated by algorithm RNG.
- FindBNccInterference(), AverageNccInterference(): calculate graph edge interference and average graph edge interference, respectively, for graphs generated by algorithm NCC.
- ReturnNccInterference(), AverageNccInterference(): calculate graph node interference and average graph node interference, respectively, for graphs generated by algorithm NCC.

We have studied algorithm NCC in terms of node interference. Our experimental results for several NCC heuristics are in accordance with the theoretical upper bound; furthermore, they show that in practice in average networks node interference never reaches the upper bound ($O(\log n)$, where $n$ is the number of nodes) but rather remains always lower than 6.

In terms of edge interference, we have evaluated algorithms GLIT, RNG, EMST and several NCC heuristics. These heuristics result from using different formulas for increasing node transmission range when necessary for preserving network connectivity. Algorithms GLIT, RNG and EMST are designed in order to construct graphs of low edge interference; however, including NCC (which outperforms in terms of node interference) is interesting in order to experimentally evaluate the possibility of existence of a unified model for measuring interference. NCC was initially designed for maintaining node interference low. This was obtained by having nodes in as few
Fig. 2. NCC implementation vs the theoretical upper bound.

Fig. 3. Comparison of algorithms EMST, GLIT, RNG and NCC in terms of edge-interference.

Fig. 4. Comparison of NCC and NCC-based heuristics in terms of edge-interference.

as possible overlapping ranges (i.e., low node interference); if these ranges are constructed/selected appropriately, then it is worth-investigating whether such a construction also decreases the number of nodes falling into the ranges of any two neighbouring nodes (i.e., low edge interference). Our results show that such a model is rather hard to deploy (certainly not with algorithm NCC or NCC-based heuristics). We initially observe (see Fig. 2) that algorithm NCC as well as several NCC-based heuristics practically perform well also for average-case instances obtaining better interference than the theoretically expected bounds. In practice in average networks NCC causes node interference that never reaches the theoretical upper bound of $O(\log n)$, where $n$ is the number of nodes, but rather remains always lower than 6. Both algorithms GLIT and EMST are based on the Kruskal’s algorithm but use a different definition for the edge weights; GLIT uses edge weights as interference while EMST uses the Euclidean length of an edge as its interference. We observe that edges with small Euclidean length do not necessarily imply/correspond edges with lower interference; EMST uses edges of smallest possible Euclidean length, but GLIT obtains lower interference. Furthermore, they show that even in arbitrary average-case networks, finding shortest edges cannot guarantee optimal interference. On the other hand, comparing to the rest of the algorithms we use, the performance of algorithm EMST is not extremely bad; except for GLIT, EMST constructs graphs of interference lower than those of algorithms RNG and NCC, as depicted in Figure 4. So our findings show that decreasing the edges Euclidean length can have a positive impact on the network interference but it certainly cannot be considered as a dominating factor for interference elimination. Figure 4 shows the surprisingly high edge interference raised by algorithm NCC and several heuristics. It is interesting that algorithm NCC while having proved to be (worst-case) optimal for the directed-edge node-interference model, results in very poor performance in terms of edge-interference thus implying that it remains hard to find an optimal algorithm - at least in practice - in terms of both edge- and node-interference.

ACKNOWLEDGMENT

This work was partially supported by the European Union under IST FET Integrated Project 015964 AEOLUS and by the General Secretariat for Research and Technology of the Greek Ministry of Development under programme PENED 2003.

REFERENCES


