Pervasive and Mobile Computing 4 (2008) 597–615

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Fast track article

Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks

Furuzan Atay Onat [a,*], Ivan Stojmenovic [b,c], Halim Yanikomeroglu [a]

[a] Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada

[b] Electronic, Electrical and Computer Engineering, The University of Birmingham, Birmingham, United Kingdom

[c] SITE, University of Ottawa, Ottawa, Ontario, Canada

ARTICLE INFO

Article history:
Received 6 August 2007
Received in revised form 26 April 2008
Accepted 29 April 2008
Available online 9 May 2008

Keywords:
Random unit disk graphs
Graph generation
Ad hoc
Sensor
Actuator networks

ABSTRACT

In this paper, we consider generation of graphs that represent specific scenarios that appear in wireless ad hoc, actuator, sensor and Internet networks. Most simulation studies for these networks use connected random unit disk graphs generated by placing nodes randomly and independently from each other. However, in real life usually networks are created in a cooperative manner; certain restrictions are imposed during the placement of a new node in order to improve network connectivity and functionality. This article is an initial study on how constrained connected random unit graphs (C-CRUG) can be generated by fast algorithms and what kind of desirable characteristics can be achieved compared to completely random graphs, especially for sparse node densities.

1. Introduction

Wireless multihop networks use cooperative communication where nodes forward each others' packets. If the nodes are mobile, the network topology changes over time. Wireless multihop networks include wireless ad hoc networks, wireless Internet access and sensor-actuator networks. Wireless ad hoc networks do not rely on any infrastructure. Typically nodes communicate with each other without any access to the outside world. In wireless Internet networks, individual users access the Internet through the access points, which are connected to the wireline backbone. Wireless sensor networks are large scale networks formed by small size sensor devices that rely on battery power. Sensor networks can be used for a variety of applications. Due to limited transmit power of sensors, data is sent in a cooperative fashion, where nodes forward each others' packets. Sensor-Actuator Network (SAN) is a new paradigm in which the network not only senses the events but also acts on them through actuator nodes. The actuators are more capable nodes such as robots or humans equipped with vehicles. Their tasks might include relocating sensors, collecting sensor readings and warning a center about suspicious activity within the sensing field. Actuators are expected to be mobile and thus the network topology changes over time.

In most studies on multihop wireless networks, nodes that are close to each other are assumed to have a direct radio link. The maximum distance to which a node can transmit directly is called the node's *transmission range*. It is usually assumed that all nodes have a common transmission range. Unit disk random graphs are often used to represent this topology. A unit disk graph is determined by node positions and a fixed communication range, R, for all nodes. Any two nodes whose distance does not exceed R are connected by a bidirectional edge. In simulation studies, it is desirable to generate connected random unit disk graphs (CRUG) to evaluate the performance of different network protocols. These graphs are normally generated

* Corresponding address: Department of Systems and Computer Engineering, Carleton University, Eng., 1125 Colonel By Drive, K1S 5B6 Ottawa, ON, Canada.

E-mail addresses: furuzan@sce.carleton.ca (F.A. Onat), ivan@site.uottawa.ca (I. Stojmenovic), halim@sce.carleton.ca (H. Yanikomeroglu).

1574-1192/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2008.04.011

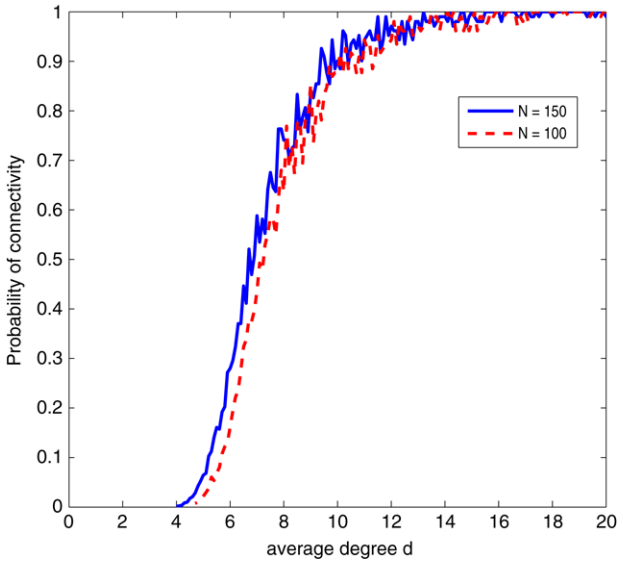

Fig. 1. Illustration of critical behavior, nodes randomly placed in a 100×100 region.

by placing nodes randomly and independently from each other, and testing connectivity at the end. Sparse random unit disk graphs have high probability of being partitioned, which increases the generation time.

Although it might be appropriate to assume independent positions in some cases, such as sensors dropped from air or thrown manually, in many practical scenarios networks are generated iteratively by choosing the location of a node with respect to the other nodes in the network. For instance, in wireless ad hoc networks and wireless LANs, users usually tend to choose their own locations depending on the locations of other users. Consider a conference with attendants forming a network with their laptops. The attendants would choose to sit where they can connect to the network and avoid overpopulated areas to have reasonable throughput. In sensor networks, for energy efficiency different sensors are turned on and off without jeopardizing sensing coverage and network connectivity. Although the sensor positions are expected to be random, the positions of the nodes that are active are correlated with each other due to coverage and connectivity constraints. In sensor-actuator networks, actuators create their own communication graph to enhance their coordination, data communication and ability to serve the network better.

All these considerations impose certain restrictions on node locations. The graphs where each node satisfies a set of constraints based on its own location and the location of other nodes are not necessarily drawn from the same distribution as completely random unit disk graphs. Hence, we use a different name for this family (or families): we refer to them as the *constrained connected random unit graphs (C-CRUG)*. In this paper, we discuss how to generate C-CRUGs corresponding to different scenarios by fast algorithms with desirable properties.

Connectivity is the first requirement for a network to function properly (e.g., to enable routing between any two nodes). A network is said to be *connected* if all of its nodes can reach each other either directly or in a multihop fashion. Most network protocols are designed assuming that the network is connected. Usually, performance of different network protocols are tested or compared to each other on a large number of connected random unit disk graphs. Thus, efficient algorithms that generate such graphs with given parameters are needed. Consider generating connected random unit disk graphs with given number of nodes N, and average node degree d under independent node distribution. The two coordinates of each node are selected from a uniform distribution to place the nodes uniformly inside a square. In Fig. 1, we plot the empirical probability that a generated graph is connected as a function of average degree d for $N = 100$ and $N = 150$ nodes. An abrupt change in connectivity is observed between degrees $d = 4$ and $d = 10$. For average degrees smaller than 7, most graphs generated by this method are disconnected.

Unlike the standard generation method, in which nodes are placed randomly and independently (and connectivity is verified at the end), inspired by a method described in [1], our methods place nodes one by one. To decide where to place the next node, we use the properties of the nodes already placed such as coverage and approximate degree, which is calculated using an approximate transmission radius.

Main contributions of this paper are as follows:

- We propose a novel way of generating connected random unit graphs for simulation studies considering the scenario to be studied.
- We propose different constraints on node positions that represent practical scenarios in wireless ad hoc, sensor, actuator, and Internet networks.
- We design several algorithms to generate C-CRUGs based on these constraints: Coverage Algorithms (CA1 and CA2), Maximum Degree Proximity Algorithm (MAX-DPA), Weighted Proximity Algorithm (WPA), Eligible Proximity Algorithm

(EPA), Minimum Degree Proximity Algorithm (MIN-DPA), and Clustered Minimum Degree Proximity Algorithm (C-MIN-DPA). We show that, in addition to better reflecting the dynamics governing network formation in real life, these algorithms can be more efficient for generating sparse connected networks than any other method in the literature. We propose MAX-DPA, MIN-DPA, WPA, and EPA for simulating wireless ad hoc and actuator networks, CA1 and CA2 for simulating sensor networks, and C-MIN-DPA for simulating wireless Internet networks.
- We study the nature of the graphs generated by some of these algorithms.

The rest of this paper is organized as follows. A summary of the related work is given in Section 2. We first present the general properties of the new generation methods in Section 3. The algorithms for mobile ad hoc networks and actuator networks are given in Section 3.1. Sections 3.2 and 3.3 are devoted to the algorithms for sensor networks and wireless Internet access. The conclusions of the paper are summarized in Section 4.

2. Related work

Connectivity properties of random unit disk graphs have been studied in percolation theory. In general, percolation theory deals with clustering in large random systems [2]. The systems where nodes are distributed according to Poisson point process and each node is assigned a random region independent of other nodes are referred to as Boolean models. An infinite random unit disk graph can be represented as a Boolean process with a given node density λ (nodes per unit disk area) and a deterministic disk radius α. Let us denote this process as $\mathcal{B}(\lambda, \alpha)$. Nodes whose disks intersect with each other are connected. All nodes belonging to the same connected component form a *cluster*. Thus, if $\alpha = R/2$, a cluster represents the nodes that can reach each other.

The main result of percolation theory states that $\mathcal{B}(\lambda, \alpha)$ has two distinct phases. For a fixed $\alpha > 0$:

- If λ is smaller than a critical value $\lambda^*(\alpha)$, the system has no unbounded cluster with probability one. This phase is called *subcritical phase*.
- If λ is greater than $\lambda^*(\alpha)$, the system forms an unbounded cluster with positive probability. This phase is called *supercritical phase*.

Due to the scaling properties of $\mathcal{B}(\lambda, \alpha)$, similar regimes can be identified for a fixed λ and a critical radius $\alpha^*(\lambda)$. This result implies that, if the nodes are placed according to a uniform distribution independently from each other, it is not possible to generate a large connected random unit disk graphs with an arbitrary (λ, α) pair. The existence of an infinitely large component is necessary but not sufficient for connectivity. Nevertheless, on a finite plane, existence of very large clusters is a good indication of connectivity.

In an infinitely large network with node density λ and transmission range R, the expected value of the degree of a node (the number of its neighbors in the unit disk graph) is equal to $d = \lambda\pi R^2$. Networks are determined by their expected node degrees, and can be scaled to fit any area. In fact, due to the well-known scaling property of the Boolean model, two processes $\mathcal{B}(\lambda_1, \alpha_1)$ and $\mathcal{B}(\lambda_2, \alpha_2)$ are equivalent if $\lambda_1/\lambda_2 = \alpha_2^2/\alpha_1^2$ [2]. Based on this argument, we choose the average node degree d as the main parameter of our problem, where the average is taken over all nodes in each graph.

Connectivity properties of wireless ad hoc networks have been studied in [3–7]. In these works, critical behavior of connectivity in ad hoc networks has been shown in different settings. In [5], Gupta and Kumar study the critical transmission range for N nodes randomly distributed in a unit disk. They show that if transmission radius R is such that $\pi R^2 N = (\log(N) + c(N))/N$, the network is connected with probability one if and only if $c(N) \to \infty$ as $N \to \infty$. This means that $\log(N)$ is the critical average density for graph connectivity. In [8], Xue and Gupta also show that the graphs where each node is connected to k nearest neighbors go through a phase transition and the critical number of neighbors is $\theta(N)$. Asymptotic results of [5] are extended to finite networks in [7]. Reference [4] studies the critical transmission range both in stationary and mobile networks.

More strict conditions on connectivity might be required for fault-tolerance, handling mobility and multipath routing. A k-connected network is defined as a network that remains connected after removing $k - 1$ arbitrary nodes. Critical transmission range for k-connectivity has been studied in different contexts in [9–11]. In [12] Dubashi et al. studied a relevant problem: selecting k neighbors of each node from the set of all neighbors in connected random unit disk graph, so that overall connectivity remains high. They show that nodes may use only constant number of links without loosing overall connectivity as the network size grows.

In sensor networks, for redundancy, sensor nodes are expected to be deployed at high densities. In this case, the network connectivity is usually guaranteed and not all of the nodes are needed for monitoring the area. However, in order to extend the lifetime of the sensors and the network, only some of the sensors are turned on at any given time. Various protocols have been proposed in the literature to determine which nodes must be turned on to assure area coverage and connectivity. In [13], a simple rule has been given to check whether the coverage area of a particular node has already been covered by its neighbors. The same result also has been proven in [14].

All the algorithms considered in the present article, known or new, go through the following steps: first, a candidate graph is generated by choosing node positions in a given region (picked as an $\ell \times \ell$ square). Then, the candidate graph is tested for connectivity and this procedure is repeated until a connected graph is generated. Although random unit disk graphs have been commonly used in the literature to represent ad hoc network scenarios, there are very few methods proposed to

generate such graphs. In fact, we did not find any article in the literature that specifically deals with connected random unit disk graph generation algorithms. We will now review three methods that were identified in the literature. The well-known and widely used method in the literature takes transmission radius as the independent variable, generates each of N nodes at random inside a fixed region, and then tests network connectivity in the end via Dijskstra's shortest path algorithm [15]. In many experiments, the graphs are sufficiently dense so that the connectivity is virtually guaranteed and is actually not tested.

Our main interest, however, is to use average node degree (the average number of neighbors of each node) as the independent variable due to the scaling properties of unit graphs mentioned earlier. An algorithm to generate connected random unit disk graph with given average degree d is described in [16]. This algorithm will be called here the *standard algorithm*. In the standard algorithm, first, all the nodes are placed uniformly and independently. All $N(N-1)/2$ potential edges in the network are sorted by their lengths in ascending order. The length of the $(Nd/2)$th shortest edge is chosen as the transmission range R. The first $Nd/2$ edges in the list remain in the graph. Other edges are eliminated, making sure that the graph generated is a unit disk graph and has average degree equal to d. Then, this candidate graph is checked for connectivity. If it is not connected, the procedure is repeated until a connected graph is generated. The standard algorithm generates the same kind of unit disk graphs as the algorithm normally used in the literature, but places the emphasis on the average degree as the independent variable rather than the transmission radius. This allows us to evaluate the performance for sparse, medium degree and dense networks with clear and precise values of independent variables.

We define $T_c(N, d)$ as the average time spent to generate one connected unit disk graph with N nodes and average degree d. $T_c(N, d)$ is given by:

$$\begin{aligned} T_c(N,d) &= \big(T_g(N,d) + T_t(N)\big)\, E\,[K(N,d)] \\ &= \big(T_g(N,d) + T_t(N)\big)\, \frac{1}{P_c(N,d)}, \end{aligned} \tag{1}$$

where T_g is the time to generate a candidate graph and T_t is the time to test the connectivity of a graph. P_c is the probability that a generated candidate graph is connected. K is the random variable denoting the number of graphs generated until the first connected one, which has geometric distribution with parameter P_c.

The standard algorithm generates graphs in a fully random fashion. It is the most unbiased way of generating connected random unit graphs in the sense that it chooses any random connected unit disk graph among all possible random connected unit disk graphs having desired (N, d) with equal probability. Moreover, the computation to generate a candidate graph is minimal. That is, T_g has the smallest possible value. However, as seen in Fig. 1, due to the critical behavior of connectivity, as d gets smaller, P_c decreases abruptly and T_c increases. Other algorithms we present here aim to increase P_c by placing the nodes more intelligently, still randomly but in a more restricted fashion. Although this increases T_g, in sparse graphs it can be compensated by a decrease in $E[K(N, d)]$ in Eq. (1).

The remaining algorithms considered and described in this paper place the nodes one by one depending on the positions of the nodes placed previously. We call the placement procedure of each new node as a *round* and the set of nodes placed by the end of *round* k as S_k. The first such algorithm is proposed in [1], which will be called here the *Proximity Algorithm* (PA).

The idea of the PA is to place the nodes close to each other in order to increase P_c. An approximate radius r is used to decide if two given nodes will be neighbors in the graph generated. Approximate radius is determined as follows. Ignoring finite size effects, expected node degree is the number of remaining nodes $N-1$ times the probability that any node will be placed within the node's transmission area. The latter probability is equal to the ratio of transmission area to the total area. Then, the expected degree for a given r is $\pi r^2(N-1)/A$ where $A = \ell^2$ is the area of the square region. The value of r is chosen such that the expected node degree is equal to the desired average degree d and is given by:

$$r = \sqrt{dA/((N-1)\pi)}. \tag{2}$$

In PA, the first node is randomly generated. That is, its x and y coordinates are chosen uniformly in $[0, \ell)$. Then, the remaining $N-1$ nodes are placed one by one. In *round* k, first a position is generated at random. If this position is within distance r of at least one of the nodes in S_{k-1}, the position is accepted, the node is placed to this position and added to S_k. Otherwise, a new random position is generated until an acceptable position is obtained. This algorithm is proposed in [1] to generate sparse connected unit disk graphs. We observed that this method results in a high standard deviation of node degrees, because nodes selected early in the process will get more neighbors than nodes selected later on. Another consequence is that nodes are grouped in a portion of network around the initial few nodes, leaving a large portion of the area empty. Such properties of the PA, in fact, cause it to be slower than the standard method. Motivated by this insight, we suggest new methods, which are described in the next section.

3. New generation methods for constrained random networks

We propose two new classes of algorithms: acceptance/rejection based algorithms and center node based algorithms. In center node based algorithms for each new node Z, a center node C is chosen among the already placed nodes. The new node Z is placed around this center node. We introduce four center node based algorithms: Minimum Degree Proximity Algorithm (MIN-DPA), Clustered Minimum Degree Proximity Algorithm (C-MIN-DPA), Weighted Proximity Algorithm (WPA), and

Eligible Proximity Algorithm (EPA). These algorithms differ in the way the center node is chosen. In acceptance/rejection based algorithms, in each round a random candidate place is selected for each node. Then, this position is either accepted or rejected depending on some constraints. We propose three acceptance/rejection based algorithms: Maximum Degree Proximity Algorithm (MAX-DPA), Coverage Algorithm 1 (CA1), and Coverage Algorithm 2 (CA2). In order to better represent specific applications in practice, we consider different constraints on node positions:

Proximity constraint: To increase the probability of achieving a connected graph, it might be better to place each new node in the vicinity of already existing nodes, preserving a minimum distance. A node is said to satisfy the proximity constraint if it is closer than the approximate radius r to at least one of the existing nodes and no closer than a mutual distance d_0 to any of the existing nodes. Note that satisfying the proximity constraint for all the nodes assures connectivity for transmission radius r. The actual transmission radius of the graph, determined as the length of the $(Nd/2)$th shortest edge, can be different from r. Thus, the connectivity of the graph generated should be checked and it should be rejected if it is not connected.

Maximum degree constraint: To balance load and increase throughput and the probability of being connected, it might be preferable to avoid nodes with high degrees. A position is said to satisfy the maximum degree constraint if it does not increase the degree of the existing nodes beyond a given maximum value.

Coverage constraint: This constraint applies to sensor networks. A node position is said to have satisfied the coverage constraint if placing a node to this position extends the coverage area of the network sufficiently.

3.1. C-CRUGs for wireless ad hoc networks and actuator networks

To generate C-CRUGs for wireless ad hoc networks and actuator networks, we propose three center node based algorithms (MIN-DPA, EPA, and WPA) and an acceptance/rejection based algorithm (MAX-DPA).

3.1.1. Generating C-CRUGs for wireless ad hoc networks and actuator networks

Minimum degree proximity algorithm (MIN-DPA): The MIN-DPA aims to distribute node degrees more uniformly while maintaining connectivity. The idea is to place each new node around the node that has the smallest number of neighbors.

The first node is randomly placed. At round k, the unit disk graph defined by all the nodes in S_{k-1} and the approximate transmission radius r (given in Eq. (2)) is considered. Node degrees in this graph are called *approximate degrees*. Then, the minimum of the approximate degrees L is found. One of the nodes with degree equal to L is to be selected as the center node of this round. If there are more than one such node, one of them is selected randomly as follows: each minimum degree node is assigned weight of 1 if its transmission range is completely within the square region. If its transmission region intersects with the boundary, a weight of $2/3$ is assigned. The purpose of this distinction is to prevent nodes from concentrating around the boundary. The remaining nodes have weights zero. One node is selected at random as the center node according to these weights. That is, the normalized weights are used as the probability mass function to choose the center node.

We denote the position of the center node by $C(x_c, y_c)$ and position of the new node by $Z(x_z, y_z)$. The next node is placed at a uniformly chosen random position within the transmission position of the center node as follows: coordinates of Z relative to the center node C are represented in polar coordinates as (a, θ) where a is the distance $|CZ|$ and θ is the positive angle between x-axis and CZ. Angle θ is uniformly distributed between $[0, 2\pi)$. Cumulative distribution function of a can be determined using basic probability:

$$F_a(\alpha) = Pr\{a \leq \alpha\} = \pi\alpha^2/\pi r^2. \tag{3}$$

From Eq. (3), we see that a^2 is uniformly distributed in $(0, r^2)$. Therefore, to pick an a value with this distribution, we can generate a random number v uniformly distributed in $(0, r^2)$ and choose $a = \sqrt{v}$. After selecting a random (a, θ) according to the distributions given above, coordinates of Z are determined as:

$$x_z = x_c + a\cos\theta, \qquad y_z = y_c + a\sin\theta. \tag{4}$$

If the center node is close to the boundary, point Z might be outside the square region. In this case, a new random position (x_z, y_z) is generated. Then, the new position is passed through the proximity test and rejected if it is closer than d_0 to any of the nodes. This procedure continues until all the nodes are placed. The pseudocode of the MIN-DPA is given in Algorithm-1.

Weighted proximity algorithm (WPA): Unlike MIN-DPA, WPA allows all the nodes placed be considered in center node selection. It assigns weights to all existing nodes based on their approximate degrees and their *desired degrees*. Desired degree of a node at a given position is defined as the expected number of neighbors it will have if all nodes are distributed uniformly.

Let us denote current approximate degree (currently considered number of neighbors) of node i as d_i and its desired degree as d_i'. At *round* k, $2 \leq k \leq N$, the weights to select the center node are assigned as follows:

Algorithm 1 MIN-DPA(N, d, ℓ, d_0)

$r = \sqrt{d\ell^2/((N-1)\pi)}$
connected = FALSE
while (not connected) **do**
  $x = rand(0, \ell)$, $y = rand(0, \ell)$
  $p_1 = (x, y)$
  $S_1 = \{node\_1\}$
  for $k = 2, \ldots, N$ rounds **do**
    calculate approximate degrees of all nodes in S_{k-1}
    L = min of approximate degrees
    for each node m in S_{k-1} that has degree L **do**
      if $D_m \subset D$ **then**
        $weight(m) = 1$
      else
        $weight(m) = 2/3$
      end if
    end for
    $C(x_c, y_c)$ = randomized_center_select(weight)
    accepted = FALSE
    while not accepted **do**
      $v = rand(0, r^2)$, $a = \sqrt{v}$, $\theta = rand(0, 2\pi)$
      $x_z = x_c + a\cos\theta$, $y_z = y_c + a\sin\theta$
      if $((x_z, y_z) \in D)$ and $((x_z, y_z)$ passes proximity test with all nodes in $S_{k-1})$ **then**
        $p_k = (x, y)$
        $S_k = S_{k-1} \cup \{node\_k\}$
        accepted = TRUE
      end if
    end while
  end for
  Calculate edge lengths $l_{ij} = |p_i, p_j|$
  Sort l_{ij} $i, j \in \{1, 2, \ldots, N\}$,
  Select $(Nd/2)$ shortest
  Form candidate graph G_c with $(Nd/2)$ edges
  Run Dijkstra for G_c
  if all costs finite **then**
    connected = TRUE
  end if
end while

– Desired degrees d_i' of all nodes in S_{k-1} are calculated. To calculate d_i', the area of the intersection of the node's transmission region and the $\ell \times \ell$ square region, shown as D_i in Fig. 2, is found. Then, desired degree of *node i* is calculated as:

$$d_i' = (N-1)\frac{D_i}{\ell^2}. \tag{5}$$

A simplified version for calculating D_i is possible by approximating the transmission area D_i as $(2/3)\pi r^2$ when it is smaller than πr^2.

– We use a weight function that assigns greater weight to the nodes with degrees smaller than their expected degrees but still considers the rest of the nodes. Weight of *node i* is assigned as follows:

$$w_i = \begin{cases} d_i' - d_i, & \text{if } d_i < d_i' \\ d_i'/(d_i + 1), & \text{if } d_i \geq d_i'. \end{cases} \tag{6}$$

A center node is chosen according to these weights and the next node is placed around the center node as described in MIN-DPA.

Eligible proximity algorithm (EPA): In EPA, the center node is chosen among the nodes whose approximate degree is smaller than d. If the minimum of approximated node degrees L is less than d, all nodes either with degree less than d and with transmission region completely within the square, or with degree less than $2d/3$ and with transmission region intersecting the square boundary are *eligible* to receive more neighbors. Weights of these eligible nodes are assigned as 1 and others are assigned as zero. If L is larger than d, then the weights are decided according to the weight function of the WPA.

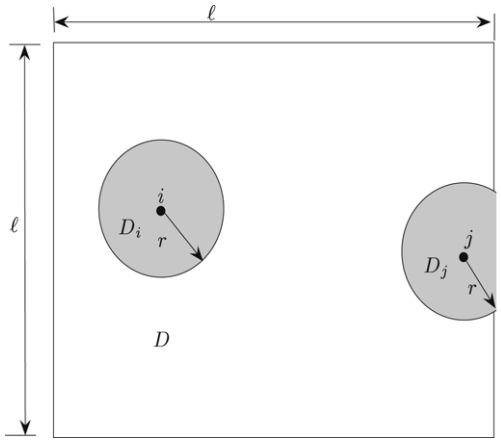

Fig. 2. Illustration of the weight function for WPA.

The algorithm has the following exception to guarantee the continuity of this procedure. Let the degrees of all the nodes be greater than or equal to d except for one node whose transmission region intersects with the boundary. If the degree of this node is greater than $2d/3$, there will be no eligible nodes. In this case, again, the weights are assigned using the weight function of the WPA.

Maximum degree proximity algorithm (MAX-DPA): The algorithms MIN-DPA, WPA and EPA try to distribute the nodes more uniformly by selecting a low degree node as the center node and placing the new node around this center node. However, the placed node might effect not only the center node but also other nodes in the proximity. Based on this observation, we propose MAX-DPA which imposes a maximum degree constraint for each node.

At *round* $k, 2 \leq k \leq N$, first a random position is generated. To be accepted, the position is required to satisfy the proximity constraint and the maximum degree constraint. To check the latter, the approximate degrees (i.e., currently considered degrees) of node k and all nodes in S_{k-1} are calculated assuming that node k were placed to this new position. If none of these degrees is greater than or equal to the maximum degree allowed $d_{\max}$, the position is accepted. Otherwise, a new position is generated. Algorithm-2 describes the MAX-DPA in more detail.

3.1.2. Performance of the algorithms for wireless ad hoc and actuator networks

To compare the algorithms described in this article, we use Monte Carlo simulations. That is, many independent random graphs are generated and various metrics measured are averaged over these graphs. Fig. 3 shows the average time it takes to generate 40 connected unit graphs with $N = 100$ nodes and with degrees $d = 5, 8, 15, 20$. For MAX-DPA, we choose $d_{\max} = d + 3$. Nodes are placed in a 100×100 area. The minimum distance allowed between the nodes (d_0) is assumed to be 1. Time is measured in seconds and plotted on a logarithmic scale due to large range of values. The fraction of connected graphs among all candidate graphs generated by different algorithms is shown in Fig. 4, which can be used to estimate P_c.

For dense graphs (at $d = 15$ and $d = 20$), all algorithms perform similarly (within 15% of each other). As seen in Fig. 4, at these average degrees, most of the graphs generated by all algorithms are connected, i.e., $P_c \approx 1$. At $d = 8$, PA and WPA are 2.5 and 5.5 times slower than the standard algorithm, respectively. However, the EPA is as fast as the standard algorithm. The MIN-DPA and the MAX-DPA outperform rest of the algorithms with a generation time approximately 25% less than the standard algorithm. These comparisons are also valid at $d = 5$ and the differences are more pronounced; PA and WPA are much slower than the standard algorithm. The EPA, MIN-DPA, and MAX-DPA are 1.7, 4.0, and 4.6 times faster than the standard algorithm, respectively.

Although the purpose of the PA is to increase P_c, as seen in Fig. 4, it actually decreases P_c significantly. In PA nodes pile up around the initially placed nodes, which causes non-uniformity in the degree of the nodes; nodes in the high density regions have higher node degrees compared to the rest. Due to the average degree constraint, a limited number of edges are placed in the graph (precisely $Nd/2$ edges). In high density area, nodes are closer and many edges are placed there unnecessarily, having left very few edges to cover other nodes in sparse areas. Consequently, non-uniform degree distribution decreases P_c and the PA is much slower than the standard algorithm.

Among center node based algorithms (MIN-DPA, WPA, EPA), WPA fails to fix PA and performs even slower than PA. On the other hand, given the good performance of EPA and MIN-DPA, we conclude that the center node approach is promising. The performance, however, highly depends on the way the center node is selected. The MIN-DPA is the fastest algorithm among the algorithms in this class and we will study only MIN-DPA as a representative of its class.

For MAX-DPA, we selected a specific $d_{\max}$ value ($d_{\max} = d + 3$), which is not necessarily optimal. Hence, we will check next if there is any room for further improvement.

Algorithm 2 MAX_DPA(N, d, ℓ, $d_{\max}$, d_0)

$r = \sqrt{d\ell^2/((N-1)\pi)}$
connected = FALSE
while (not connected) **do**
  $x = rand(0, \ell)$, $y = rand(0, \ell)$
  $p_1 = (x, y)$
  $S_1 = \{node\_1\}$
  for $k = 2, \ldots, N$ rounds **do**
    placed = FALSE
    while (not placed) **do**
      $x = rand(0, \ell)$, $y = rand(0, \ell)$
      if (x, y) passes the proximity test with all nodes in S_{k-1} **then**
        if (x, y) and all nodes in S_{k-1} pass max degree test **then**
          $p_k = (x, y)$
          $S_k = S_{k-1} \cup \{node\_k\}$
        end if
      end if
    end while
  end for
  Calculate edge lengths $l_{ij} = |p_i, p_j|$
  Sort l_{ij} $i, j \in \{1, 2, \ldots, N\}$,
  Select $(Nd/2)$ shortest
  Form candidate graph G_c with $(Nd/2)$ edges
  Run Dijkstra for G_c
  if All costs finite **then**
    connected = TRUE
  end if
end while

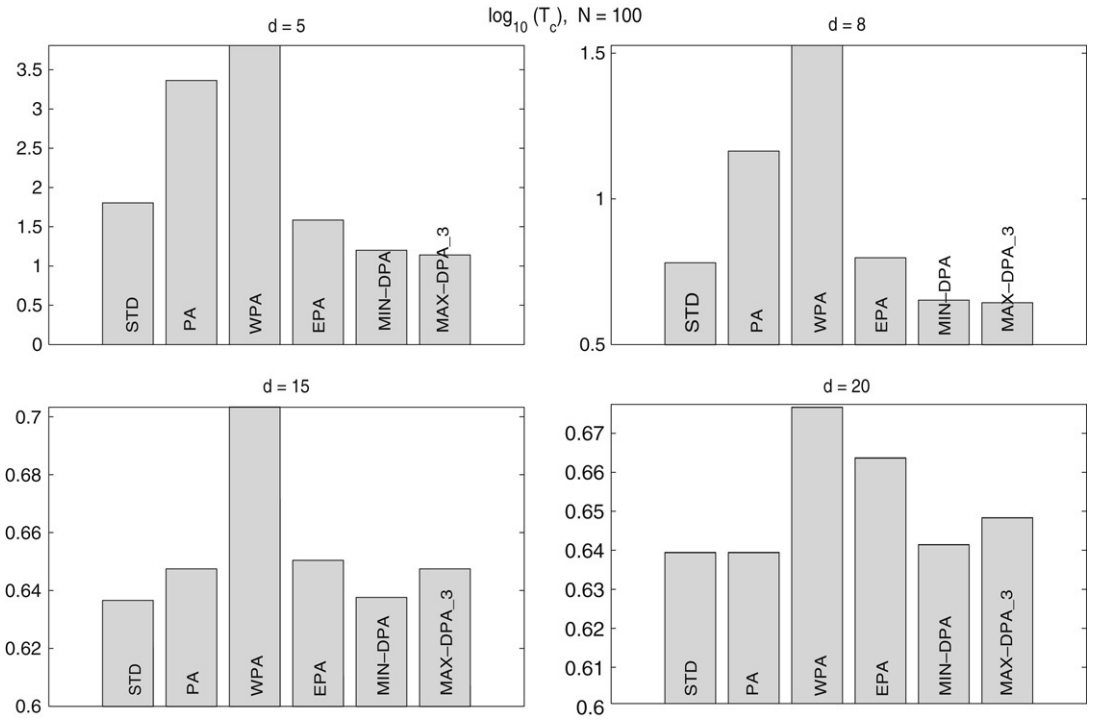

Fig. 3. Time it takes to generate 40 connected graphs with $N = 100$ (in logarithmic scale).

Effect of $d_{\max}$ in MAX-DPA: We run MAX-DPA with five different values of $d_{\max}$ for each d, namely with $d_{\max} = d + n$, where $n \in \{2, 3, 4, 5, 6\}$. The algorithm using $d_{\max} = d + n$ is denoted by MAX-DPA$_n$. Fig. 5 shows the generation times for MAX-DPA. The fraction of connected graphs among all candidate graphs generated is given in Fig. 6.

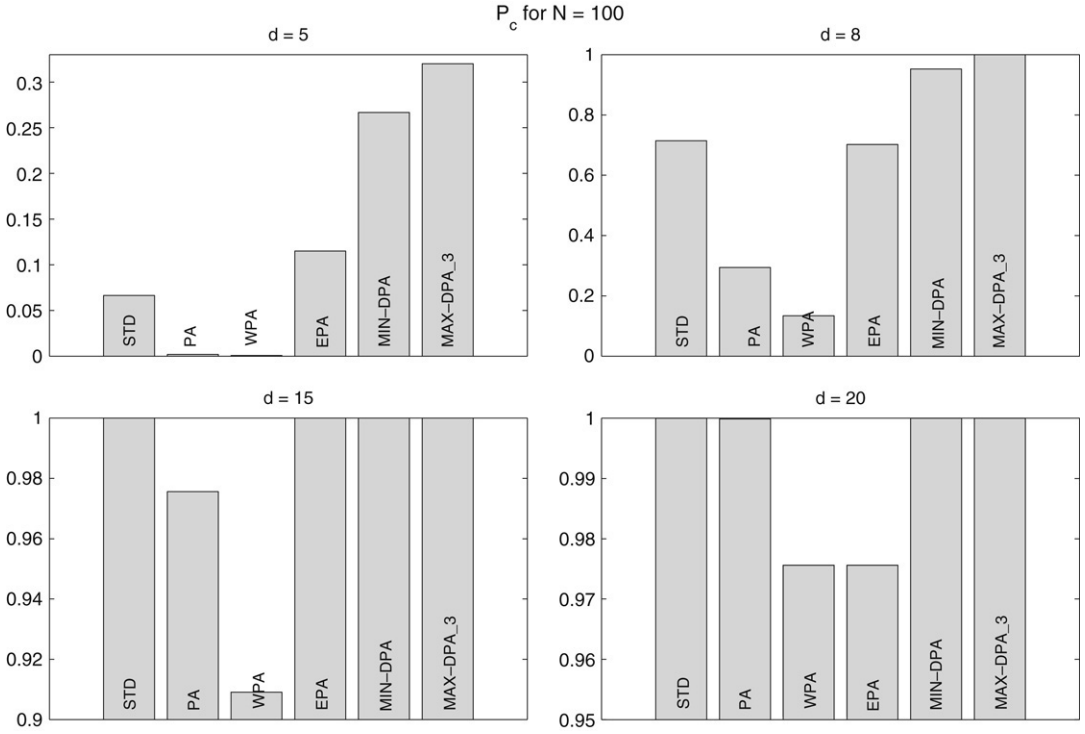

Fig. 4. Fraction of connected graphs among all generated graphs with $N = 100$. When $d = 5$ very small fraction of graphs are connected for PA and WPA, 1.9×10^{-3} and 7.0×10^{-4}, respectively.

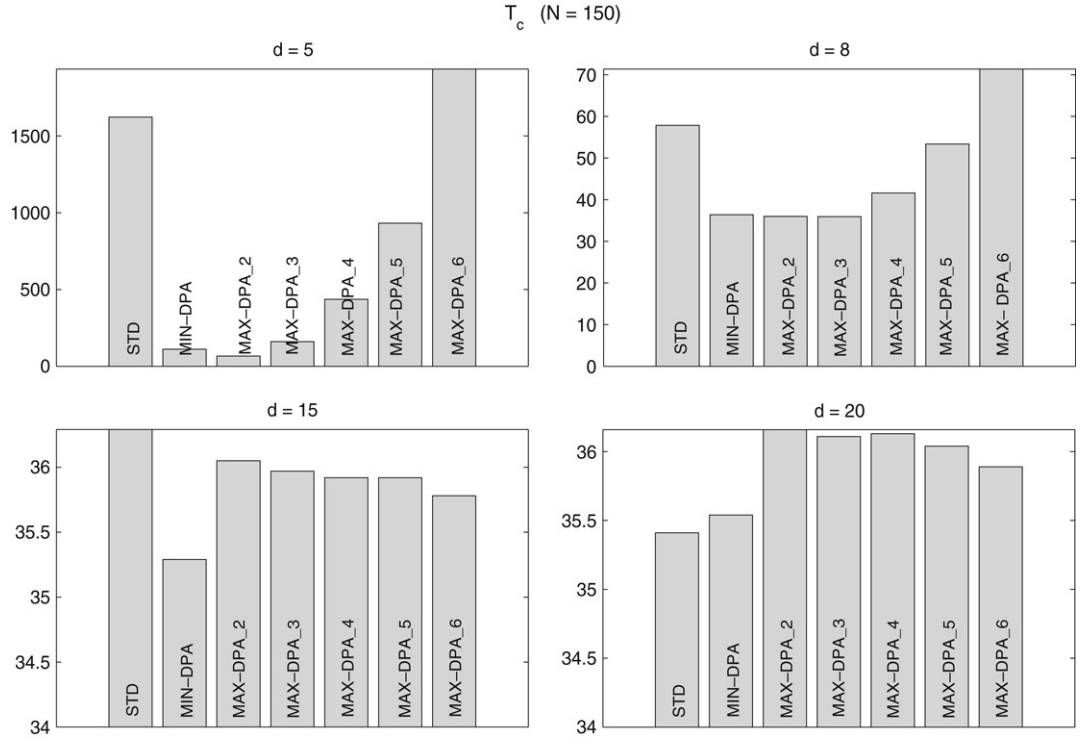

Fig. 5. Time it takes to generate 100 connected graphs with $N = 150$.

As expected, T_c is not sensitive to $d_{\max}$ at dense graphs ($d = 15, 20$). At $d = 5, 8$, the generation time of MAX-DPA improves as $d_{\max}$ decreases and is the lowest at $d_{\max} = d+2$. Fig. 6 reveals that this improvement is due to an increase in P_c.

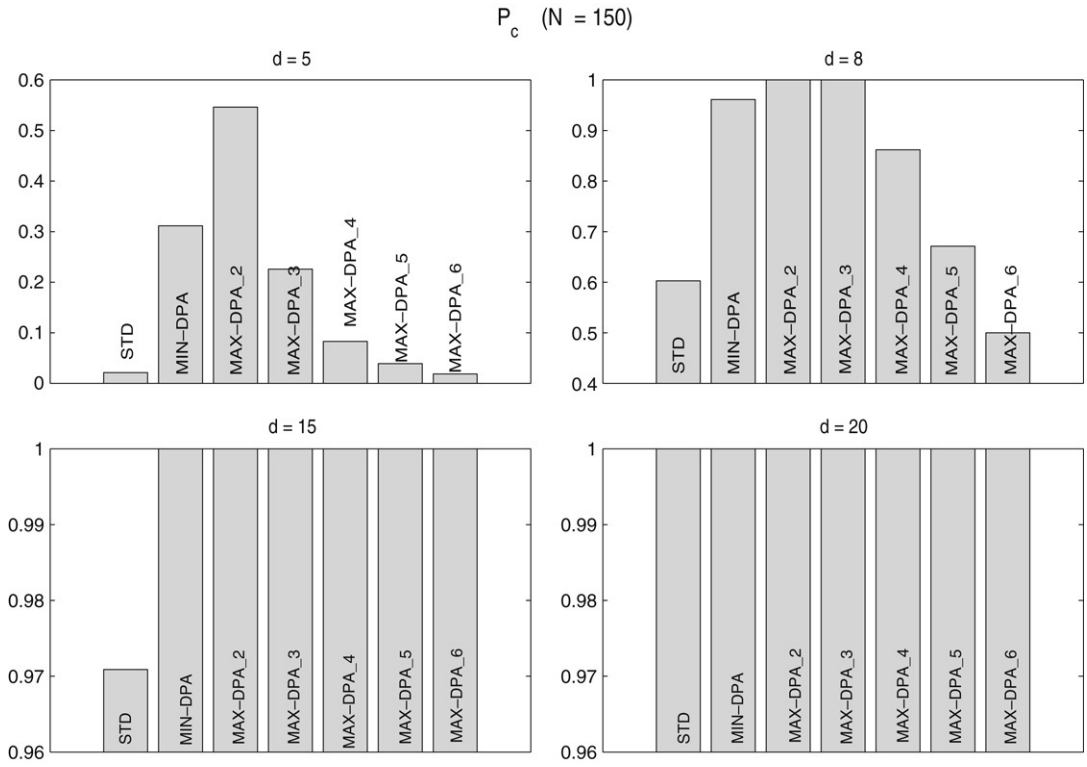

Fig. 6. Fraction of connected graphs among all generated graphs with $N = 150$.

Although it is not included in the figure, during the simulations we observed that this trend is reversed as $d_{\max}$ is decreased further. When $d_{\max}$ is smaller than $d + 2$, T_c increases abruptly. This is mainly caused by an increase in the average time to generate a graph (T_g). In the extreme case of $d_{\max} = d$, all nodes have the same degree and the resulting graph is a regular random graph with degree d. In the other extreme case, where $d_{\max}$ is much larger than d, during the placement of the nodes maximum degree test is always satisfied and the only constraint is the proximity test. Thus, when $d_{\max} \gg d$, MAX-DPA converges to PA, which has very poor performance. For instance, MAX-DPA with $d_{\max} = d + 6$ is slower than the standard algorithm.

In Fig. 6 it is also observed that at $d = 5$ a small fraction ($\approx$2%) of the candidate graphs generated by the standard algorithm are connected while MIN-DPA and MAX-DPA can generate connected graphs up to 30% and 60% of the time, respectively. Therefore, the performance difference is even more noticeable at this degree.

3.1.3. Properties of the graphs generated by different algorithms

We note that the graphs generated by these new methods are not necessarily drawn from the same distribution as those generated by the standard algorithm. That is, some graph types obtained by the standard algorithm cannot be generated or are generated with very low probability in the new algorithms. Thus, we devote this section to understanding the nature of graphs generated by different algorithms.

Fig. 14 shows candidate graphs, which are not necessarily connected, generated by different algorithms for $N = 100$ and $d = 5$. A closer look at these graphs reveals an interesting property: Let d_r denote the average degree of the candidate graphs calculated based on the approximate radius r given in Eq. (2). For the standard algorithm $d_r < 5$ while for the rest of algorithms $d_r > 5$. However, when the connectivity is checked based on the actual transmission radius corresponding to $d = 5$, as described in Section 3, the candidate graphs by the standard algorithm have a much lower probability of being connected (P_c) than some of the C-CRUG algorithms. In other words, r overestimates the actual radius for these C-CRUGs and underestimates the actual radius for the CRUGs generated by the standard algorithm.[1] These findings pose new questions about possible improvements to the C-CRUG algorithms by choosing the approximate r more carefully. Fig. 15 illustrates the final (connected) graphs generated by the same algorithms. By visual inspection of these figures it is observed that, if $d_{\max}$ is chosen appropriately the C-CRUGs generated by MAX-DPA are closest to CRUGs in terms of the distribution of node positions within the area. As $d_{\max}$ of MAX-DPA is increased its candidate graphs become more partitioned and connected

[1] For [STD, PA, MIN-DPA, MAX-DPA$_2$, MAX-DPA$_4$, MAX-DPA$_8$], d_r = [4.48, 10.48, 7.99, 5.30, 6.38, 8.12] and P_c = [0.053, 0.002, 0.233, 0.697, 0.177, 0.020].

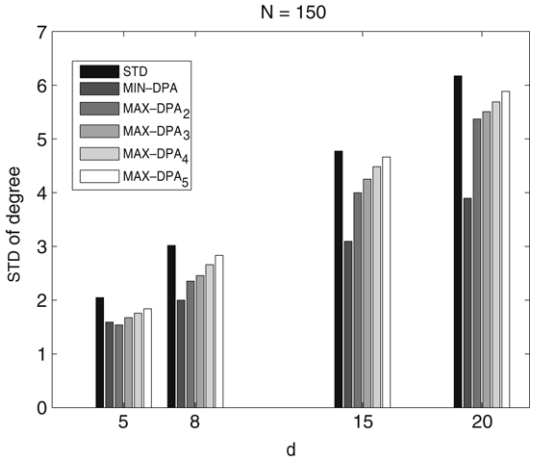

Fig. 7. Standard deviation of degrees, $N = 150$.

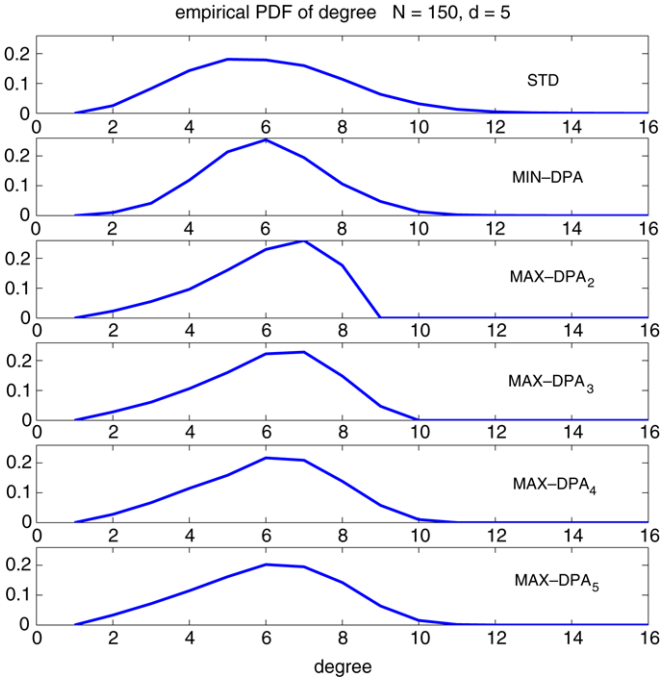

Fig. 8. Normalized histogram of degrees for $N = 150$, $d = 5$.

graphs have uneven node distribution as in PA. Next, we analyze the statistics of node degree in connected graphs and component number and size in the disconnected graphs.

Degree distribution of C-CRUGs: The distribution of the degree of an arbitrary node is an important property of a graph from a practical as well as theoretical point of view. If the standard deviation of node degree distribution is smaller in a graph, the graph is expected to have fewer critical nodes. This can make network deployment more robust in terms of routing and can also provide better load distribution under uniform traffic.

The standard deviations of these distributions are given in Fig. 7. We observe that MIN-DPA and MAX-DPA generate connected graphs with smaller standard deviation in node degrees compared to the standard algorithm. This is a somewhat expected result as these algorithms place nodes in a less randomized way by putting additional constraints. In general, standard deviation of MAX-DPA decreases as $d_{\max}$ approaches d. Eventually, at $d_{\max} = d$, standard deviation of MAX-DPA must be equal to zero since all the nodes will have the same degree. At values $d_{\max} > 10$ (not shown in the graph), the standard deviation exceeds the standard deviation of the standard algorithm and approaches the standard deviation of PA, which is higher due to the pile up around the initially placed nodes. The histograms of degree distributions are also of interest. In Fig. 8, we plot the normalized histograms of node degrees to see their empirical probability density function.

Table 1
Average number of components among disconnected graphs, $N = 150$

Algorithm	$d = 5$	$d = 8$	$d = 15$	$d = 20$
Standard	5.53	2.29	2.00	–
MIN-DPA	3.29	2.25	–	–
MAX-DPA$_2$	2.24	–	–	–
MAX-DPA$_3$	2.89	–	–	–
MAX-DPA$_4$	3.59	2.00	–	–
MAX-DPA$_5$	4.35	2.14	–	–

Table 2
Average size of the largest component among disconnected graphs, $N = 150$

Algorithm	$d = 5$	$d = 8$	$d = 15$	$d = 20$
Standard	123.55	147.05	148.67	–
MIN-DPA	122.08	144.25	–	–
MAX-DPA$_2$	147.04	–	–	–
MAX-DPA$_3$	145.43	–	–	–
MAX-DPA$_4$	143.58	148.19	–	–
MAX-DPA$_5$	141.70	147.84	–	–

We observe that the C-CRUGs generated by different algorithms having the same average degree differ in the way node degrees are distributed. The standard algorithm distributes the degrees in a larger range and in a more uniform fashion. The MIN-DPA does not allow too small node degrees. The range of MAX-DPA is narrower as a result of the maximum degree condition used in this algorithm.

Component analysis of disconnected graphs: The number and size of components formed in the disconnected instances of the candidate graph help us understand how close these graphs are to being connected. Table 1 and Table 2 give the average number of components and the average size of the largest component, respectively.

We note that at high densities ($d = 15, 20$), graphs generated are connected with probability close to 1 and practically it is very hard to generate disconnected graphs. We run our simulations until 100 connected graphs are obtained. If the generated graph is connected, the number of components is 1.

For all algorithms at sparse and medium d values ($d = 5, 8$), we see that the size of the largest component is close to the number of nodes N. On average more than 80% of the nodes belong to the same component. Average number of components is less than 6 for all protocols. We conclude that disconnected graphs generated have one very large component and the rest of the nodes constitute a few very small components. This result suggests that in most cases, full connectivity is prevented by a small number of nodes. Therefore, it might be useful to define new metrics like the percentage of connected nodes to quantify how well the network is connected. A similar conclusion is reached in [4]: it is observed that, in a network with around the same number of nodes (128), if we half the transmission range from its value required for full connectivity, 90% of the nodes stay connected forming one large component. In other words, we have to pay a high price in transmission range to increase connectivity from 90% to 100%.

3.2. Generating C-CRUGs for wireless sensor networks

In this section we aim to generate connected graphs that represent the active sensors in a wireless sensor network. Usually each sensor is turned on or off based on its contribution to the overall coverage area. The sensing region of a node is defined as the disk centered at this node with radius r_s, the sensing range, which is common for all nodes. Typically the sensing range of sensor nodes are shorter than their transmission range R. We propose Coverage Algorithm 1 and 2 (CA1, CA2) to generate connected graphs representing the graphs created after some of the sensors are turned off.

Both CA1 and CA2 fall in the class of acceptance/rejection based algorithms. That is, a randomly chosen candidate position is accepted if it satisfies a coverage constraint. These two algorithms differ in their coverage constraints. In general, coverage constraints aim to eliminate (or turn off) the nodes that do not contribute to the network sensing coverage.

3.2.1. Coverage algorithm 1 (CA1)

In CA1 a candidate position is accepted if it satisfies the following coverage constraint. The nodes within r_s of the candidate position are called sensing neighbors of the position. A position satisfies the coverage constraint if its sensing region is not completely covered by its sensing neighbors. In other words, placing the new node at this position will result in a non-zero increase in the overall coverage area of the network.

We make use of Theorem 1 of [13], which was also proven in [14] (Theorem 2), to check the coverage constraint. This theorem states that a sensing region is fully covered if there are at least two sensing regions intersecting with it, i.e., has at least two sensing neighbors, and every intersection point of any two of these intersecting sensing regions is covered by a third intersecting sensing region.

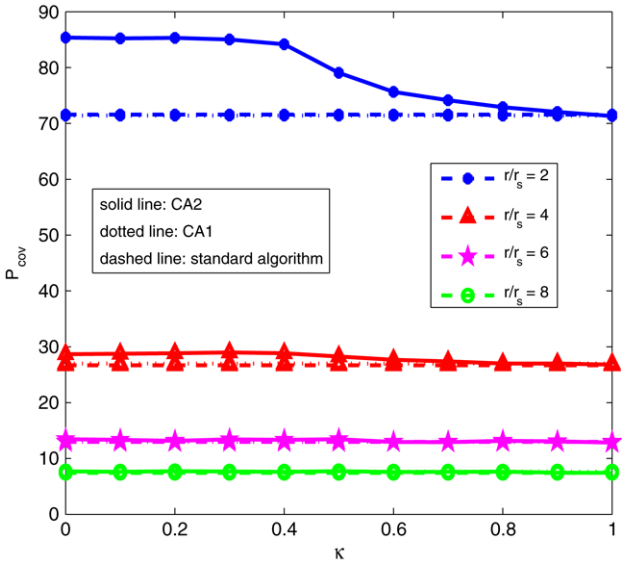

Fig. 9. P_{cov} overall percentage of the area covered by the connected graphs generated by CA1, CA2, and the standard algorithm for $N = 100$, $d = 5$, and different r/r_s ratios (2, 4, 6, 8). Results for CA2 are given as a function of κ.

3.2.2. Coverage algorithm 2 (CA2)

The CA2 uses a more strict coverage condition than CA1. A candidate position is rejected if a certain portion of its sensing area has already been covered by its sensing neighbors. Let us call the number of sensing neighbors of the candidate position as n_s. The sensing region of the candidate position is denoted by $\mathcal{R}$ and the sensing region of sensing neighbor i is denoted by $\mathcal{R}_i$. The total area already covered by the sensing neighbors is given by $A_c = \text{area}(\mathcal{R} \cap (\mathcal{R}_1 \cup \mathcal{R}_2 \ldots \cup \mathcal{R}_{n_s}))$. A natural coverage constraint is to accept the candidate position if

$$\frac{A_c}{\pi r_s^2} < \kappa,$$

where $0 \leq \kappa \leq 1$ is a parameter that controls the strictness of the constraint. However, deriving a closed form expression for A_c was proven to be difficult. Thus, we propose to use the following simple upper bound as a crude approximation for A_c:

$$A_c \leq \sum_{i=1}^{n_s} \text{area}(\mathcal{R} \cap \mathcal{R}_i) \triangleq A_{c,\text{approx}}.$$

In CA2 a candidate position is accepted if $A_{c,\text{approx}}/(\pi r_s^2) < \kappa$.

3.2.3. Performance of CA1 and CA2

In this section we compare CA1 and CA2 to the standard algorithm in terms of the fraction of connected graphs (P_c), connected graph generation time (T_c) and P_{cov}, which is defined as the percentage of the overall area covered by the connected graphs. P_{cov} of a graph is calculated using Monte Carlo integration [17]. That is, a large number of uniformly distributed random positions are generated and the percentage of the covered area is approximated by the percentage of positions that are within the coverage of at least one node. In Fig. 9 we plot the overall coverage of connected graphs generated by CA1 and CA2 with $N = 100$, $d = 5$, and different r/r_s ratios. The x-axis shows the κ value used for CA2. In this figure, we observe that for low r/r_s ratios, CA2 can significantly increase the total coverage area compared to the standard algorithm if κ value is selected small. However, CA1 has almost the same total coverage area as the standard algorithm. Fig. 11 shows the fraction of connected graphs, i.e., P_c, for the same set of parameters. It is seen that CA2 provides a higher P_c as well compared to both CA1 and the standard algorithm. Moreover, using a small κ value improves P_c. In Fig. 10, we plot the connected graph generation time (T_c) and we observe that, as expected, the increase in P_c of CA2 shown in Fig. 11 results in a decrease in T_c. Hence, we conclude that CA2 can generate connected graphs with a larger coverage in a shorter time compared to the standard algorithm. We note that a small κ value will force the algorithm to place the nodes as far from each other as possible. However, if r/r_s is large, i.e., r_s is small this becomes less effective since there is less overlap between the sensing regions of the nodes. For very small r_s, the coverage constraint is always satisfied in both CA1 and CA2. Hence, the performance is very close to the performance of the standard algorithm.

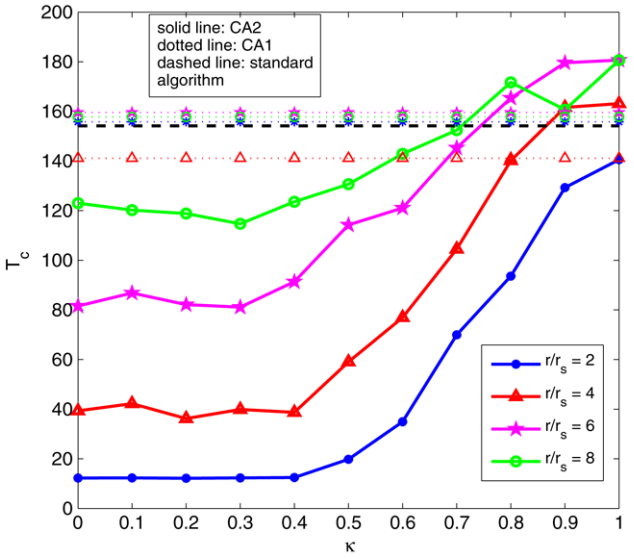

Fig. 10. T_c time required to generate connected graphs using CA1, CA2, and the standard algorithm for $N = 100$, $d = 5$, and different r/r_s ratios (2, 4, 6, 8). Results for CA2 are given as a function of κ.

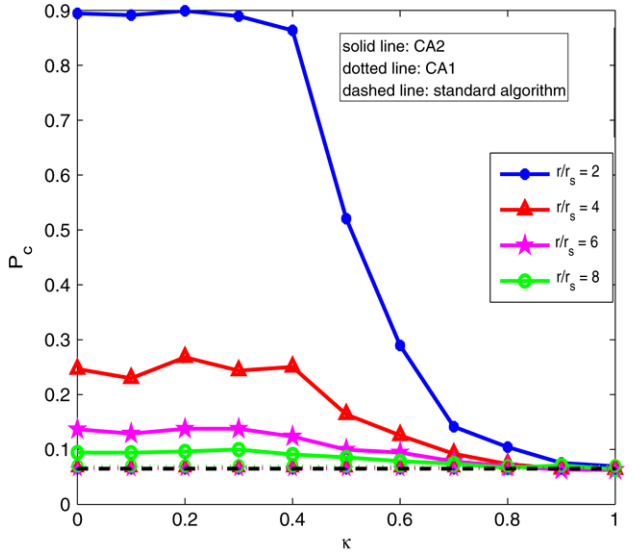

Fig. 11. P_c fraction of connected graphs among all generated graphs using CA1, CA2, and the standard algorithm for $N = 100$, $d = 5$, and different r/r_s ratios (2, 4, 6, 8). Results for CA2 are given as a function of κ.

3.3. Generating C-CRUGs for wireless internet networks

In wireless Internet access a small number of the nodes, called access points (AP), are connected to the outside world with high capacity links, usually through wireline connections. The rest of the nodes, individual users, access the Internet by reaching any of the APs either directly or through multihop communication. Since user-to-user and user-to-AP links are wireless links, they usually have lower capacity.

3.3.1. Clustered standard (C-standard) and clustered MIN-DPA (C-MIN-DPA)

We modify the standard algorithm and MIN-DPA algorithm, presented in Section 3.1.1, to represent the networks formed for wireless Internet access. We also explored MAX-DPA for this scenario. The C-MIN-DPA outperforms C-MAX-DPA in terms of generation time, hence, we only present results on C-MIN-DPA here. The number of APs is denoted by K. For given N, K and d values, these algorithms proceed as described below:

– First, K APs are distributed randomly in the area choosing independent and uniformly distributed positions. These APs are assumed to be connected (e.g., by wired links).

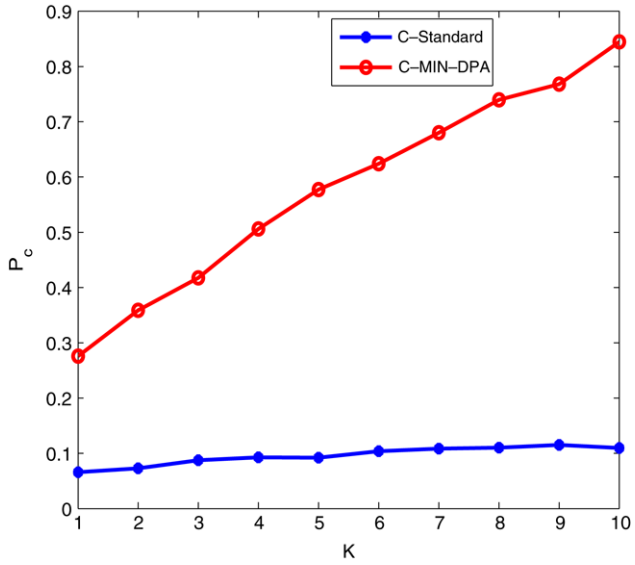

Fig. 12. P_c calculated over 100 connected graphs with $N = 100$, $d = 5$ using the C-standard algorithm and C-MIN-DPA with different K values.

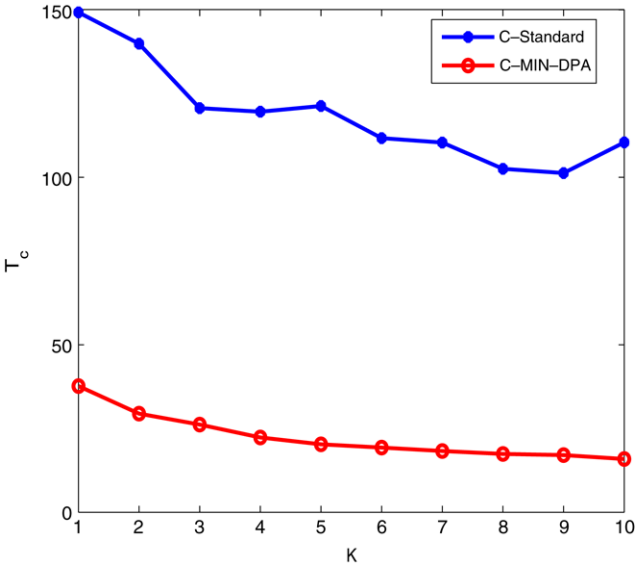

Fig. 13. T_c calculated over 100 connected graphs with $N = 100$, $d = 5$ using the C-standard algorithm and C-MIN-DPA with different K values.

- The approximate radius r is calculated as $r = \sqrt{dA/((N + K - 1)\pi)}$.
- Then, N nodes (users) are placed one by one according to the selected algorithm (standard or MIN-DPA). Since the AP-AP links have much higher capacities than the wireless links with no interference to the rest of network, the approximate degree d_i of each existing node, which is updated in each round, is calculated based on the user–user and user–AP links only.
- The candidate graph is checked for connectivity by allowing only $(N + K)d/2$ shortest edges in the graph. If it is not connected, it is rejected and the same procedure is repeated. We note that having every node connected to at least one AP is equivalent to full network connectivity.

3.3.2. *Performance of C-standard and C-MIN-DPA*

In Figs. 12 and 13 we plot P_c and T_c as a function of the number of APs. Interestingly, in C-Standard adding more APs does not increase P_c significantly. In C-MIN-DPA, however, adding 10 APs to a network with 100 users, increase P_c from 0.28 to 0.85. As seen in Fig. 13, the generation time decreases as we increase the number of APs. We note that MIN-DPA is equivalent to the C-MIN-DPA with $N - 1$ users and $K = 1$ access points.

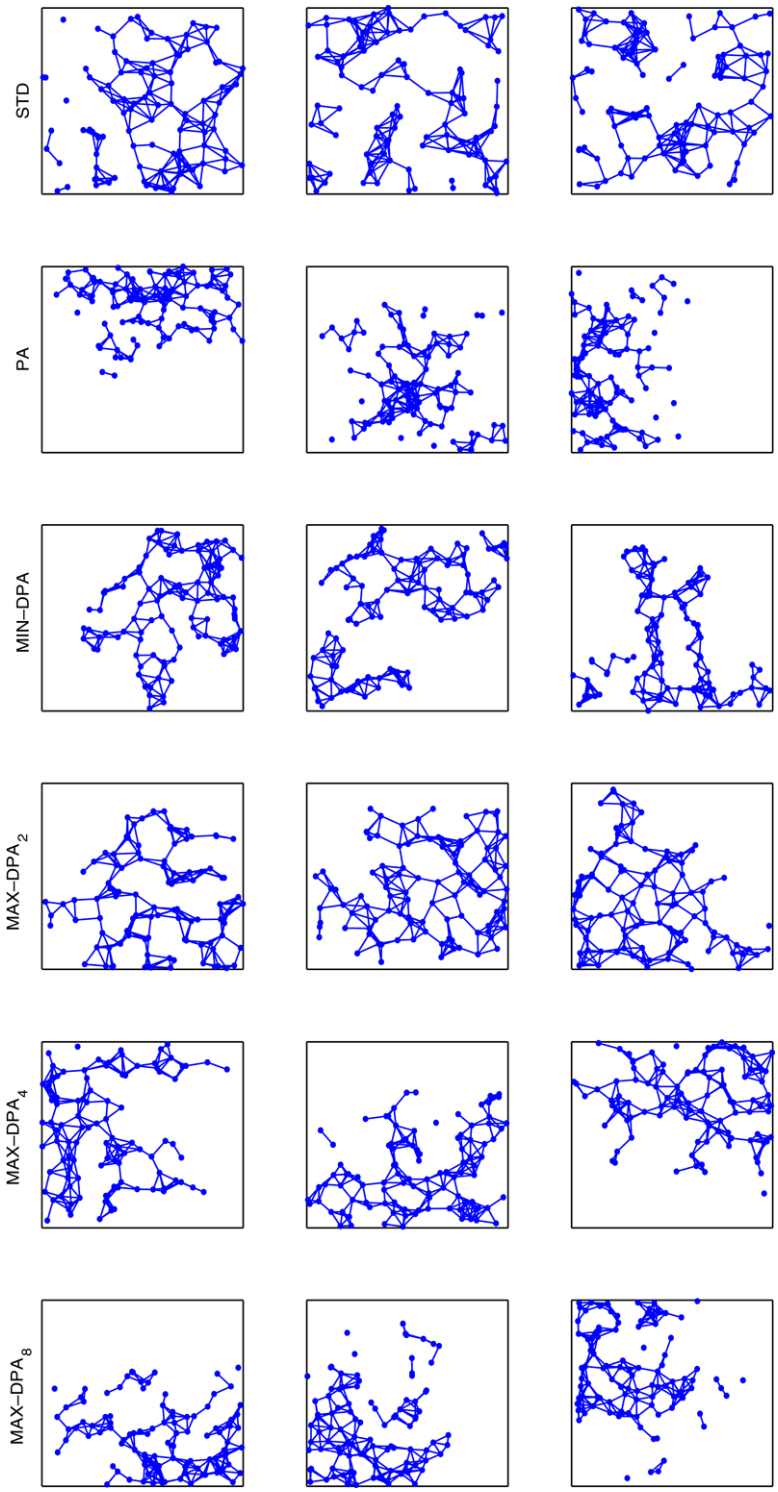

Fig. 14. Example candidate graphs generated by different algorithms, $N = 100, d = 5$.

4. Conclusions and future work

In this paper, we studied methods to generate connected random unit disk graphs where nodes are placed sequentially, which might correspond to different scenarios such as mobile users in auditorium-like environments, sensors activated

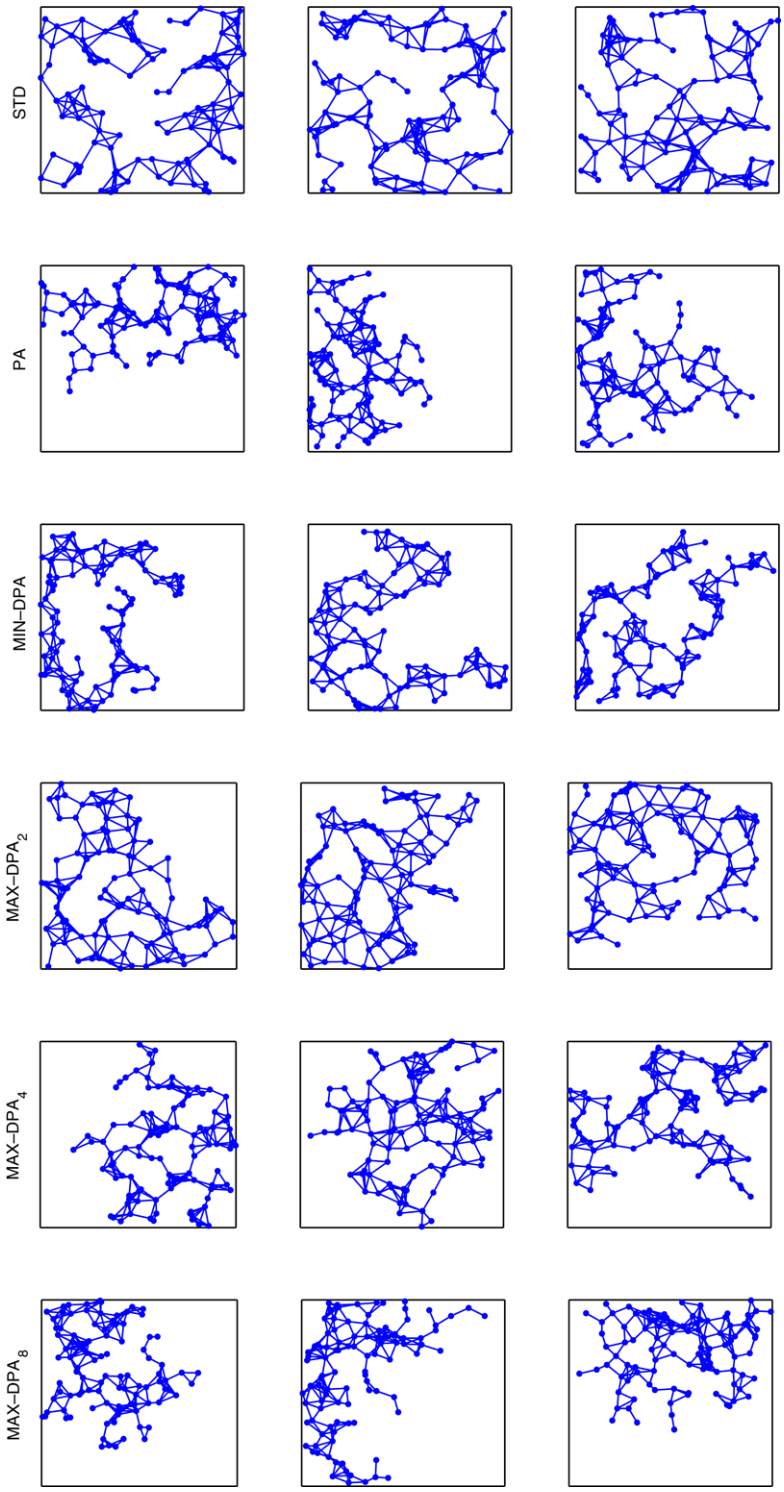

Fig. 15. Example connected graphs generated by different algorithms, $N = 100$, $d = 5$.

based on coverage and mobile users accessing wireless Internet. We especially considered sparse graphs which take much longer to generate. We described several novel algorithms for creating such graphs.

For ad hoc networks and actuator networks we proposed two algorithms (MIN-DPA and MAX-DPA) and through Monte Carlo simulations, we showed that these algorithms are significantly faster than the well-known standard method.

For these algorithms, we examined the properties of connected and disconnected graphs generated by the algorithms through degree distribution and component size analysis. We observed that in connected graphs the degree distribution differs from algorithm to algorithm. The two new algorithms result in smaller standard deviation in degree. Among the disconnected graphs generated by different algorithms, we noted a common pattern of partitioning where there is one large component and a few small components. In other words, in most cases full connectivity is prevented by a small number of nodes. This result points out that percentage of connected nodes can be an alternative measure of network connectivity.

We proposed two new algorithms CA1 and CA2 for energy efficient sensor networks and showed that CA2 efficiently generates connected graphs with improved coverage area. We also proposed a modified version of MIN-DPA for wireless Internet access scenarios, which allow multiple access points as gateways to the Internet. We note that there are also ways to generate graphs with higher node density around the APs. These include

- Modifying C-MIN-DPA to encourage node positions around the APs. This can be done by increasing the weight of the APs compared to the user nodes while selecting the center nodes.
- Modifying C-Standard to distribute each node independently but non-uniformly, for instance, using a probability distribution function with peaks at AP positions.

These modified algorithms are expected to be slower than the standard algorithm due to uneven node density but generate more realistic graphs.

More complex algorithms can be designed by selecting different combinations of proximity, degree, and coverage constraints in order to represent new scenarios such as sensor networks with multiple data fusion centers. In addition, the algorithms presented here may be modified to address some other problems. For example, we believe that the selection of best destinations for data replication [18] may follow similar algorithms.

There are a number of interesting open problems that arise from this work. How well the newly proposed graph generation methods model different scenarios that motivated this article? How will the performance of different network protocols be effected if they are evaluated in the graphs generated by the new methods? More importantly, can we predict the connectivity properties of these graphs by formal analysis? We believe that this work will stimulate new research on actuator graphs that justifiably deviate from currently prevailing uniform random distribution graphs.

Acknowledgements

The second author's research is partially supported by the UK Royal Society Wolfson Research Merit Award and NSERC Strategic Grants STPGP 336406-07 and STPSC356913-2007B

References

[1] M. Jorgic, I. Stojmenovic, M. Hauspie, D. Simplot-Ryl, Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks, in: Proc. the Third Annual IFIP Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net, June 2004.

[2] R. Meester, R. Rahul, Continuum Percolation, Cambridge University Press, 1996.

[3] T.K. Philips, S.S. Panwar, A.N. Tantawi, Connectivity properties of a packet radio network model, IEEE Transactions on Information Theory 35 (1989) 1044–1047.

[4] P. Santi, D.M. Blough, The critical transmitting range for connectivity in sparse wireless ad hoc networks, IEEE Transactions on Mobile Computing 2 (2003) 25–39.

[5] P. Gupta, P.R. Kumar, Critical power for asymptotic connectivity in wireless networks, in: Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming, Boston, 1998, pp. 547–566.

[6] O. Dousse, P. Thiran, M. Hasler, Connectivity in ad-hoc and hybrid networks, in: Proc. IEEE INFOCOM, vol. 2, June 2002.

[7] M. Desai, D. Manjunath, On the connectivity in finite ad hoc networks, IEEE Communications Letters 10 (2002) 437–490.

[8] F. Xue, P. R. Kumar, The number of neighbors needed for connectivity of wireless networks, Wireless Networks 10 (2004) 169–181.

[9] M.D. Penrose, On *k*-connectivity for a geometric random graph, Random Structures and Algorithms 15 (1999) 145–164.

[10] X.Y. Li, Y. Wang, P.J. Wan, C.W. Yi, Fault tolerant deployment and topology control in wireless ad hoc networks, Journal of Wireless Communications and Mobile Computing (WCMC) 4 (2004) 109–125.

[11] H. Zhang, J.C. Hou, On the critical total power for asymptotic *k*-connectivity in wireless networks, in: Proc. IEEE INFOCOM, March 2005.

[12] D. Dubhashi, O. Haggstrom, L. Orecchia, A. Panconesi, C. Petrioli, A. Vitaletti, Localized techniques for broadcasting in wireless sensor networks, Algorithmica 49 (2007) 412–446.

[13] A. Gallais, J. Carle, D. Simplot-Ryl, I. Stojmenovic, Localized sensor area coverage with low communication overhead, in: Fourth Annual IEEE International Conference on Pervasive Computer and Communications, PerCom, 2006.

[14] H. Zhang, J.C. Hou, Maintaining sensing coverage and connectivity in large sensor networks, Wireless Ad Hoc and Sensor Networks: An International Journal 1 (2005) 89–123.

[15] D.P. Bertsekas, Data Networks, 2nd ed., Prentice Hall, 1991, p. 401, Chapter 5.

[16] I. Stojmenovic, X. Lin, Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks, IEEE Transactions on Parallel and Distributed Systems 12 (2001) 1023–1032.

[17] E.W. Weisstein, Monte Carlo Integration, From MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/MonteCarloIntegration.html.

[18] X. Chen, Data replication approaches for ad hoc wireless networks satisfying time constraints, International Journal of Parallel, Emergent and Distributed Systems 22 (2007) 149–161.

Furuzan Atay Onat received the B.S. degree in Electrical and Electronics Engineering from Middle East Technical University, Ankara, Turkey and the M.S degree in Electrical and Computer Engineering from Rutgers University, New Jersey, in 2000 and 2004, respectively. During the summer of 2006, she was an intern at Bell Labs, Lucent Technologies, NJ. Currently, she is a Ph.D. candidate in the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada. Her research interests are in wireless communications and networking with special emphasis on multihop networks and cooperative communications.

Ivan Stojmenovic received all degrees in mathematics. He has published over 200 different papers, and edited four books on wireless ad hoc and sensor networks and applied algorithms with Wiley. He is currently editor of dozen, and founder and editor-in-chief of three journals. Stojmenovic is in the top 0.56% most cited authors in Computer Science (Citeseer 2006). One of his articles was recognized as the Fast Breaking Paper, for October 2003 (as the only one for all of computer science), by ISI. He is recipient of the Royal Society Research Merit Award, UK. He chaired and/or organized over 30 workshops and conferences. He is recently elected to IEEE Fellow status (class 2008).

Halim Yanikomeroglu was born in Giresun, Turkey, in 1968. He received a B.Sc. degree in Electrical and Electronics Engineering from the Middle East Technical University, Ankara, Turkey, in 1990, and a M.A.Sc. degree in Electrical Engineering (now ECE) and a Ph.D. degree in Electrical and Computer Engineering from the University of Toronto, Canada, in 1992 and 1998, respectively. He was with the R&D Group of Marconi Kominikasyon A.S., Ankara, Turkey, from January 1993 to July 1994. Since 1998 Dr. Yanikomeroglu has been with the Department of Systems and Computer Engineering at Carleton University, Ottawa, where he is now an Associate Professor; he is also the Associate Chair for Graduate Studies in the department. Dr. Yanikomeroglu's research interests cover many aspects of the physical, medium access, and networking layers of wireless communications with a special emphasis on multihop/relay/mesh networks and cooperative communications. He has been involved in the steering committees and technical program committees of numerous international conferences in communications; he has also given 15 tutorials in such conferences. He has been involved in the organization of the IEEE Wireless Communications and Networking Conference (WCNC) over the years. He is a member of the WCNC Steering Committee, and he was the Technical Program Co-Chair of WCNC 2004. He is the Technical Program Committee Chair of the IEEE WCNC 2008. He was an editor for IEEE Transactions on Wireless Communications [2002–2005] and IEEE Communications Surveys & Tutorials [2002–2003], and a guest editor for Wiley Journal on Wireless Communications & Mobile Computing. He was an Officer of the IEEE's Technical Committee on Personal Communications (Chair: 2005–06, Vice-Chair: 2003–04, Secretary: 2001–02), and he was also a member of the IEEE Communications Society's Technical Activities Council (2005–06). He is also a registered Professional Engineer in the province of Ontario, Canada.