Abstract

We propose and evaluate two models to determine the minimum number of very high frequency (VHF) repeaters necessary to accomodate a given geographic distrubiton of users. By utilizing cluster analysis, each model uniquely designs a network of open and "continuous tone-coded squelch system" (CTCSS) repeaters to simultaneously accomodate the desired user load. In addition, the models are mindful of connectivity issues and seek to establish the best connectivity for the set of users. Through the comparison of these two models, we seek to establish the minimum number of repeaters required.

In the "Bender" Snaking Model, a network is established by creating a "snake" or chain of open repeaters across the area. The model determines the most effective placement for each open repeater and is mindful to maintain channel availability cy placing CTCSS repeaters when necessary.

In the Branching Model, a backbone network is established between the two most populous areas and branch networks are subsequentially added to existing network. After the branched network has been completed, CTCSS lines are placed to both mitigate channel saturation and establish dedicated long-distance lines. This model seeks to create the best connectivity for the users in the area with the minimum number of repeaters used.

We test our models on two likely area distrubitions: a city/suburban-like user distribution and a rural-like user distribution. We compare the results and propose the minimum number of repeaters necessary for each scenario. By comparing the two models, we are able to decide if the number is realistic and what the benefits of a different network design may correlate to for the users.

Finally, we stress-test our models with 10,000 users in the same area and discuss the defects of line-of-sight propagation caused by mountains in the area.

Optimizing VHF Repeater Coordination Using Cluster Analysis

MCM Competition Problem B

Control Number: 11759

Contents

1 Problem Restatement

Without the aid of repeaters, VHF radio would only permit low-power users to communicate when direct line-of-sight could be established between the transmitter and receiver. Repeaters alleviate this restriction by amplifying and rebroadcasting these signals in order to make them available to a larger geographical area. By accounting for mutual repeater interference due to geographical proximity and utilizing a "continuous tone-coded squelch system" (CTCSS) or "private line" (PL), we seek to find the minimal number of repeaters required to support 1,000 simultaneous users. The users inhabit a 40 mile radius flat circular area and are permitted to broadcast between 145-148 MHz. The repeaters transmit frequency is 600 kHz above or below the received frequency and 54 different CTCSS tones are available. We then examine how our model adapts to accommodate 10,000 users and consider the potential defects of line-of-sight propagation in the presence of mountainous areas.

2 Assumptions and Justifications

- **Geometry is Euclidean.** Since the area is given to be flat, we assume that Euclidean geometry may be used.
- **The system is closed.** We assume that all signals originate from within our system. We also assume that there will be no outside interference in the system.
- **Antennas are isotropic.** Because the effective transmission area of each user and each antenna is relatively small compared to the whole area, we assume that antennas operate isotropically. This is a fundamental assumption made in modern network design [4].
- **Each user is a "low-power" user.** Typical VHF radio transmitters are effective across small towns without repeater support [8]. Since the area being considered is over 5000 square miles, the area of a small town is negligible and we can assume that users will not be able to communicate with one another without repeater support, i.e. each user is a low-power user.
- **Each user "plays nice."** In order to avoid users purposefully or accidentally drowning out the signals of others, we require that all users have the same equipment, e.g. all users broadcast at the same intensity and all users have the same range limitations. As a result, the requirements to connect a single user will be static. Specifically, each users will have to be within some fixed maximum distance from a repeater to connect to the network.
- **There are more than 1,000 (or 10,000) potential users.** It would be unrealistic for any company or group of individuals to place an appreciable number of repeaters in

order to accommodate one individual. We assume that the number of potential users in the area exceeds the number that must be simultaneously accomodated.

- **The geographic distribution of users is known.** The geographical distribution of users must be a known constraint before repeater requirements can be determined. The demand for network connectivity will not exist unless there is a preexisting community present.
- **Users and repeaters are distinct entities.** In reality, most VHF radio repeaters are maintained by individual users or a localized group of users. These repeaters are openly available and are typically not used as mobile stations [9]. Thus we assume that users are not broadcasting from repeaters and that they may be treated as two distinct entities.
- **VHF signals are not affected by physical entities in the area.** The physical presence of users and antennas will not interfere with the propagation of waves in the VHF spectrum. However, land features such as mountains will affect propagation [11].

3 Available Technology

The problem statement makes two different pieces of technology available: repeaters and continuous tone-coded squelch systems. We will now outline these technologies.

3.1 Repeaters

Repeaters are stationary devices that pick up weak signals (i.e. signals from users), amplify them, and retransmit them on a different frequency. This allows users to circumvent the line-of-sight limitation of VHF and broadcast their signal greater distances. To avoid inteference with the incoming (weak) signal, the repeater rebroadcasts the new (strong) signal 600 kHz above or below the received signal. To avoid repeaters interfering with one another the Metropolitan Coordination Association states that **repeaters must be at least 10 miles apart** [1]. Overlapping repeater "zones" will allow signals to pass from one repeater to another, allowing signals to travel significant distances.

Note that the range of a repeater is directly correlated to its height. The line-of-sight calculation to determine the effectve distance is given by disance in miles $= \sqrt{1.5A_f}$, where A_f is the height of the antenna in feet [11].

3.2 Continuous Tone-Coded Squelch System

Continuous Tone-Coded Squelch Systems (CTCSS), often called Private Lines (PL), further mitigates interference by associating a subaudible tone with signals being received/transmitted by the repeater. In order to communicate through a private line repeater, users must also broadcast this tone. This allows users in a densely populated area to communicate on the same channel with minimal interference. **Private line repeaters are not necessarily closed** [2] as these CTCSS tones are often published. Since it is our intention to *increase* the number of available channels, **CTCSS tones will be common knowledge to all users**.

4 The "Bender" Snake Model

The "Bender[1]" Snake Model seeks to maximize the number of connected users by efficiently creating a snake-like chain of open repeaters across the area.

Figure 1 describes this model without reclustering.

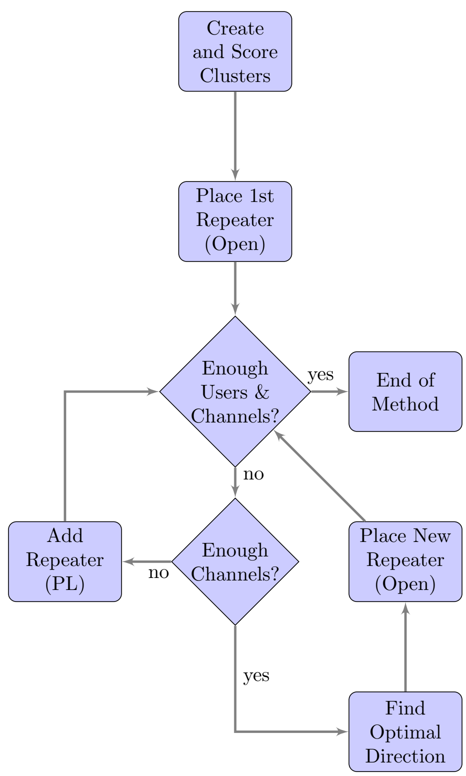


Figure 1: Flow Chart for "Bender" Snake Model

[1]Named after Bender, a team member's dog who has a habit of snaking around doors, corners and furniture.

4.1 Description

Using k-means clustering[10], we establish clusters of users to deterrmine the optimal initial open repeater placement. Optimal placement is determined by the greatest number of users we can cover when placing a single open repeater. This may also be referred to as "scoring" a cluster where a higher score correlates to coverage for more users. A second open repeater is optimally placed along the perimeter of the newly established network area. This process is repeated iteravely by placing a new open repeater along the permitter of the most recently established open repeater. The placement of the second open repeater is always in the direction of a cluster point, i.e. other high density locations. This ensures that a network is established between the most users using the least number of repeaters. The model is mindful of ensuring that enough channels are available to users by placing CTCSS repeaters accordingly.

By design, network growth tends toward establishing connectivity near and between cluster points. As more users around each cluster are accomodated, the score associated with their respective cluster should reflect that change. To account for this, cluster scores may be reevaluated to encourage intelligent networking in the model. This is known as reclustering.

4.2 Mathematical Interpretation

The model is designed to be highly versatile and supports a number of different parameters that may be changed based on a given situation. They are:

Parameter	Description
n	Number of users within 40 miles
k	Number of k-means cluster points
d_s	Maximum distance for user-to-repeater communication
h_r	Height of repeater towers
d_h	Repeater output distance
Δf	Frequency separation / channel width

Before we begin, we must define a few additional terms. Let N_c be the number of people with network connectivity (i.e. within range of an open repeater) and let N_f be the number of people with access to an available frequency range. Let O denote the number of available channels. Initially, we will have $N_c = 0$, $N_f = 0$ and $O = 0$.

So let n be the number of users within the 40 mile radius. For user i with $1 \leq i \leq n$, let (x_i, y_i) denote the user's location in Cartesian coordinates. We arrange these coordinates into an $n \times 2$ matrix M where

$$M_{j,1} = x_j \text{ and } M_{j,2} = y_j \text{ for all integers } j \in [1, n].$$

For each user, determine the number of users within the seperation distance d_s. Recall that we are using Euclidean geometry, so when considering the jth user if

$$d_s \geq \sqrt{(M_{j,1} - M_{k,1})^2 - (M_{j,2} - M_{k,2})^2}$$

then the kth user is within range of the jth user. We denote the user with the greatest number of additional users in range as the pth user.

We place an open repeater at the location of the pth user and set $R_1 = (M_{p,1}, M_{p,2})$. The allowable frequency range is 3 MHz so there will be $3/\Delta f$ channels available. The action of placing the first repeater makes this many channels available. Now $O = 3/\Delta f$.

Let N_1 be the number of people who are within range of our first repeater. We must update our N_c and N_f values accordingly. Thus

$$N_c = N_1 \text{ and } N_f = \min(N_c, O)$$

Now we calculate the metric by which this model determines Private Line (CTCSS) repeater placement. Let

$$D = \max(N_c - O, 0)$$

be the deficit of available channels (i.e. the number of users who do not have access to a repeater channel). **We update our matrix M by removing the users who are within range of the repeater**. Now M is a $(n - N_c) \times 2$ matrix. From here, if $D > 0$ we add CTCSS repeaters to mitigate this deficit and if $D = 0$, we continue to place open repeaters and expand the network.

If $D > 0$, we will add a Private (CTCSS) Line. We calculate the optimal angle to place the CTCSS line a distance of d_p from our first repeater. We then determine the location to place the CTCSS repeater.

The explicit CTCSS algorithm is given below.

Data: Previously placed open repeater location, R_{i-1}
Result: The optimal location to place a new CTCSS Line $R_i = (x_i, y_i)$
Let P be a partition of $[0, 360]$ (in our case studies $|P| = 360$);
for $\theta \in P$ **do**
    $R_\theta = (x_\theta, y_\theta) = R_{i-1} + (d_p \cos(\theta), d_p \sin(\theta))$;
    for $j \in [1, n - N_c]$ **do**
        Let $x_j = M_{j,1}$ and $y_j = M_{j,2}$;
        if $\sqrt{(x_\theta - x_j)^2 + (y_\theta - y_j)^2} \leq d_s$ **then**
            $u_\theta = u_\theta + 1$;
        end
    end
end
Let θ_c be the θ for which u_θ is maximum. Then $R_i = R_{i-1} + (\, d_p \cos(\theta_c), d_p \sin(\theta_c)\,)$ is the optimal location for the new repeater.

Algorithm 1: Finding CTCSS Repeater Location

The addition of a CTCSS line corresponds to an additional $3/\Delta f$ channels. Therefore our new value for O is $O = O' + 3/\Delta f$ where O' is the previous value of O. Now we recalculate N_c, N_f, and our deficit D using our updated matrix M (which does not include points already covered by repeaters).

If $D = 0$ then all users who desire access have access. To increase the number of supported users, we will add an open repeater to expand our network. We use "k-means" cluster analysis to determine the most densly populated areas. After the cluster points are determined, we collect their locations and let $\{c_1, ..., c_k\}$ be that collection of coordinates. The model snakes around the map by adding repeaters to include more users in the network.

The explicit open repeater placement algorithm is given below.

Data: Previously placed open repeater location, R_{i-1}
Result: The optimal location to place a new open repeater $R_i = (x_i, y_i)$
for $t \in [0, k]$ **do**
    Let θ_t be the angle from the current repeater R_{i-1} to c_t;
    $R_t = (x_t, y_t) = R_{i-1} + (d_h \cos(\theta_t), d_h \sin(\theta_t))$;
    for $j \in [0, n - N_c]$ **do**
        Let $x_j = M_{j,1}$ and $y_j = M_{j,2}$;
        if $\sqrt{(x_t - x_j)^2 + (y_t - y_j)^2} \leq d_s$ **then**
            $u_t = u_t + 1$;
        end
    end
end
Let t_c be the t for which u_t is maximum. Then $R_i = R_{i-1} + (d_h \cos(\theta_{t_c}), d_h \sin(\theta_{t_c}))$ is the optimal location for the new repeater.

Algorithm 2: Finding Open Repeater Location

The action of adding an open line did not add any new channels. We have, however, added new users to the network and must recalculate N_c, N_f, and our deficit D using our updated matrix M (which does not include points already covered by repeaters). If $D > 0$, then we apply Algorithm 1 again and if $D = 0$, we apply Algorithm 2.

We repeat this process until $N_f \geq 1000$. This would mean that there are at least 1,000 simultaneously supported users on our network.

5 The Branching Model

The aptly named Branching Model creates a backbone network of open repeaters that supports a number of branch connections. All open repeater branches are designed along the shortest distance possible. This model provides us with a point of comparison for the first model.

Figure 2 describes the creation of the backbone and branches.

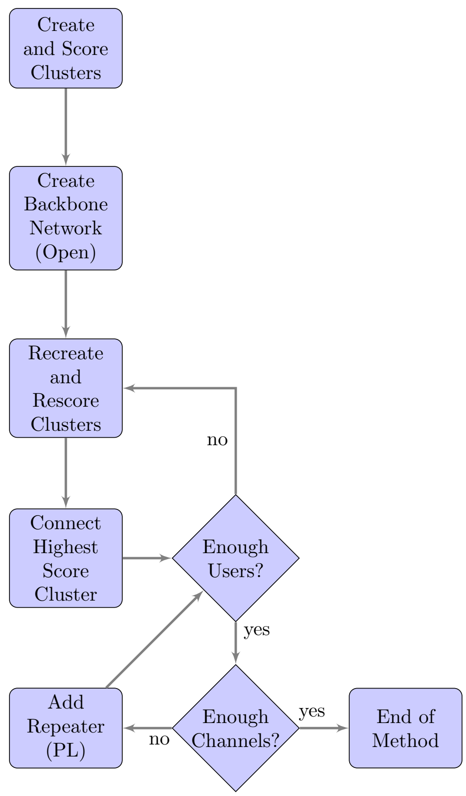


Figure 2: Flow Chart for Branching Model

5.1 Description

Akin to the first model, the Branching Method uses k-means cluster analysis and scoring to determine the optimal placement of repeaters. The difference, however, is the process in which this model creates the network. The model creates a backbone network of open repeaters between the two highest scoring clusters. After the backbone has been established, the model reclusters and rescores the remaining users and creates a branch of open repeaters between the existing network and the highest scoring cluster. This ensures that the fewest repeaters are used in branching out the network to high density locations. After the entire network has been established, the model places CTCSS repeaters to ensure channel availability is not a concern in user-dense locations so that all users may be supported simultaneously.

This model requires reclustering after every iteration.

5.2 Mathematical Intepretation

The Branching Model supports even more customization than the first model. The parameters relevant to this model are listed below.

Parameter	Description
n	Number of users within 40 miles
k	Number of k-means cluster points
d_s	Maximum distance for user-to-repeater communication
h_r	Height of repeater towers
d_h	Repeater output distance
Δf	Frequency separation/ channel width
l_n	Number of Long Distance Lines
l_c	Number of Locations in Long Distance Connections

We will review a few definitions for consistency. Let N_c be the number of people with network connectivity (i.e. within range of a repeater) and N_f be the number of people with access to an available frequency range. The number of channels available will be denoted again by O. Initially, $N_c = 0$, $N_f = 0$ and $O = 0$.

Let n be the number of users within a 40 mile radius. With k-means clustering, we identify and score clusters of users.. Letting $\{c_1, ..., c_k\}$ be the locations of the k cluster points, the model creates a backbone of repeaters between the two highest scoring cluster locations.

For each cluster point c_i, we calculate how many users are within d_s of c_i, i.e. how many users will benefit from the placement of a repeater there. We set $R_1 = c_i$ for whichever i has the most people within range. Now calculate N_c as before. The algorithm continues until the desired number of users have been added (in our case this is 1000 users). **This only means that users are within range of a repeater. This does not ensure that all users have available channels.** This will be resolved after the branched network has been established.

The explicit open repeater branching algorithm is given below.

Data: First open repeater location, $R_1 = (x, y)$
Result: The locations of the repeaters
for $i \in [1, k]$ **do**
  Let $(x_i, y_i) = c_i$;
  $\phi(i) = \sqrt{(x - x_i)^2 + (y - y_i)^2}$;
end
Then $T = (x_i, y_i)$ st $\phi(i) = \max(\phi([1, k]))$;
$\theta =$ angle between R_1,T;
$s =$ distance between R_1, T;
Set $j = 2$;
while $s > d_h$ **do**
  $R_j = R_{j-1} + (d_h \cos(\theta), d_h \sin(\theta))$;
  Recalculate s;
  $j = j + 1$;
end
while $N_c < 1000$ **do**
  Rerun "k-means" cluster analysis to obtain new $\{c_1, ..., c_k\}$;
  $T = c_i$ st c_i has the most users in range;
  Choose i st R_i is the existing repeater closest to T;
  Let s be the distance between R_T and T;
  Let θ be the distance between R_T and T;
  while $s > d_h$ **do**
    $R_j = R_{j-1} + (d_h \cos(\theta), d_h \sin(\theta))$;
    Recalculate s;
    $j = j + 1$;
  end
end

Algorithm 3: Branching Method Open Repeater Placement

With the open network established, we must resolve the issue of channel availability. This is easily accomplished by placement of CTCSS repeaters in high user density areas. In our opinion, the method in which this model establishes long-distance CTCSS lines is outstanding. By specifying a different value for l_n, one can change the number of CTCSS lines connecting the most highly populated areas. While this potentially increases the number of repeaters, it offers greater connectivity between regions. This is accomplished by first running k-means cluster analysis on the user data and choosing the l_c clusters with the most users in range. Starting from the cluster with the most users, we create a chain of repeaters with a particular CTCSS channel, a distance of d_h apart, until the next closest repeater is in range (similar to the second half of Algorithm 3). This creates a long distance connection on a specific CTCSS channel. This allows long distance users to communicate with more densely populated areas (e.g. rural or sub-urban users communicating with an urban user) without wasting an open frequency in the dense location.

Once the specified number of long distance connections have been made, any deficits in channel demand are mitigated by placing local CTCSS repeaters in highly populated areas (again by k-means clustering).

6 Model Comparison Summary

The fundamental similarities and differences between the models are:

Similarities

- **k-means clustering is prevalant in both models.** Both models make use of k-means cluster analysis to identify large groups of users. By ranking these clusters in terms of potential connected users, both models attempt to provide the most efficient connectivity scheme possible.
- **Variable-strength repeaters may be employed in both models.** By accounting for variable broadcasting strength, isolated users may be accomodated without fear of channel interference or an inordinate use of additional repeaters in both models.
- **The change of frequency from a repeater** ($\pm$600 kHz) **is resolved last in both models.** After the repeaters are configured, the "up" and "down" repeater broadcast assignments may be made in both models to fit the specific configuration of the network.

Differences

- **The models generate the network differently.** The "Bender" Snake Model places open repeaters along a continuous trajectory as determined by cluster points. In contrast, The Branching Model creates a single backbone and allows for growth in any direction toward a cluster point.
- **Reclustering is required for the Branching Model.** In order for branching to occur, the optimum target location must be determined after every iteration. The other model may utilize reclustering but does not require it.
- **The method in which private lines are introduced differs between the models.** The "Bender" Snake Model places private line repeaters as is necessary whereas the Branching Model places private line repeaters after the entire network has been established.

7 Case Studies

We developed two population distributions to test our models on. The two cases are: a city-like user distribution and a rural distribution with small towns of users. When we discuss "parity," we refer to the ±600 kHz difference in the recieving and broadcast frequencies of the repeaters. The graphs that show these assignments represent each open line repeater as a node, each labeled with "+" or "-" accordingly. We assign parity to each repeater such that no one repeater is connected to the rest of the network solely through another repeater with identical parity. This prevents a signal being either stepped up or stepped down repeatedly such that it eventually falls outside the available frequency range and cannot be recieved.

The parameters for these case studies were set to values we deemed reasonable based on our research from our referenced sources.

7.1 City Distribution

This distribution represents a city (located at the center of the area) with surrounding suburbs/neighborhoods.

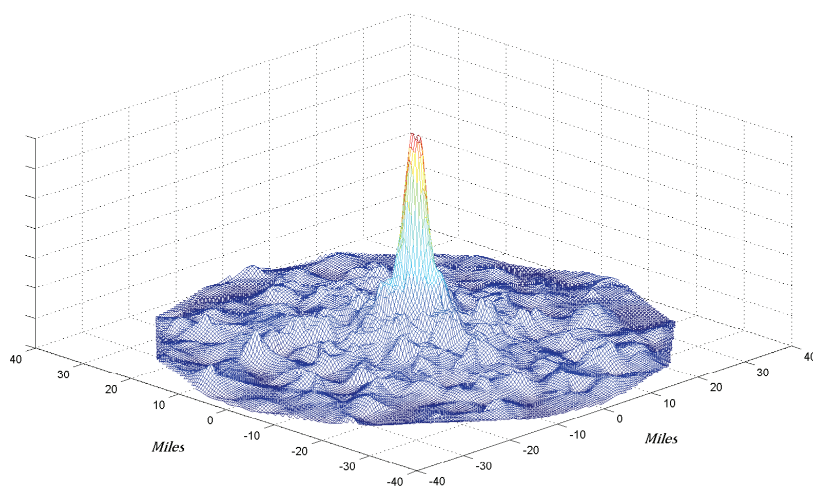


Figure 3: Surface Plot of Users

Snaking Model

Parameter	Description
$n = 1400$	The number of users within 40mi
$k = 5$	Number of k-means cluster points
$d_s = 10mi$	Maximum distance for user to repeater communication
$h_r = 150ft$	Height of repeater towers to be placed
$d_h = 15mi$	Repeater output distance[2])
$\Delta f = .025$	Frequency seperation
$l_n = 3$	Number of Long Distance Lines
$l_c = 5$	Number of Locations in Long Distance Connections

The model places the first open repeater slightly north of the city to cover most of the users located there. This creates a large channel deficit and the model places two CTCSS repeaters on this iteration to compensate. Next, a repeater is placed to the northwest and the simulation proceeds to spiral counterclockwise around the city, placing CTCSS repeaters as necessary. The model also places two CTCSS repeaters on the fourth iteration. There are 9 open line repeaters and 8 CTCSS repeaters for a **total of 17 repeaters**.

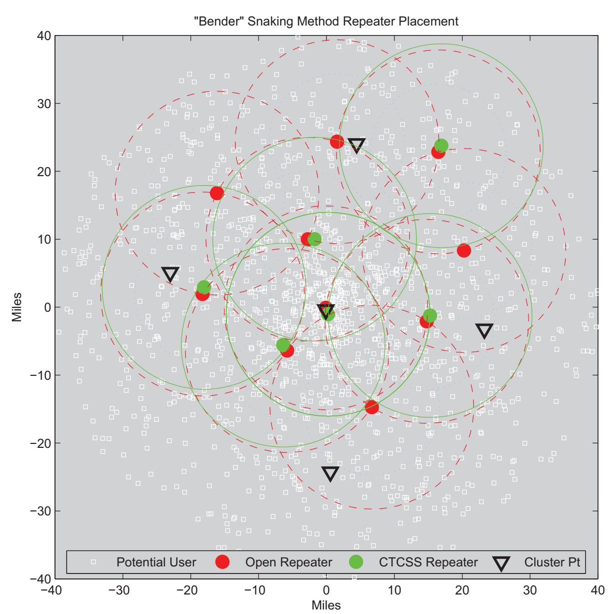


Figure 4: Repeater Placement (Snaking Model)

The model creates a closed loop of 9 open line repeaters. Since this number is odd, the parity does not work bi-directionally around the loop. There will be some signal leakage when the two step-up repeaters communicate directly but the signal will travel in the opposite direction around the entire network and be recieved.

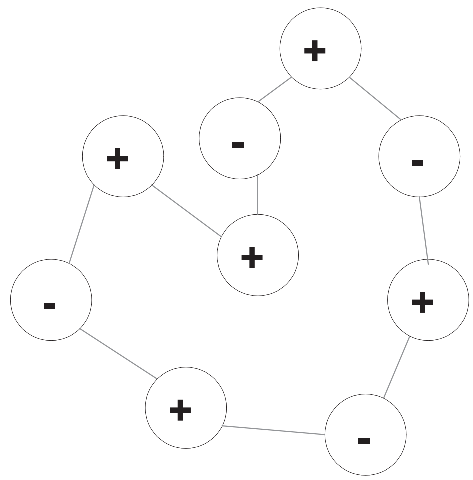


Figure 5: Repeater Parity (Snaking Model)

Branching Model The model places the first open repeater in the city and creates three main branches to cover the surrounding suburbia. This web structure is highlighed with the black lines. The branching structure of open line repeaters is designed to efficiently cover the surface area rather than simply rushing from one population center to the next linearly. This structure is created first, and CTCSS lines are placed afterwords.

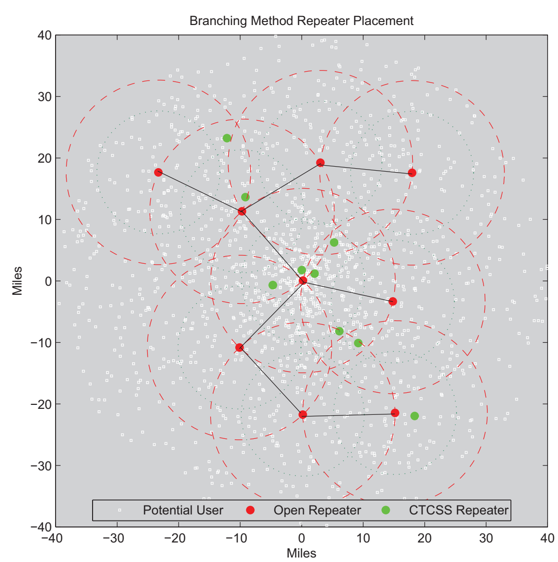


Figure 6: Repeater Placement (Branching Model)

The process of CTCSS repeater placement is designed to create higher connectivity than the Snaking model. In the Snaking model, CTCSS repeaters are used to provide more channels locally, but when the network is loaded to capacity not all users will be able to talk long-distance. In the Branching model, we assign one squelch tone as long distance, here denoted by 1 (blue circles). Each of these points has 3 CTCSS repeaters, denoted as types 1,2 and 3. Tones 4-8 provide local lines in a manner similar to Snaking. This model creates 8 open

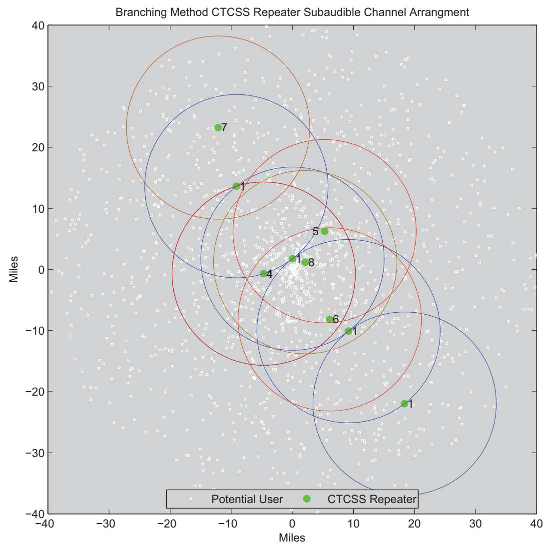


Figure 7: CTCSS Line Placement (Branching Model)

repeaters and 17 CTCSS repeaters for **a total of 26**. This number is substantually higher than the "minimum number" of 17 that the Snaking model produced. These extra lines are necessitated by the long distance backbone. The aim of this model is to provide both the *best connectivity* with the fewest number of repeaters.

The parity assignment is quite simple here.

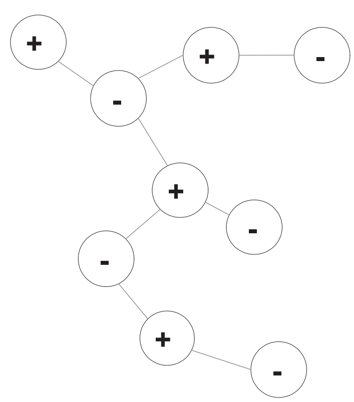


Figure 8: Repeater Parity (Branching Model)

7.2 Rural Distrubition

This distribution represents a rural area with eight small towns of concentrated population with approximately 100 people spread randomly throughout the area. The total user population is 1400. We attempt to provide connectivity to 1000 people using both the Snaking and Branching models.

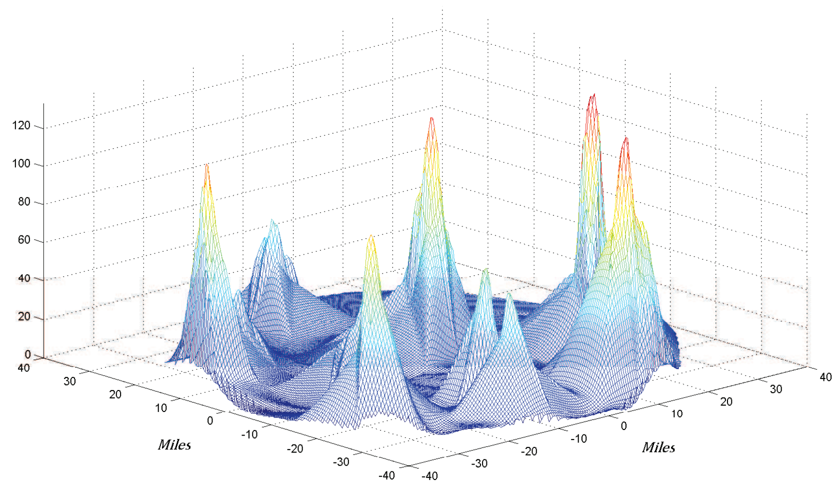


Figure 9: Surface Plot of Users

Snaking Model The model starts near the northwestern-most town. We note that it places the open-line repeater so that it covers all of the targeted town but approximately half of the population in the second town. This coverage is optimal, however a connectivity deficit is created since more than 120 people lie within range and channel availability becomes an issue. The model then places a CTCSS repeater near the open repeater to provide more channels locally. This is not sufficient to cover the deficit, so the model places an additional CTCSS repeater on the same spot, this one with a different squelch tone. This resolves the deficit so it resumes placing open-line repeaters. The next placement is essentially due south of the previous, as it gravitates towards the two southern clusters that represent the two towns in the area. The placement of this new repeater does not create a channel deficit so there is no need for another CTCSS repeater. The model proceeds south again on the next iteration, then snakes around towards the southeastern town, and finally turns north, placing CTCSS repeaters as needed along the way. The open line repeater near the middle is placed last. Note that the 6th iteration (the southeastern most group) also places two CTCSS repeaters on the same spot for a total of 8 CTCSS repeaters and 10 open repeaters. The model uses **18 total repeaters** to create a network that accomodates 1080 users.

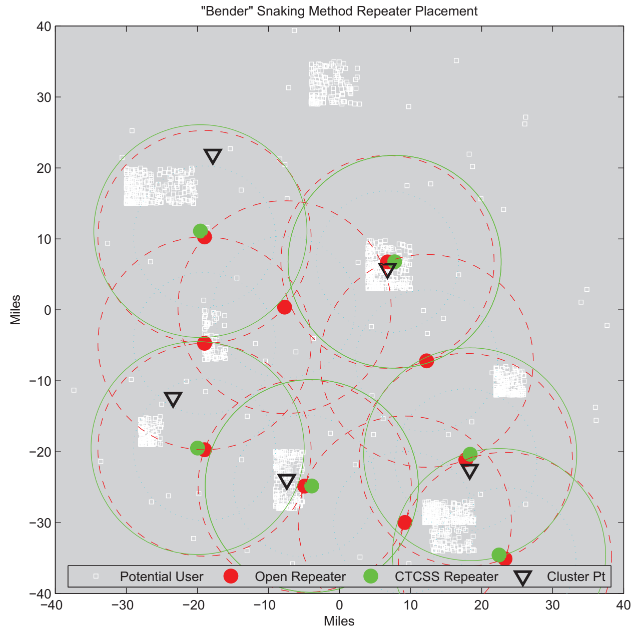


Figure 10: Repeater Placement (Snaking Model)

Repeater parity is assigned after the simulation is complete. This is generally a simple process for the Snaking model. However, note the southeastern node group where there are two step up repeaters connected. While this will result in some signal leakage outside of the available spectrum, the path that involves the step down node allows these two step up nodes to communicate without signal loss.

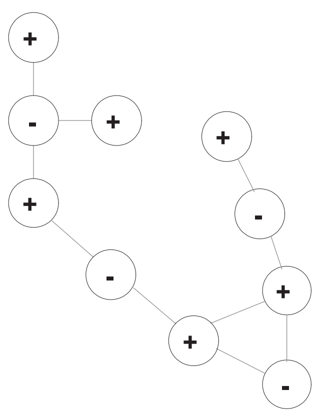


Figure 11: Repeater Parity (Snaking Model)

Branching Method The branching structure is highlighted with the black lines. The four-long node line that connects the two biggest towns is the main spine, and all other node structures branch off of this. The branching structure of open line repeaters is designed to efficiently cover the surface area rather than simply rushing from one population center to the next linearly. This structure is created first, and CTCSS lines are placed afterwords.

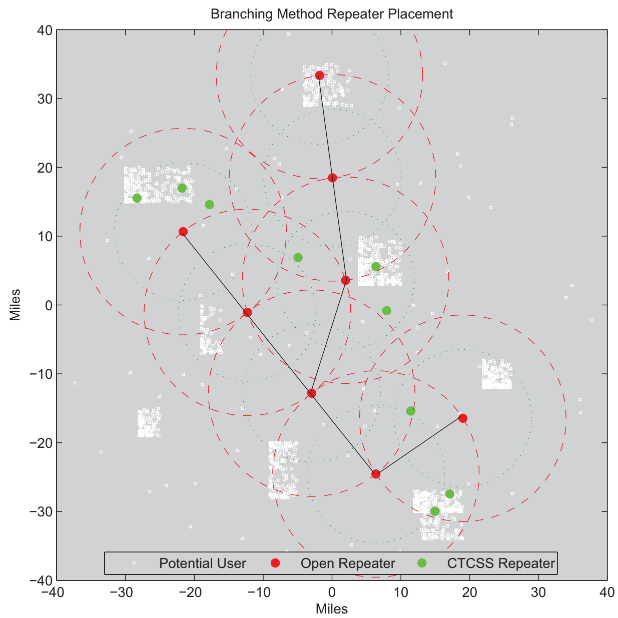


Figure 12: Repeater Placement (Branching Model)

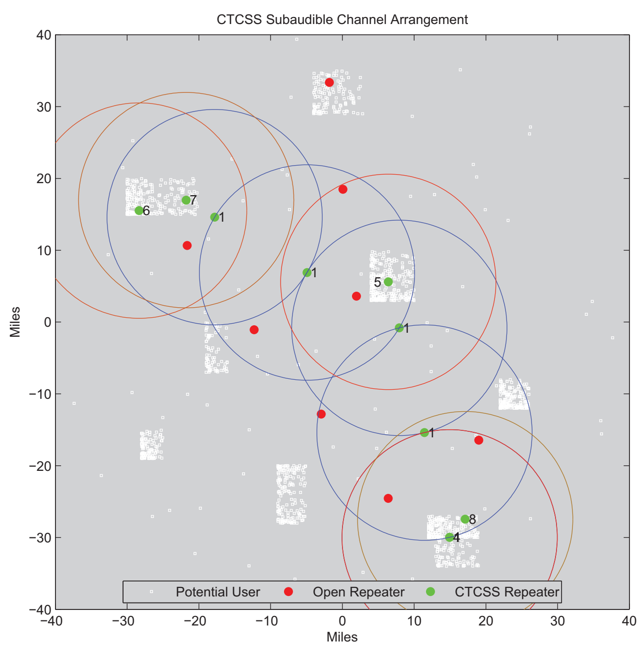


Figure 13: CTCSS Line Placement (Branching Model)

The process of CTCSS repeater placement is designed to create higher connectivity than the Snaking model. In the Snaking model, CTCSS repeaters are used to provide more channels locally, but when the network is loaded to capacity not all users will be able to talk long-distance. In the Branching model, we assign one squelch tone as long distance, here denoted by 1 (blue circles). Each of these points has 3 CTCSS repeaters, one each of types: 1, 2 and 3. Tones 4-8 provide local lines in a manner similar to the Snaking method. This model creates 8 open repeaters and 17 CTCSS repeaters for **a total of 25**. This number is substantually higher than the "minimum number" of 18 that the Snaking model produces. These extra lines are necessitated by the building of the long distance backbone. The aim of this model is to provide better connectivity, and not provide an absolute minimum repeater number.

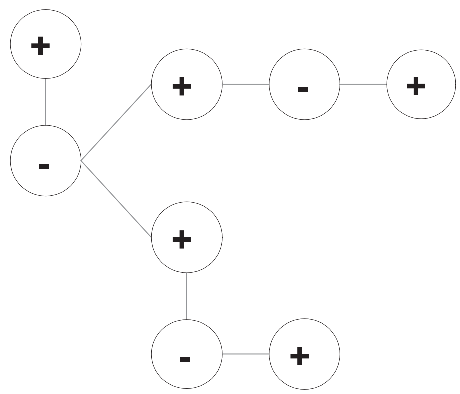


Figure 14: Repeater Parity (Branching Model)

Parity assignment for this model is fairly trivial.

8 10,000 Simultaneous Users

Our model is highly adaptable to varying situations and stresses. As such, when considering repeater placement for a network capable of simulateously supporting 10,000 users, we simply run our models against a data set with a little over 10,000 users (we use 12,000). Our models work exactly as before when trying to support 1,000 users. An important point to note however, is that the frequency seperation must be lowered to accomodate more users. We choose the frequency separation to be 10 kHz for 10,000 users (whereas before we chose 25 kHz for 1,000 users).

The minimum number of repeaters necessary to simulateously support 10,000 users is 19 open repeaters and 33 CTCSS repeaters all running on a different CTCSS tone for a total of **42 repeaters.** We conclude that even with 10,000 users, an efficient network can be established within the constaints of the problem.

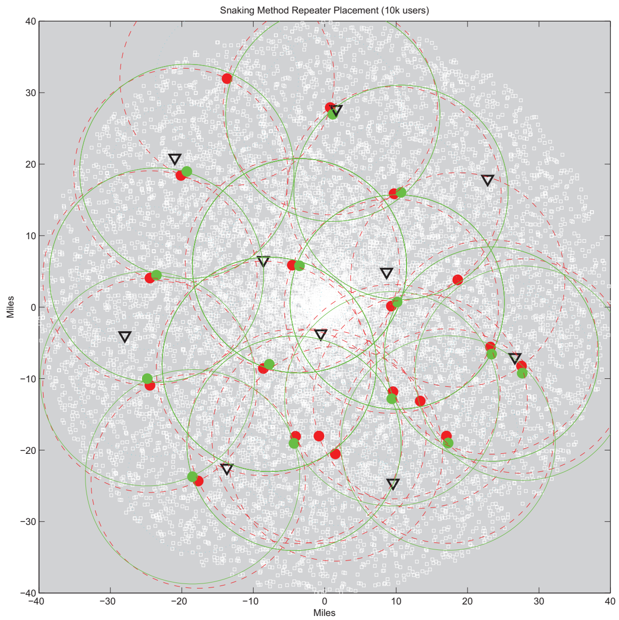


Figure 15: Results for 10,000 Simultaneous Users

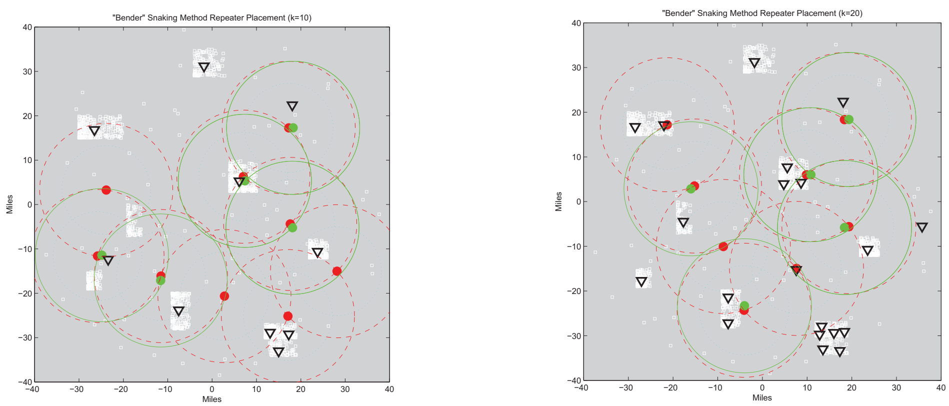
9 Mountainous Terrain

While VHF radio signals are blocked by large land structures, the line-of-sight propagation method also permits an increased effective range when the height of the antenna is increased. As a result, the mountains could be used to our advantage by placing repeaters on top of them rather than around them. This is not a completely trivial fix, however, as the new effective distance is proportional to the square root of the antenna's height. This would provide diminishing returns.

In the case where there is one large topographical peak (i.e. one large mountain peak), the obvious solution is to place one strong repeater on the mountain peak to provide the most coverage. However, now consider the case where this mountain does not have one discernable, well-defined peak and the area containing these peaks is relatively large and wide-spread (i.e. a mountain range). These numerous peaks may block the signal from a single repeater on the mountain and may not be accessible from all surrounding areas. The strength of the signal may vary with the angle due to an uneven distribution of peaks and valleys in the mountain range. In this situation, we would use a multiple-repeater network configured around the base and valleys that naturally occur in the mountain range. This would circumvent the mountain range and allow for connectivity. However, this also eliminates the line-of-sight advantage that the mountain could provide. There would most likely be ways to leverage the height advantage the mountains provide on a case-by-case basis.

10 Sensitivity to Parameters

Number of Cluster Points Since our model so heavily relies upon k-means cluster analysis, it is natural to wonder how the number of cluster points affects model performance. By running our model with variable inital cluster points ($k = 5, 10, 20$) we are able to gauge whether this has a significant impact on performance. We chose to use the rural population distrubition since it provided a more interesting analysis.



(a) $k = 10$ (b) $k = 20$

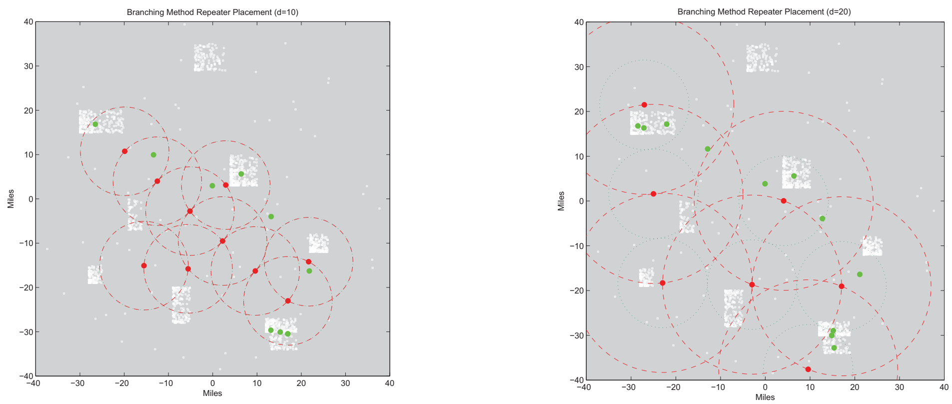
Figure 16: "Bender" Snaking Model Cluster Point Sensitivity Rural Case

When $k = 5$ or $k = 10$, the model determines that nine open line repeaters and eight CTCSS lines are necessary. When $k = 20$, the model requires eight open lines and eight CTCSS lines be available. We conclude that our model does not seem to be very sensitive to changes in the number of initial cluster points.

Seperation Distance When determining optimal placement for repeaters, our model places each repeater a distance of d_h apart so that repeater connectivity and communication is guaranteed. The value we use for d_h will have a large influence on the performance of the model. In our case studies, we set $d_h = 15$. This is the distance that a 150 ft repeater would be able to transmit its signal. For $d_h = 10$ and $d_h = 20$, the height of the necessary repeater is 66 ft. 8 in. and 266 ft. 8 in. respectively.

d_h	Tower Height (ft)	Open
10	66'	10
15	150'	8
20	266'	7

Clearly, tower height (and hence seperation distance) has a significant impact on model performance. However, while increasing the tower height does decrease the minimum number



(a) $d = 10$ (b) $d = 20$

Figure 17: Branching Model Seperation Distance Sensitivity Rural Case

of repeaters necessary, it will increase the cost of each repeater. This is a choice the user will have to make.

Initial Number of People Our algorithms run until the desired number of people are connected to the network and there are enough channels available to those people. Changing the starting population of users drastically impacts model performance. This is not surprising as a higher population would allow our model to capture the desired number of people faster.

d_h	Open	CTCSS
1400	10	8
3000	3	8
10000	1	8

Therefore, the higher the initial number of potential users, the fewer repeaters are necessary to sustain 1000 simulateous users.

11 Strengths & Weaknesses

Strengths

- **Versatility of the models.** Both models are highly versatile and can account for many changing parameters. We were very impressed that our model accomodates 10,000 users under the established requirements.

- **Smart clustering.** The fact that reclustering can be implemented (or is required) in the models creates a smarter algorithm that targets the highest priority targets *at that moment*. This allows for the “best” decision possible to be made at any given iteration.

- **Efficient use of CTCSS Lines.** Both models, even with 10,000 users, do not exhaust the use of private lines. These unused lines could accomodate more traffic if it were desired.

Weaknesses

- **Large reliance on k-cluster analysis.** Other clustering methods exist and our choice to exclusively use k-clustering limits our model. Running the models with different clustering methods would have been more illuminating to see if there were more efficient ways to network and support the users.

- **No use of Quality Threshold.** Quality Threshold (QT) is a clustering method for which a distance treshhold, not the number of clusters, is set. Implenenting this method of clustering could have improved efficiency.

- **Difficulty with populations close to target.** We found that if only 1000 users were present, the algorithm would circle around itself trying to hunt down the last remaining user. We supplanted this concern with the assumption that there were more users than we desired to connect. We felt this was a realistic liberty to take as the problem did not state that there were precisely 1000 potential users in the area.

12 Conclusion

The absolute fewest number of repeaters required to support 1,000 users is 17. This number was created by the “Bender” Snaking Model with the city distribution of users. We felt this number was reasonable for the given area and user population. The Branching Model yielded a network of 26 repeaters but established connectivty for a significantly larger area.

When considering the rural distribution, the “Bender” Snaking Model reported 18 necessary repeaters while the Branching Model reported 25. The difference in required repeaters (7) was consistent with the difference in repeaters for the city distribution (9).

By comparing the two models, we were able to make a few fundamental conclusions:

- **Better connectivity requires more repeaters.** We can’t argue with the minimum number of repreaters reported by our model but we did note that better connectivity (which potentially correlates into better service) for the users required more repeaters. This seems true to life as a more robust networks of any kind can support a greater load.

- **54 CTCSS lines are not necessary.** We never exhausted our pool of CTCSS lines. Even with the 10,000 user load, 12 CTCSS tones were still available to use.
- **CTCSS lines have multiple applications.** While CTCSS lines are primarily used to reduce interference problems in densely populated areas, they also may be used to establish dedicated long-distance communication lines.

References

[1] Metropolitan Coordination Association, Inc. Coordination Guidelines, February 2011. http://www.metrocor.net/coordination_guidelines.htm.

[2] Metropolitan Coordination Association, Inc. CTCSS (PL) Tones Frequencies, February 2011. http://www.metrocor.net/ctcss.htm.

[3] Institute for Telecommunication Sciences. High Frequency Propegation Models, February 2011. http://elbert.its.bldrdoc.gov/hf.html.

[4] Adrian W. Graham, Nicholas C. Kirkman, and Peter M. Paul. *Mobile Radio Network Design in the VHF and UHF Bands*. John Wily & Sons, Ltd, West Sussex, England, 2007.

[5] Leif J. Harcke, Kenneth S. Dueker, and David B. Leeson. Frequency Coordination in the Amateur Radio Emergency Service. *ECJ*, 1(1):31–36, 2004.

[6] Barry LcLarnon. Vhf/Uhf/Microwave Radio Propagation: A Primer for Digital Experimenters. A workship given at the 1997 TAPR/ARRL Digital Communications Conference, February 2011. http://www.tapr.org/ve3jf.dcc97.html.

[7] The American Radio Relay League. Band Plan, February 2011. http://www.arrl.org/band-plan-1#2m.

[8] Wikipedia. 2-Meter Band, February 2011. http://en.wikipedia.org/wiki/2-meter_band.

[9] Wikipedia. Amateur Radio Repeater, February 2011. http://en.wikipedia.org/wiki/Amateur_radio_repeater.

[10] Wikipedia. k-means Clustering, February 2011. http://en.wikipedia.org/wiki/K-means_clustering.

[11] Wikipedia. Very High Frequency, February 2011. http://en.wikipedia.org/wiki/Very_high_frequency.

13 Appendix A: Full-Page Plots

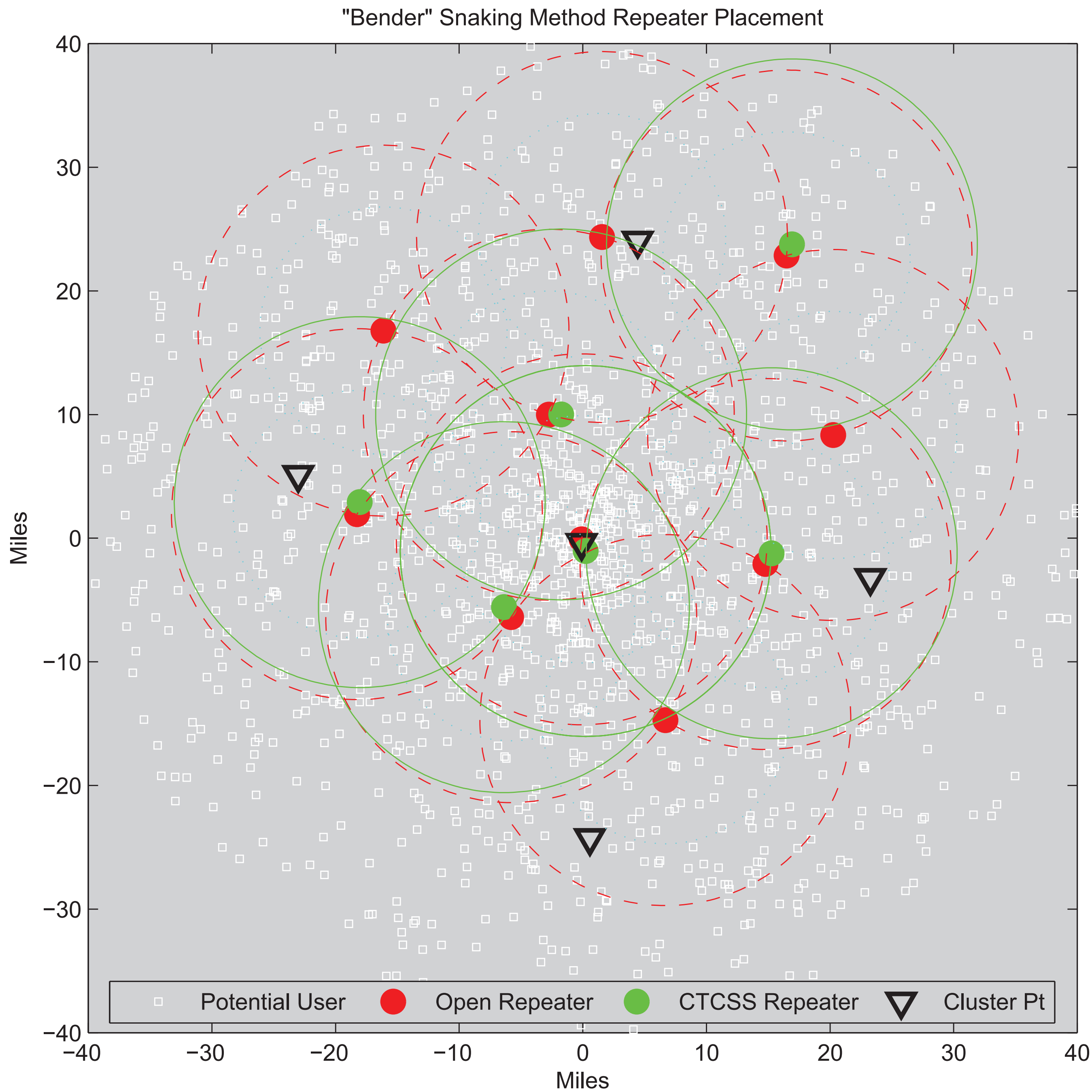


Figure 18: Repeater Placement (Snaking Model)

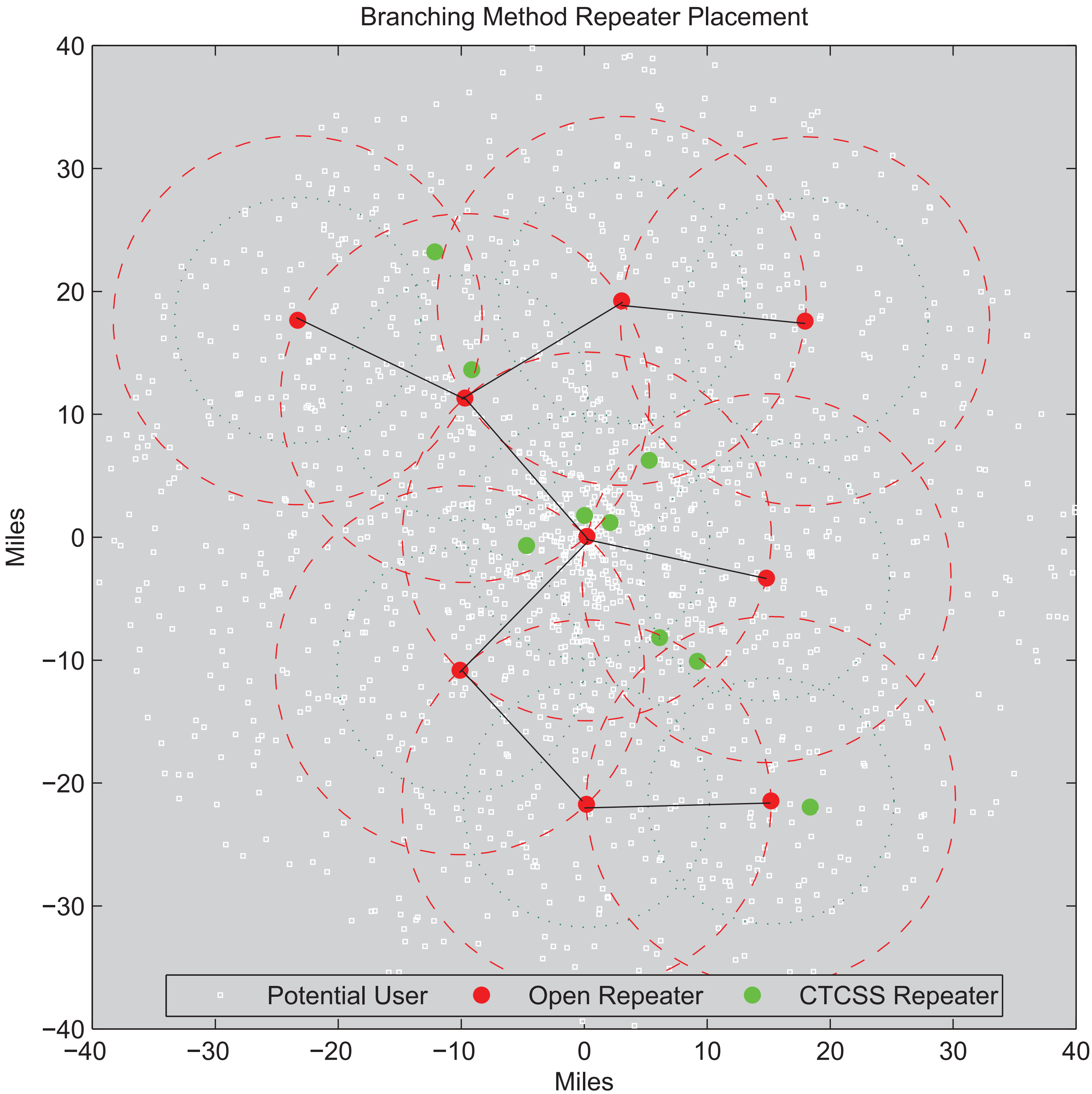


Figure 19: Repeater Placement (Branching Model)

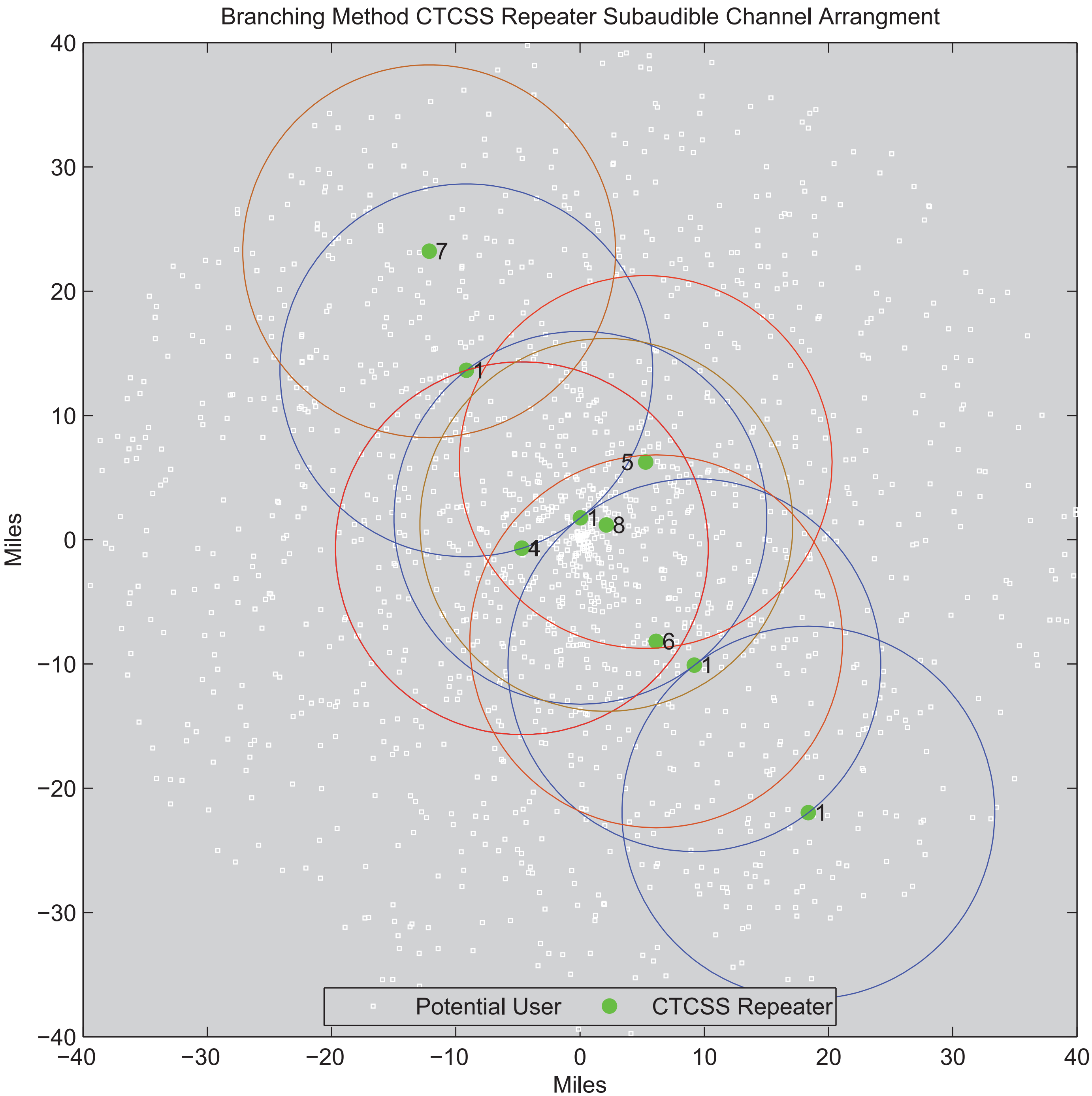


Figure 20: CTCSS Line Placement (Branching Model)

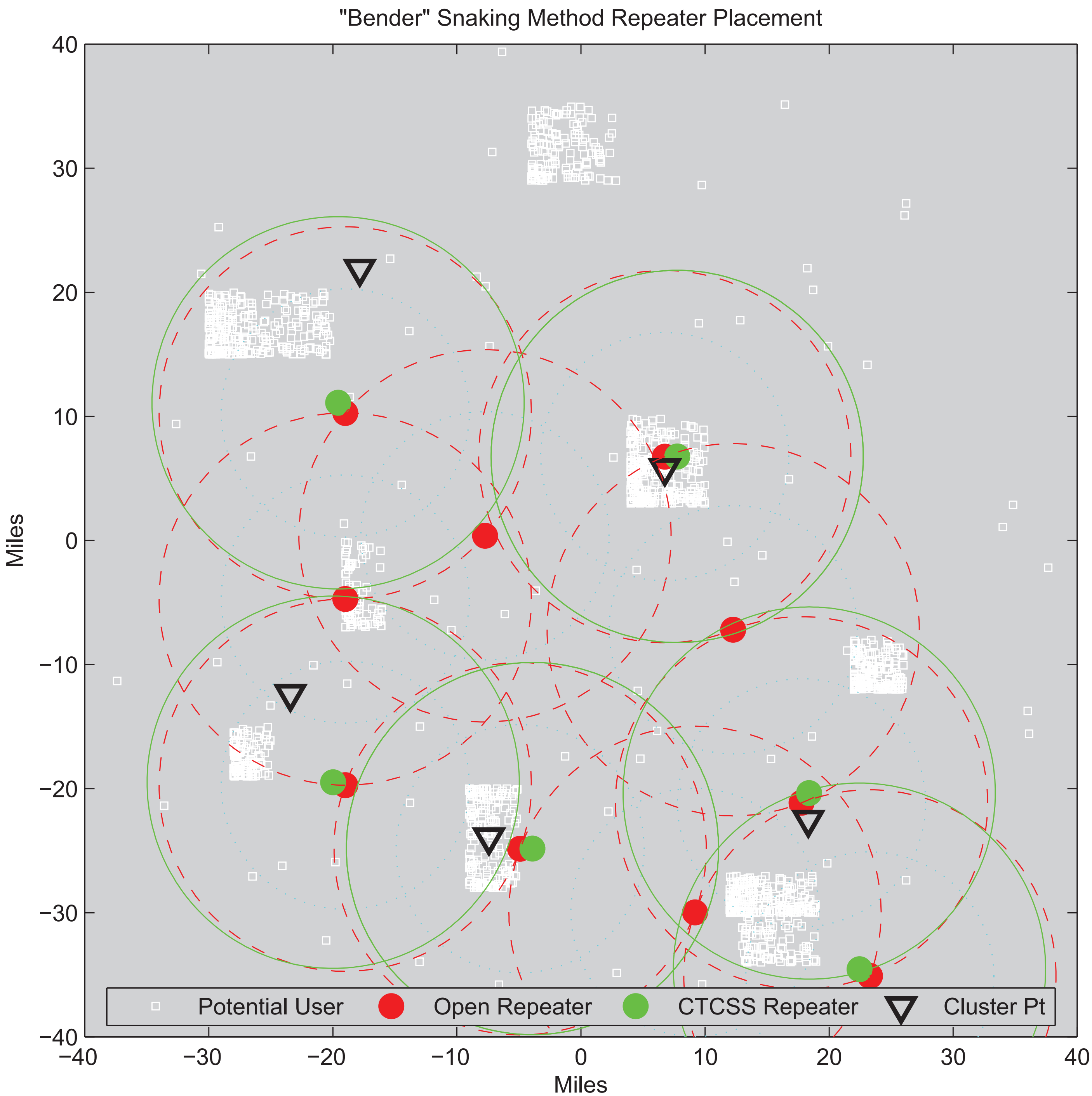


Figure 21: Repeater Placement (Snaking Model)

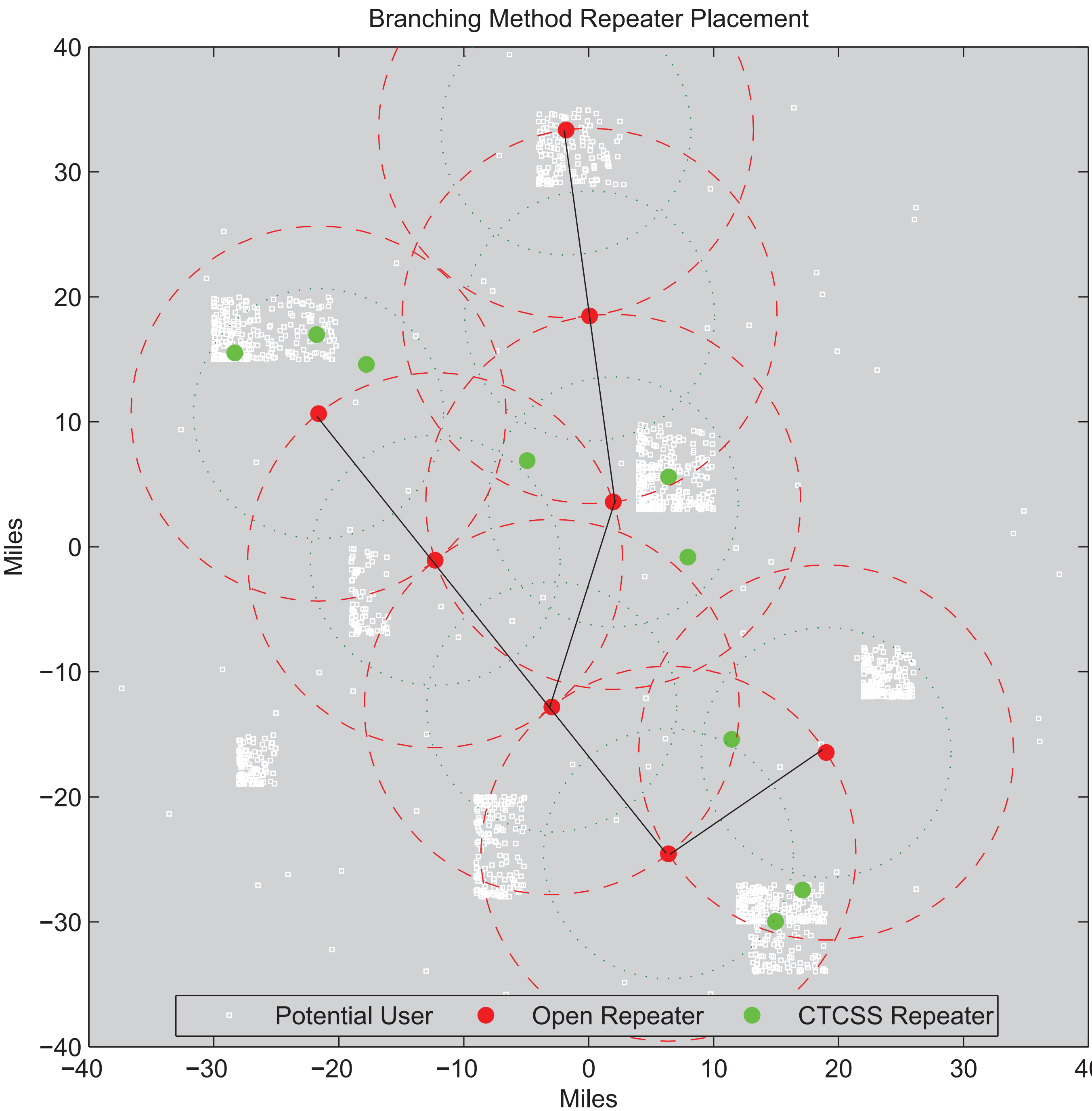


Figure 22: Repeater Placement (Branching Model)

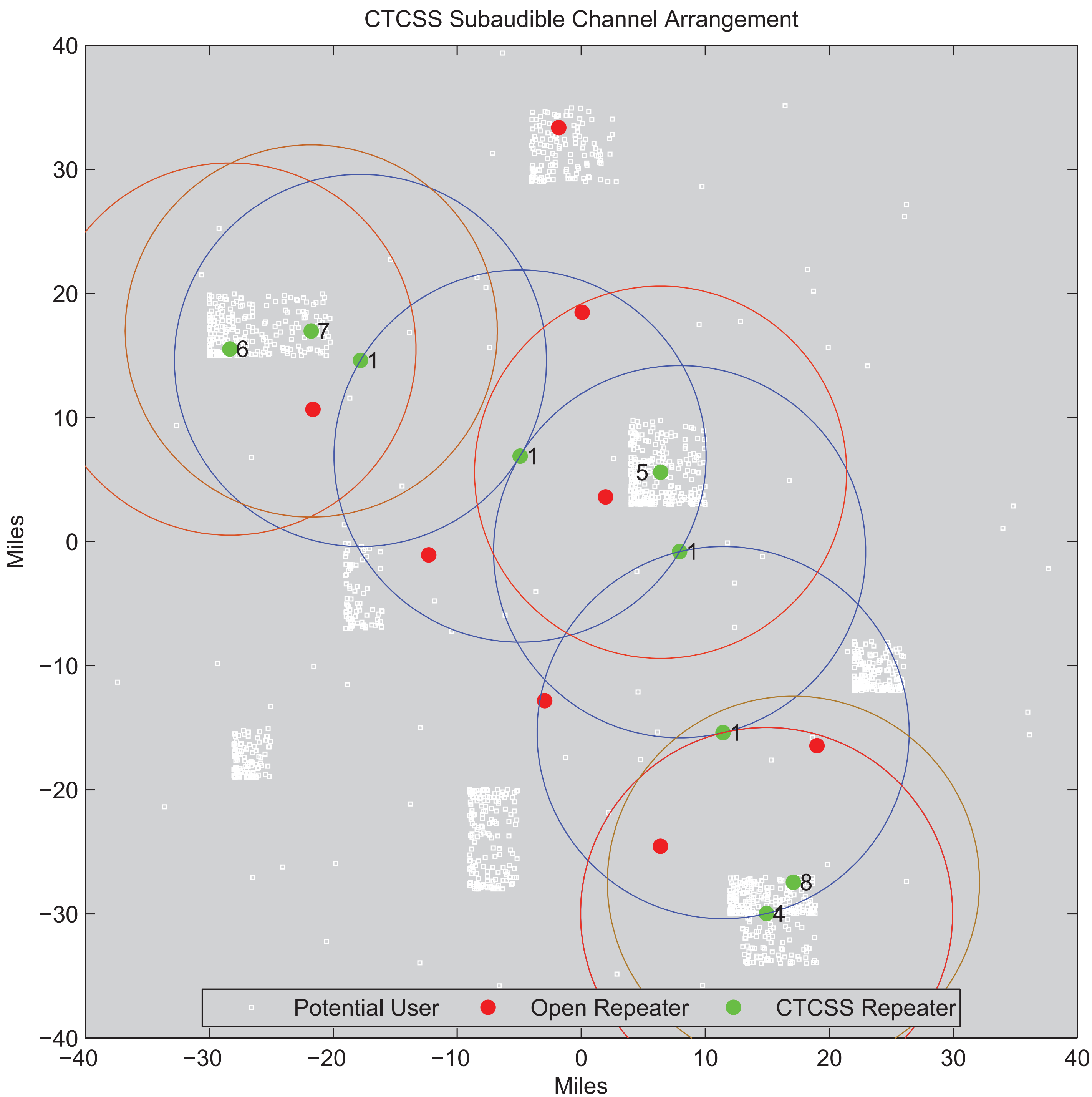


Figure 23: CTCSS Line Placement (Branching Model)

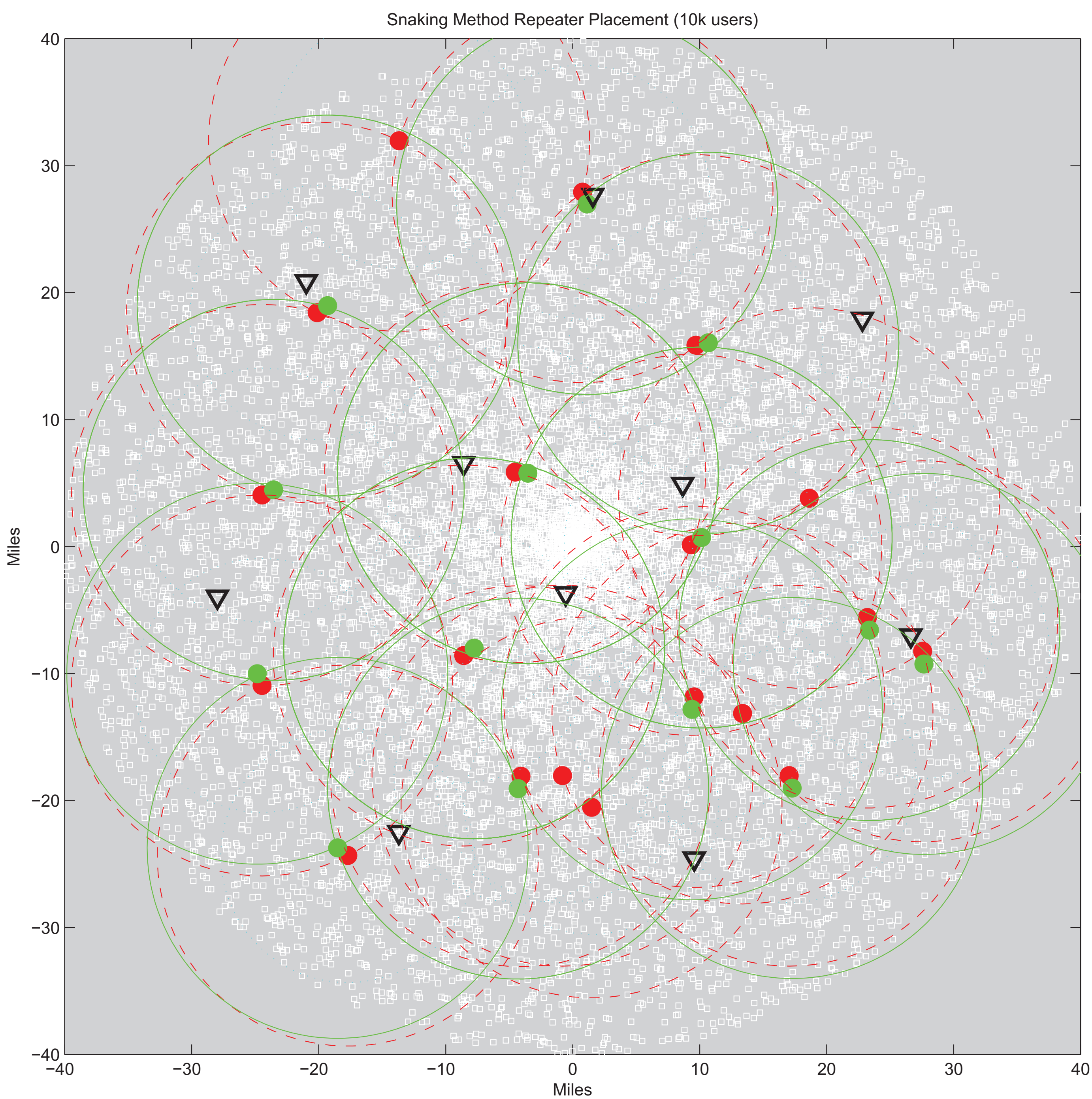


Figure 24: Results for 10,000 Simultaneous Users

14 Appendix B: Source Code

“Bender” Snaking Model:

```
load 'C:\Users\*****\Desktop\MCM 2011\townsdata3.txt';
sepdist = 10; %seperation distance
plsepdist =1; %Private line seperation distance
xcor = townsdata3(:,2);
ycor = townsdata3(:,3);
totalnum = zeros(2000,2);
connected = 0;
n=zeros(2000,5);
inrange = zeros(360);
ply = zeros(360,1);
plx = zeros(360,1);
ind = 0;
openchannels=0;
∆f = .025;
cover = 0;
it = 1;
plit =0;
rep=zeros(50,2);
population = length(xcor);
iteration=0;
concat=horzcat(xcor,ycor);
notcovered=zeros(population,2);
numclus=10;
[clusid,clusters]=kmeans(concat,numclus);
clusscr=zeros(numclus);


for i = 1:population
    for n = 1:numclus
    if clusid(i) == n
        clusscr(n) = clusscr(n) + 1;
    end
    end
end

%Plot Users and clusters
plot(concat(:,1),concat(:,2),'.','MarkerSize',1);
hold on
plot(clusters(:,1),clusters(:,2),'kx',...
     'MarkerSize',20,'LineWidth',2);
hold on
plot(clusters(:,1),clusters(:,2),'ko',...
     'MarkerSize',20,'LineWidth',2);
hold off
```

```
%Find inital repeater location
for i=1:5,
    for j=1:population,
        n(i,j)=sqrt(((clusters(i,1)-xcor(j))^2)+((clusters(i,2)-ycor(j))^2));
        if n(i,j)≤sepdist
            totalnum(i,1)=totalnum(i,1)+1;
        end
    end
end

[C,I]=max(totalnum);
index=I(1);
rep(1,1)=clusters(index,1);
rep(1,2)=clusters(index,1);
openchannels =3/Δf;

for i=1:population,
    if sqrt(((rep(1,1)-xcor(i))^2)+((rep(1,2)-ycor(i))^2))>sepdist
        ind = ind +1;
        notcovered(ind,1)=xcor(i);
        notcovered(ind,2)=ycor(i);
    end
end

cover = totalnum(index,1);
population = population-cover;
connected = min(totalnum(index,1),openchannels);
deficit = max(totalnum(index,1) - connected,0);



disp('the first repeater will be placed at'),disp('x='),disp(rep(1,1)),disp('y='), disp(rep
disp('number connected'),disp(connected);
disp('deficit'),disp(deficit);
disp('under cover'),disp(cover);
disp('open channels'),disp(openchannels);
disp('population'),disp(population);
dlmwrite('C:\Users\*****\Desktop\MCM 2011\winner.txt',I,'\t');
dlmwrite('C:\Users\*****\Desktop\MCM 2011\distances.txt',n,'\t');
dlmwrite('C:\Users\*****\Desktop\MCM 2011\proximity.txt',totalnum,'\t')

while connected<1000,
    iteration=iteration+1;
    inrange=zeros(1000);
    onrange=zeros(1000);
        if deficit >0
            plit=plit+1;
            for theta=1:360,
                plx(theta) = rep(it,1)+plsepdist*cos((3.14159*theta)/180);
                ply(theta) = rep(it,2)+plsepdist*sin((3.14159*theta)/180);
                if ((plx(theta)≥-40) && (plx(theta)≤40))
                    if ((ply(theta)≥-40) && (ply(theta)≤40))
                        for i=1:population,
                            dist(i,1)=sqrt((plx(theta)-notcovered(i,2*iteration-1))^2+(ply
```

```
                        if dist(i,1)≤sepdist
                            inrange(theta) = inrange(theta) +1;
                        end
                    end
                end
            end
        end
        [C,I] = max(inrange);
        angle = I(1);
        reppl(plit,1)= rep(it,1)+plsepdist*cos((3.14159*angle)/180);
        reppl(plit,2)= rep(it,2)+plsepdist*sin((3.14159*angle)/180);
        added=0;

        openchannels = openchannels +(3/∆f);

        %%REMOVE ENTRIES WHICH HAVE BEEN USED
        ind=0;
        for i=1:population,
            if sqrt(((rep(it,1)-notcovered(i,2*iteration-1))^2)+((rep(it,2)-notcovered
                ind = ind +1;
                notcovered(ind,2*iteration+1)=notcovered(i,2*iteration-1);
                notcovered(ind,2*iteration+2)=notcovered(i,2*iteration);
            else
                added=added+1;
            end
        end

        cover = cover + added;
        population = population - added;
        connected = min(cover,openchannels);
        deficit = max(cover - connected,0);

        disp('the PL repeater will be placed at'),disp('x='),disp(reppl(plit,1)),disp(
        disp('added'),disp(added);
        disp('cover'),disp(cover);
        disp('people connected:'),disp(connected);
        disp('deficit'),disp(deficit);
        disp('open channels'),disp(openchannels);
        disp('population'),disp(population);
    end

if deficit==0
        it = it+1;
        inrange=zeros(5);
        onrange = zeros(5);
        d = zeros(5);
        for clusterno=1:5,

            d(clusterno)=sqrt((rep(it-1,1)-clusters(clusterno,1))^2+(rep(it-1,2)-cluste
            if ((rep(it-1,1)≥clusters(clusterno,1))&&(rep(it-1,2)≤clusters(clusterno,2
                ang(clusterno)= acos((clusters(clusterno,1)-rep(it-1,1))/d(clusterno)),
            elseif ((rep(it-1,1)>clusters(clusterno,1))&&(rep(it-1,2)≥clusters(cluster
                ang(clusterno)=2*pi-acos((clusters(clusterno,1)-rep(it-1,1))/d(cluster
            elseif ((rep(it-1,1)≤clusters(clusterno,1))&&(rep(it-1,2)≥clusters(cluster
```

```
                    ang(clusterno)=2*pi-acos((clusters(clusterno,1)-rep(it-1,1))/d(clustern
                else
                    ang(clusterno)=acos((clusters(clusterno,1)-rep(1,1))/d(clusterno));
                end


%%
%{

                if ((rep(it-1,1)≥clusters(clusterno,1))&&(rep(it-1,2)≤clusters(clusterno,2)
                    ang(clusterno)=pi-asin((clusters(clusterno,2)-rep(it-1,2))/d(clusterno)
                elseif ((rep(it-1,1)≥clusters(clusterno,1))&&(rep(it-1,2)≥clusters(clustern
                    ang(clusterno)=pi+asin((rep(it-1,2)-clusters(clusterno,2))/d(clusterno)
                elseif ((rep(it-1,1)≤clusters(clusterno,1))&&(rep(it-1,2)≥clusters(clustern
                    ang(clusterno)=2*pi-asin((rep(it-1,2)-clusters(clusterno,2))/d(clustern
                else
                    ang(clusterno)=asin((clusters(clusterno,2)-rep(it-1,2))/d(clusterno))
                end

                %}
%%
                x(clusterno)=rep(it-1,1)+sepdist*1.5*cos(ang(clusterno));
                y(clusterno)=rep(it-1,2)+sepdist*1.5*sin(ang(clusterno));
                if ((x(clusterno)≥-40) && (x(clusterno)≤40)&&(y(clusterno)≥-40)&&(y(cluste
                        for i=1:population,
                            if sqrt((x(clusterno)-notcovered(i,2*iteration-1))^2+(y(cluste
                                inrange(clusterno) = inrange(clusterno) +1;
                            end
                        end
                        for i=1:population,
                            if sqrt((clusters(clusterno,1)-notcovered(i,2*iteration-1))^2+
                                onrange(clusterno) = onrange(clusterno) +1;
                            end
                        end
                 else
                        inrange(clusterno)=-100;
                 end

             end

            [C,I] = max(inrange);

            angleid = I(1);
            rep(it,1)=rep(it-1,1)+sepdist*1.5*cos(ang(angleid));
            rep(it,2)=rep(it-1,2)+sepdist*1.5*sin(ang(angleid));

            if ((rep(it,1)≥-40)&&(rep(it,1)≤40)&&(rep(it,2)≥-40)&&(rep(it,2)≤40))
                ind=0;
                added=0;
                for i=1:population,
                    if sqrt(((rep(it,1)-notcovered(i,2*iteration-1))^2)+((rep(it,2)-notcove
                        ind = ind +1;
                        notcovered(ind,2*iteration+1)=notcovered(i,2*iteration-1);
                        notcovered(ind,2*iteration+2)=notcovered(i,2*iteration);
```

```
                    else
                        added=added+1;
                    end
                end
            else
                theid=1;

                while ((rep(it,1)<-40)||(rep(it,1)>40)||(rep(it,2)<-40)||(rep(it,2)>40))
                    rep(it,1)=rep(it-1,1)+sepdist*1.5*cos(ang(theid));
                    rep(it,2)=rep(it-1,2)+sepdist*1.5*sin(ang(theid));
                    theid =theid +1;
                end

                ind=0;
                added=0;
                for i=1:population,
                    if sqrt(((rep(it,1)-notcovered(i,2*iteration-1))^2)+((rep(it,2)-notcove
                        ind = ind +1;
                        notcovered(ind,2*iteration+1)=notcovered(i,2*iteration-1);
                        notcovered(ind,2*iteration+2)=notcovered(i,2*iteration);
                    else
                        added=added+1;
                    end
                end
            end

            cover = cover + added;
            population = population - added;
            connected = min(cover,openchannels);
            deficit = max(cover - connected,0);

            disp('the open repeater will be placed at'),disp('x='),disp(rep(it,1)),disp('y=
            disp('added'),disp(added);
            disp('cover'),disp(cover);
            disp('open channels'),disp(openchannels);
            disp('people connected:'),disp(connected);
            disp('deficit'),disp(deficit);
            disp('population'),disp(population);
    end
end

disp('number of open repeaters'),disp(it);
disp('number of PL repeaters'),disp(plit);

xplot=zeros(it,1);
yplot=zeros(it,1);
for i=1:it
    xplot(i,1)=rep(i,1);
    yplot(i,1)=rep(i,2);
end

xplotpl=zeros(plit,1);
yplotpl=zeros(plit,1);
for i=1:plit
```

```
    xplotpl(i,1)=reppl(i,1);
    yplotpl(i,1)=reppl(i,2);
end

%}

plot(xcor,ycor,'s','MarkerSize',4,'MarkerEdgeColor',[1 1 1]);
xlabel('Miles');
ylabel('Miles');
title('Snaking Method Repeater Placement');
hold on

plot(xplot,yplot,'o','MarkerFaceColor','r','MarkerEdgeColor','r','MarkerSize',10,'LineWidt
hold on


plot(xplotpl,yplotpl,'o','MarkerFaceColor','g','MarkerEdgeColor','g','MarkerSize',10,'LineW
hold on

plot(clusters(:,1),clusters(:,2),'v',...
     'MarkerSize',10,'LineWidth',2,'MarkerEdgeColor','k');
hold on





 for i=1:it,
    h=rep(i,1); k=rep(i,2); r=15; N=256;
    t=(0:N)*2*pi/N;
    plot(r*cos(t)+h,r*sin(t)+k,'Color','r','LineStyle','--');
    hold on
 end


  for i=1:it,
    h=rep(i,1); k=rep(i,2); r=10; N=256;
    t=(0:N)*2*pi/N;
    plot(r*cos(t)+h,r*sin(t)+k,'Color','c','LineStyle',':');
    hold on
 end

 for i=1:plit,
    h=reppl(i,1); k=reppl(i,2); r=15; N=256;
    t=(0:N)*2*pi/N;
    plot(r*cos(t)+h,r*sin(t)+k,'Color','g','LineStyle','-');
    hold on
 end


 axis([-40 40 -40 40]);
 axis('square')
```

```
hold off
```

Branching Model:

```
load 'C:\Users\*****\Desktop\MCM 2011\newtestdata2.txt';
newx = newtestdata2(:,2); %x cor of data
newy = newtestdata2(:,3); %y cor of data
notcovered = horzcat(newx,newy);
surfsens=2;           %distance from point to include in surfscr
sepdist = 10;         %seperation distance of repeaters
scr=zeros(length(newx),1); %how many points are within surfsens
rep=zeros(40,2);      %repeater locations
numclus=5;            %number of clusters
clusscr=zeros(numclus,3); %how many points are within sepdist of cluster point
iteration=0;
clf;
numuncov=length(newx);
it=0;
connected = 0;
newrepang = 0;
numlongdist=10;
actlongdist=5;
plsepdist=15;
numberoflongdistancelines=3;
∆f=.025;
perline=3/∆f;
openchannels=perline;

for i=1:length(newx),
    for j=1:length(newx)
        if sqrt((newx(i)-newx(j))^2+(newy(i)-newy(j))^2)≤surfsens
            scr(i)=scr(i)+1;
        end
    end
end

concat = horzcat(newx,newy,scr); %% (xcor,ycor,zcor)

[¬,clusters]=kmeans(concat,numclus);

disp('initial uncovered'),disp(numuncov);


%{
%Plot Users and clusters
plot(concat(:,1),concat(:,2),'.');
hold on
plot(clusters(:,1),clusters(:,2),'kx',...
     'MarkerSize',20,'LineWidth',2);
hold on
plot(clusters(:,1),clusters(:,2),'ko',...
     'MarkerSize',20,'LineWidth',2);
```

```
hold off
%}


%Set first repeater locations
for i=1:numclus,
    reptemp(i,1)=clusters(i,1);
    reptemp(i,2)=clusters(i,2);
end

%Calculate score of each cluster and order by score
for i=1:numclus,
    for j=1:length(newx),
        if sqrt((reptemp(i,1)-newx(j))^2+(reptemp(i,2)-newy(j))^2)≤sepdist
            clusscr(i,1)=reptemp(i,1);
            clusscr(i,2)=reptemp(i,2);
            clusscr(i,3)=clusscr(i,3)+1;
        end
    end
end

srtclus = sortrows(clusscr,-3); %sorted clusters

%Set first repeater locations
rep(1,1)=srtclus(1,1);
rep(1,2)=srtclus(1,2);

added=0;
i=0;

%Calculate added and remove covered entries
for j=1:numuncov,
    if sqrt((rep(1,1)-notcovered(j,1))^2+(rep(1,2)-notcovered(j,2))^2)≤sepdist
        added=added+1;
    else
        i=i+1;
        notcovered(i,3)=notcovered(j,1);
        notcovered(i,4)=notcovered(j,2);
    end
end

repcoverage(1)=added;
numuncov=numuncov-added;
connected = connected+added;

targetloc(1)=srtclus(2,1);
targetloc(2)=srtclus(2,2);

newrepdist = sqrt((rep(1,1)-targetloc(1))^2+(rep(1,2)-targetloc(2))^2); %Distance from cur



if ((rep(1,1)≥targetloc(1))&&(rep(1,2)≤targetloc(2)))
    newrepang= acos((targetloc(1)-rep(1,1))/newrepdist);
```

```
elseif ((rep(1,1)≥targetloc(1))&&(rep(1,2)≥targetloc(2)))
    newrepang=2*pi-acos((targetloc(1)-rep(1,1))/newrepdist);
elseif ((rep(1,1)≤targetloc(1))&&(rep(1,2)≥targetloc(2)))
    newrepang=2*pi-acos((targetloc(1)-rep(1,1))/newrepdist);
else
    newrepang=acos((targetloc(1)-rep(1,1))/newrepdist);
end

repno=1;

iteration=0;
lastplaced=1;

%add repeater to make connection
while connected<1000,

    iteration=iteration+1;
    while newrepdist>sepdist,
        it=it+1;
        repno = repno+1;
        rep(repno,1)=rep(lastplaced,1)+sepdist*cos(newrepang);
        rep(repno,2)=rep(lastplaced,2)+sepdist*sin(newrepang);
        lastplaced=repno; %set the last placed to the current repeater
        newrepdist=sqrt((targetloc(1)-rep(repno,1))^2+(targetloc(2)-rep(repno,2))^2);

        %Count covered and remove from dataset
        added=0;
        i=0;
        for j=1:numuncov,
            if sqrt((rep(repno,1)-notcovered(j,2*it+1))^2+(rep(repno,2)-notcovered(j,2*it+
                added=added+1;
            else
                i=i+1;
                notcovered(i,2*it+3)=notcovered(j,2*it+1);
                notcovered(i,2*it+4)=notcovered(j,2*it+2);
            end
        end

        coveragestat=0;
        for j=1:length(newx),
            if sqrt((rep(repno,1)-newx(j))^2+(rep(repno,2)-newy(j))^2)≤sepdist
                coveragestat=coveragestat+1;
            end
        end
        repcoverage(repno)=coveragestat;
        numuncov=numuncov-added;
        connected = connected + added;
        disp('added'),disp(added);
        disp('adding another');
    end
    disp('iteration'),disp(iteration);
```

```
    %New targetloc
    concat = zeros(numuncov,2);
    for i=1:numuncov;
        concat(i,1) = notcovered(i,2*it+3);
        concat(i,2) = notcovered(i,2*it+4);
    end

    [clusid,clusters]=kmeans(concat,numclus);

    clusscr=zeros(numclus,3);



    %Set temp repeater locations
    for i=1:numclus,
        reptemp(i,1)=clusters(i,1);
        reptemp(i,2)=clusters(i,2);
    end
    %Calculate score of each cluster and order by score
    for i=1:numclus,
        clusscr(i,1)=reptemp(i,1);
        clusscr(i,2)=reptemp(i,2);
        for j=1:numuncov,
            if sqrt((reptemp(i,1)-concat(j,1))^2+(reptemp(i,2)-concat(j,2))^2)≤sepdist
                clusscr(i,3)=clusscr(i,3)+1;
            end
        end
    end

    srtclus = sortrows(clusscr,-3); %sorted clusters

    %Set new target location
    targetloc(1)=srtclus(1,1);
    targetloc(2)=srtclus(1,2);
    repdist=zeros(repno,1);
    for i=1:repno,
        repdist(i,1)=sqrt((targetloc(1)-rep(i,1))^2+(targetloc(2)-rep(i,2))^2);
    end

    [C,I]=min(repdist);
    lastplaced=I(1);
    newrepdist = sqrt((rep(lastplaced,1)-targetloc(1))^2+(rep(lastplaced,2)-targetloc(2))^


    if ((rep(lastplaced,1)≥targetloc(1))&&(rep(lastplaced,2)≤targetloc(2)))
        newrepang= acos((targetloc(1)-rep(lastplaced,1))/newrepdist);
    elseif ((rep(lastplaced,1)≥targetloc(1))&&(rep(lastplaced,2)≥targetloc(2)))
        newrepang=2*pi-acos((targetloc(1)-rep(lastplaced,1))/newrepdist);
    elseif ((rep(lastplaced,1)≤targetloc(1))&&(rep(lastplaced,2)≥targetloc(2)))
        newrepang=2*pi-acos((targetloc(1)-rep(lastplaced,1))/newrepdist);
    else
        newrepang=acos((targetloc(1)-rep(lastplaced,1))/newrepdist);
    end
```

```

    disp('connected'),disp(connected);


end

disp('number uncovered'),disp(numuncov);


%%Plot Data
xplot=zeros(repno,1);
yplot=zeros(repno,1);
for i=1:repno,
    xplot(i)=rep(i,1);
    yplot(i)=rep(i,2);
end



%%
concat = zeros(length(newx),2);
for i=1:length(newx);
    concat(i,1) = newx(i);
    concat(i,2) = newy(i);
end

concat=horzcat(newx,newy);
[clusid,clusters]=kmeans(concat,numlongdist);

cluslongscr=zeros(numclus,2);

for i=1:length(clusters),
    for j=1:length(newx)
        if sqrt((clusters(i,1)-newx(j))^2+(clusters(i,2)-newy(j))^2)≤sepdist;
            cluslongscr(i,1)=i;
            cluslongscr(i,2)=cluslongscr(i,2)+1;
        end
    end
end

cluslongsrt=sortrows(cluslongscr,-2);
plplots=zeros(actlongdist,2);

for i=1:actlongdist,
    plplots(i,1)=clusters(cluslongsrt(i),1);
    plplots(i,2)=clusters(cluslongsrt(i),2);
end

plrep=zeros(100,2);

longdistmat=zeros(actlongdist,2);
currentlong=1;
plrep(1,1)=plplots(1,1);
plrep(1,2)=plplots(1,2);
```

```
plrep(1,3)=1;

for j=1:actlongdist,
    longdistmat(j,1)=j;
    longdistmat(j,2)=sqrt((plrep(1,1)-plplots(j,1))^2+(plrep(1,2)-plplots(j,2))^2);
end


ldsort=sortrows(longdistmat,2);
tar(1,1)=plplots(ldsort(2,1),1);
tar(1,2)=plplots(ldsort(2,1),2);
prevtarget=ldsort(2,1);
d2tar=longdistmat(ldsort(2,1),2);

if ((plrep(1,1)≥tar(1,1))&&(plrep(1,2)≤tar(1,2)))
    angletotarget= acos((tar(1,1)-plrep(1,1))/d2tar);
elseif ((plrep(1,1)≥tar(1,1))&&(plrep(1,2)≥tar(1,2)))
    angletotarget=2*pi-acos((tar(1,1)-plrep(1,1))/d2tar);
elseif ((plrep(1,1)≤tar(1,1))&&(plrep(1,2)≥tar(1,2)))
    angletotarget=2*pi-acos((tar(1,1)-plrep(1,1))/d2tar);
else
    angletotarget=acos((tar(1,1)-plrep(1,1))/d2tar);
end

while d2tar>plsepdist,
    currentlong=currentlong+1;
    plrep(currentlong,1)=plrep(currentlong-1,1)+plsepdist*cos(angletotarget);
    plrep(currentlong,2)=plrep(currentlong-1,2)+plsepdist*sin(angletotarget);
    plrep(currentlong,3)=1;
    d2tar=sqrt((plrep(currentlong,1)-tar(1,1))^2+(plrep(currentlong,2)-tar(1,2))^2);
end

for i=1:actlongdist,
    newzones(i,1)=plplots(i,1);
    newzones(i,2)=plplots(i,2);
end
newzones(prevtarget,:)=[];
newzones(1,:)=[];

counter=1;
while counter<actlongdist-1;
    counter=counter+1;

    longdistmat=zeros(actlongdist-counter,2);
    for i=1:actlongdist-counter,
        longdistmat(i,1)=i;
        longdistmat(i,2)=sqrt((plrep(currentlong,1)-newzones(i,1))^2+(plrep(currentlong,2)-
    end

    ldsort=sortrows(longdistmat,2);

    prevtarget=ldsort(1,1);
    tar(1,1)=newzones(prevtarget,1);
    tar(1,2)=newzones(prevtarget,2);
```

```
    d2tar=longdistmat(prevtarget,2);


    if ((plrep(currentlong,1)≥tar(1,1))&&(plrep(currentlong,2)≤tar(1,2)))
        angletotarget= acos((tar(1,1)-plrep(currentlong,1))/d2tar);
    elseif ((plrep(currentlong,1)≥tar(1,1))&&(plrep(currentlong,2)≥tar(1,2)))
        angletotarget=2*pi-acos((tar(1,1)-plrep(currentlong,1))/d2tar);
    elseif ((plrep(currentlong,1)≤tar(1,1))&&(plrep(currentlong,2)≥tar(1,2)))
        angletotarget=2*pi-acos((tar(1,1)-plrep(currentlong,1))/d2tar);
    else
        angletotarget=acos((tar(1,1)-plrep(currentlong,1))/d2tar);
    end

    while d2tar>plsepdist,
        currentlong=currentlong+1;
        plrep(currentlong,1)=plrep(currentlong-1,1)+plsepdist*cos(angletotarget);
        plrep(currentlong,2)=plrep(currentlong-1,2)+plsepdist*sin(angletotarget);
        plrep(currentlong,3)=1;
        d2tar=sqrt((plrep(currentlong,1)-tar(1,1))^2+(plrep(currentlong,2)-tar(1,2))^2);
    end

    newzones(prevtarget,:)=[];


end


noplrep=currentlong;
openchannels=openchannels+numberoflongdistancelines*perline;
deficit=max(connected-openchannels,0);

indexnum=0;
indexnum2=0;
channelnumber=numberoflongdistancelines;
plcolor(1:currentlong,1)=51;
plcolor(1:currentlong,2)=51;
plcolor(1:currentlong,3)=255;
pluncovered=horzcat(newx,newy);
numuncovered=length(newx);
while deficit>0
   channelnumber=channelnumber+1;
   currentlong=currentlong+1;
   indexnum=indexnum+1;
   if indexnum==actlongdist
      indexnum=1;
      concat = zeros(numuncovered,2);
      for i=1:numuncovered;
          concat(i,1) = pluncovered(i,1);
          concat(i,2) = pluncovered(i,2);
      end

      [clusid,clusters]=kmeans(pluncovered,numlongdist);

```

```
      cluslongscr=zeros(numlongdist,2);

      for i=1:length(clusters),
            for j=1:length(pluncovered)
                if sqrt((clusters(i,1)-pluncovered(j,1))^2+(clusters(i,2)-pluncovered(j,2))
                    cluslongscr(i,1)=i;
                    cluslongscr(i,2)=cluslongscr(i,2)+1;
                end
            end
      end

        cluslongsrt=sortrows(cluslongscr,-2);
        plplots=zeros(actlongdist,2);

        for i=1:actlongdist,
            plplots(i,1)=clusters(cluslongsrt(i),1);
            plplots(i,2)=clusters(cluslongsrt(i),2);
        end
   end
   plrep(currentlong,1)=plplots(indexnum,1);
   plrep(currentlong,2)=plplots(indexnum,2);
   plrep(currentlong,3)=channelnumber;
   index1=1;
   for i=1:numuncovered,
      if sqrt((plrep(currentlong,1)-pluncovered(i))^2+(plrep(currentlong,2)-pluncovered(i))
          pluncovered(index1,1)=pluncovered(i,1);
          pluncovered(index1,2)=pluncovered(i,2);
          index1=index1+1;
      else
          numuncovered=numuncovered-1;
      end
   end
   plcolor(currentlong,1)=255-indexnum2*20;
   plcolor(currentlong,2)=40+indexnum2*20;
   plcolor(currentlong,3)=0;
   noplrep=noplrep+1;
   openchannels=openchannels+perline;
   deficit=max(connected-openchannels,0);
   indexnum2=indexnum2+1;
end

plrepplot=zeros(currentlong,2);
for i=1:currentlong,
    plrepplot(i,1)=plrep(i,1);
    plrepplot(i,2)=plrep(i,2);
end


%}
%%
plot(newx,newy,'s','MarkerSize',2,'MarkerEdgeColor',[1 1 1]);

hold on
xlabel('Miles');
```

```
ylabel('Miles');
title('Random User Data');
hold on

plot(xplot,yplot,'.','MarkerFaceColor','none','MarkerEdgeColor',[1 0 0],'MarkerSize',20,'Li
hold on

%}

%{
plot(plplots(:,1),plplots(:,2),'o','MarkerFaceColor','g','MarkerEdgeColor',g,'MarkerSize',1
%}

for i=1:noplrep,
    plot(plrepplot(i,1),plrepplot(i,2),'.','MarkerFaceColor','g','MarkerEdgeColor','g','Ma
    hold on
end


 for i=1:repno,
    h=rep(i,1); k=rep(i,2); r=10; N=256;
    t=(0:N)*2*pi/N;
    plot(r*cos(t)+h,r*sin(t)+k,'Color','r','LineStyle','--');
    hold on
 end




  for i=1:repno,
    h=rep(i,1); k=rep(i,2); r=10; N=256;
    t=(0:N)*2*pi/N;
    plot(r*cos(t)+h,r*sin(t)+k,'Color',[43/255 129/255 86/255],'LineStyle',':');
    hold on
  end
%}
 %{
 for i=1:noplrep,
    h=plrep(i,1); k=plrep(i,2); r=15; N=256;
    t=(0:N)*2*pi/N;
    plot(r*cos(t)+h,r*sin(t)+k,'Color',[plcolor(i,1)/255 plcolor(i,2)/255 plcolor(i,3)/255
    hold on
 end



 for i=1:currentlong,
     if i==7,
        text(plrep(i,1)-2,plrep(i,2),num2str(plrep(i,3)));
     elseif i==8,
         text(plrep(i,1)+.5,plrep(i,2),num2str(plrep(i,3)));
     elseif i==12,
     else
         text(plrep(i,1)+.5,plrep(i,2),num2str(plrep(i,3)));
```

```
    end
 end
%}


%{
h=plot(clusters(:,1),clusters(:,2),'v',...
     'MarkerSize',10,'LineWidth',2,'MarkerEdgeColor','k');
axis([-40 40 -40 40]);
set(h, 'Color', [.8 .8 .8]);
get(h);
  %}

axis([-40 40 -40 40]);
axis('square')
hold off
disp('number of repeaters'),disp(repno);

disp('number of PL repeaters'),disp(noplrep);
disp('number of channels'),disp(openchannels);
```