Optimizing Network Resources for Transmitting Video on Public Safety LTE Networks

First Responders Group

March 2015

Homeland Security

Science and Technology

Intentionally Blank

Public Safety Communications Technical Report

Optimizing Network Resources for Transmitting Video on Public Safety LTE Networks

DHS-TR-PSC-14-01
March 2015

Reported for: **The Office for Interoperability and Compatibility by the Department of Commerce's the Public Safety Communications Research Program**

Homeland Security
Science and Technology

Publication Notice

Disclaimer

Contact Information

Please send comments or questions to: SandTFRG@HQ.DHS.GOV

Intentionally Blank

Contents

Figures

Tables

Intentionally Blank

Abbreviations

Acronym	Definition
3GPP	3rd Generation Partnership Project
CIF	Common Intermediate Format (352 x 288 pixels)
CP	Cyclic Prefix
DHS	Department of Homeland Security
DL	Downlink
DoC	Department of Commerce
eNodeB	Evolved Node B
E-UTRAN	Evolved UMTS Terrestrial Radio Access Network
FirstNet	First Responder Network Authority
ICM	Iterated Conditional Modes
IP	Internet Protocol
ISI	Inter Symbol Interference
ITS	Institute for Telecommunication Sciences
LTE	Long Term Evolution
MCMC	Markov Chain Monte Carlo
NTIA	National Telecommunications and Information Administration
OFDM	Orthogonal Frequency-Division Multiplexing

Acronym	Definition
OIC	Office for Interoperability and Compatibility
PSCR	Public Safety Communications Research Program
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RB	Resource Block
RE	Resource Element
TTI	Transmission Time Interval
UE	User Equipment
UL	Uplink
UMTS	Universal Mobile Telecommunications System
VGA	Video Graphics Array (640 x 480 pixels)

Abstract

This report examines the problem of how to allocate limited network resources in a public safety Long Term Evolution (LTE) network for disseminating video streams to end users. We developed a mathematical model of the network environment and defined a new metric called **usefulness** to represent the overall value the network resources are providing to the users. We incorporate the concept of local control for public safety practitioners into our model through the use of a customizable utility function and simulate its performance. We then apply known optimization techniques and algorithms to our model and analyze the results.

Key words: LTE, Resource Scheduling, Video

1 Introduction

The roll out of public safety Long Term Evolution (LTE) wireless networks will have a profound effect on how public safety personnel perform their missions. Current network technologies only support voice, although some networks have been updated to provide low-bandwidth data services to User Equipment (UE). The new LTE network's support for high-bandwidth applications, such as video streaming, will fundamentally transform the way public safety practitioners do their jobs. Yet, inefficient use of the spectrum can quickly limit their potential. Due to the dynamic and complex nature of the radio spectrum, scheduling who gets which resource, when they get it, and for what duration, is a non-trivial task. The eNodeB in the LTE network architecture is tasked with analyzing the current state of the network and making these scheduling decisions.

Currently, such algorithms are developed by the eNodeB manufacturers themselves and kept as proprietary technology. This limits our understanding of how they evaluate the network and our ability to analyze their usefulness in a public safety network. While we have no reason to doubt that they are optimized for commercial telecommunications, the requirements of public safety could possibly cause inefficiencies within the algorithms. However, the black box nature of the algorithms makes it impossible to truly perform such an analysis.

We plan to develop a LTE resource scheduler that is optimized for the requirements of a public safety LTE network. Specifically, we address the issue of delivering video streams to public safety practitioners. Our work is broken into two main efforts: 1) identifying an efficient optimization algorithm for our network model, and 2) extending our model to support local control through the adaptation of its utility function.

1.1 Background

Long Term Evolution (LTE) is an IP based, pure packet-switched network, and as such, no central controller is required. Instead, it uses an Evolved Packet Core (EPC) for its core network. The EPC transports data in packets without ever establishing a physical connection via dedicated circuits (3GPP n.d.). The network consists of a base station, an eNodeB, UE and the EPC. Being configured in this manner allows for decreased connection and handover times over previous cellular techniques. Additionally, communication and decisions between UE and base stations are increased.

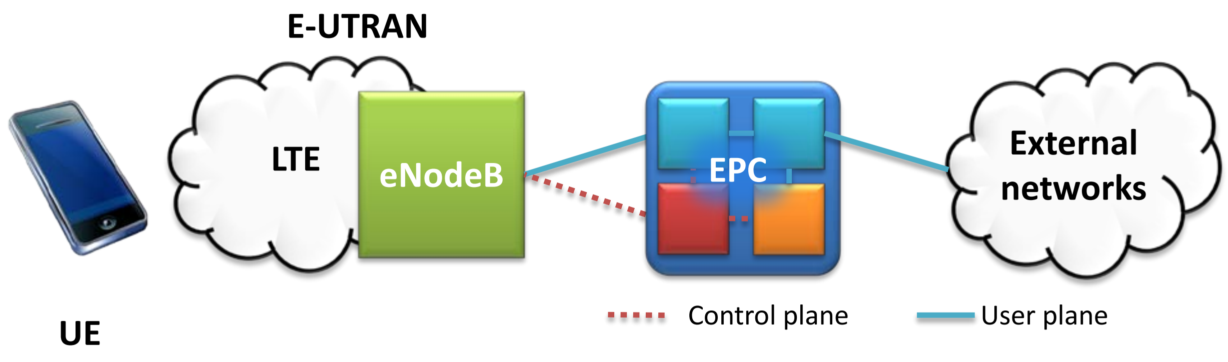


Figure 1: A simplified diagram of the LTE network environment.

LTE incorporates an adaptive scheduler into the eNodeB, which enables the rapid adjustment and efficient utilization of network resources within its quality of service (QoS) (E. 3GPP 2012) requirements, both in the uplink (UL) and downlink phase (DL). One component of this is the transmission time Interval (TTI), which for LTE is only one millisecond (ms). Decreasing the TTI in LTE further improved system latency, helping to meet one of the design criteria. A principle benefit of the packet switched network is the ability to schedule user(s) in both time and frequency dimensions. Both of these dimensions are described in the following sections. The scheduler allocates network resources, in the form of resource blocks (RBs), defined as 12 consecutive subcarriers for a single frequency slot. RBs are allocated to users by giving the UE a specific amount of sub carriers (frequency domain) for a predetermined amount of time (time domain) (E. 3GPP 2012).

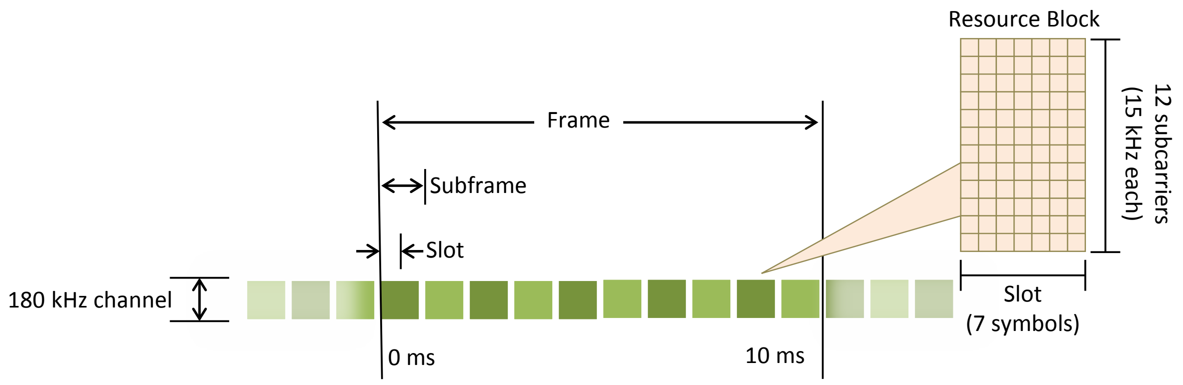


Figure 2: A simple depiction of the relationship between frequency and time highlighting a normal CP resource block and a single 180 kHz channel.

1.1.1 Time Domain

The LTE frame is 10 milliseconds long, divided into 10 sub frames, each 1.0 millisecond long, which are further divided again into 0.5 millisecond duration slots. Each slot is made up of Orthogonal Frequency-Division Multiplexing (OFDM) symbols and, depending on the mode, uses either a seven-symbol normal cyclic prefix (CP) or a six-symbol extended CP. The CP is used as a guard period at the beginning of each OFDM symbol, to negate any potential Inter Symbol Interference (ISI) resulting from a high-rate data stream being transmitted serially.

1.1.2 Frequency Domain

In the frequency domain, RBs are grouped into 12 subcarriers, each 15 kHz wide and occupying a total of 180 kHz. The data carrier and smallest element in LTE is the resource element (RE), which is one subcarrier × one symbol. This means that one RB is 12 subcarriers × 7 slots = 84 REs with a normal CP, or 72 REs with an extended CP. However, not all REs are used for data service. Some are reserved for signaling information, which is a unique feature of LTE among packet-switched networks that traditionally use a physical layer preamble. Symbol coding is performed on the Res. Depending on the channel parameters, symbols can be coded with QPSK (2 bits/symbol), 16QAM (4 bits/symbol) or 64QAM (6 bits/symbol). The number of resource blocks available depends on the available bandwidth, but goes from 6-100 PRBs as bandwidth increases. As network conditions allow, multiple bandwidths are available: 1.4MHz, 3 MHz, 5MHz, 10MHz, 15MHz or 20MHz (public safety will use a 10MHz bandwidth for FirstNet). The result of this architecture is that a 10 MHz LTE channel consists of 50 x 2000 = 100,000 RB/sec that can be scheduled to requesting UEs, although any RBs assigned to any given user must be consecutive in the frequency domain.

2 Network Model

In order to optimize the network resources of a LTE network, we created a mathematical model of the environment and its nodes. A node is defined as the endpoint of a video stream from the base station to a UE. For this work, a node is synonymous with a user, but that does not need be the case. A user can request two video streams, in which case one UE is modeled in our network as containing two nodes (one for each video stream). For consistency with any future work, we use the terminology node. The inputs of our model are as follows:

- **Current network resource allocation**. The resource allocation is the result of a mapping of all RBs in the network to a specific requesting node. There are many different ways the RBs can be mapped, with our goal being to identify the specific mapping that provides the most value to the network users.
- **Network state of each node**. Every node can experience different wireless channel characteristics between itself and the eNodeB due to local interference, attenuation, refraction, etc. This results in either: a) a particular LTE channel not being available to them for data transfer, or b) the necessity of using an encoding scheme that is more error resistant (such as QPSK vs QAM64), thus resulting in a lower bitrate.
- **Video stream characteristics**. The characteristics of the video stream, such as lighting, motion and target size, have a direct effect on the bitrate required to provide the user with a given utility. We discuss this in detail in Section 2.2.

Our model is then used to compute the sum of the utility of each video stream $u(n, f, v)$, weighted by the requesting node's priority $p(n)$. This value is referred to as the network's *usefulness* and is shown in Equation (1). Usefulness is a unit-less metric that gives a relative measure of the value provided by a given resource allocation.

$$U(f) = \sum_{\mathcal{N}}^{n} u(n, f, v) \cdot p(n) \qquad (1)$$

In a typical optimization effort, a cost function is defined for the system and optimization occurs by minimizing the system's cost. In our scenario, we want to maximize the usefulness that the LTE network can provide to the users, i.e., maximize the value provided by the resource blocks based on how they are allocated. We thus use $U(f)$ as our cost function for the network and we attempt to optimize the network by identifying a resource allocation f that maximizes $U(f)$.

2.1 Priority

An important issue for public safety practitioners in the field is the concept of "local control" over their network. In an emergency event, local personnel will have the best understanding of what the network needs to support and the goals to be accomplished. The performance of the scheduling algorithm can be improved by harnessing this information and incorporating it into its calculations, thus allowing the algorithm the flexibility to generate an incident-specific optimized resource allocation, as opposed to being a rigid algorithm that optimizes for the general case.

Node priority allows public safety practitioners to provide such input. When an emergency event occurs and public safety personnel arrive on the scene, each public safety worker has a different role that may require different demands on network services. For example, a firefighter inside a burning building will require a higher guarantee of network resources than a policeman cordoning off the incident area. By assigning priorities to first responders based on inputs from public safety personnel on-site, the algorithm can assign network resources to where they are best utilized.

To illustrate a generic case, if we plot Nodes vs. Resources within a resource constrained environment, sorting the nodes from highest priority to lowest, we can visualize the impact of priority. In Figure 3(a), we see that since all nodes have the same priority, every node is allocated the same number of network resources – in essence, every node has an equal reduction in the number of resources they receive. However, if we were to uniformly assign priority values to nodes and re-plot the results (still sorting nodes by priority), we get Figure 3(b); higher priority nodes receive their full resource requests at the expense of lower priority nodes.

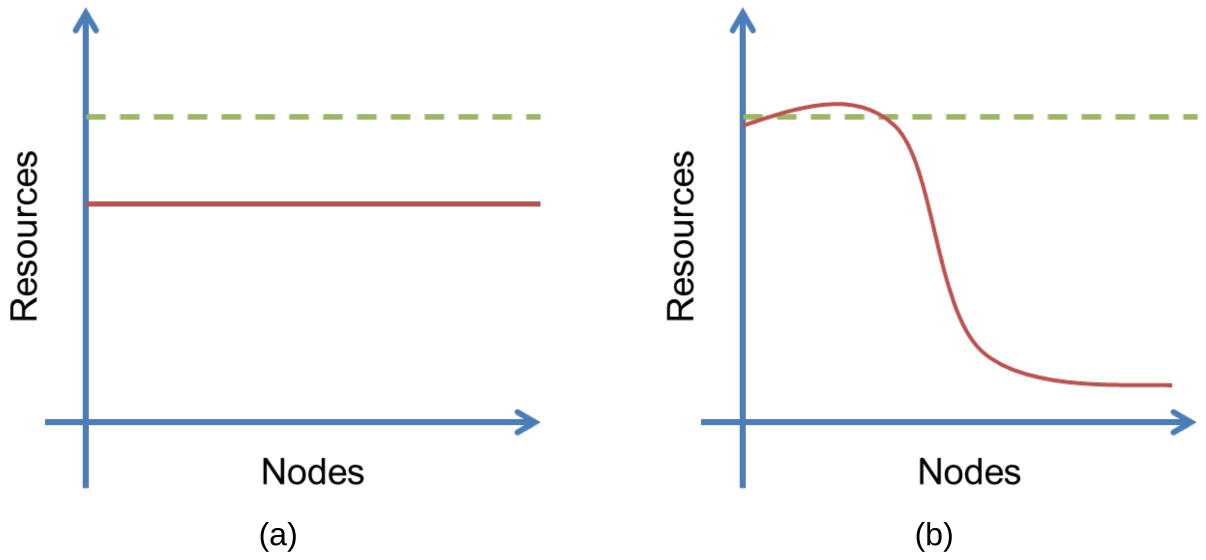


Figure 3: Effect of priority on resource allocation. (a) All nodes assigned same priority. (b) Nodes assigned different priorities (sorted by priority).

Priority thus allows us to consider resources assigned to more important nodes with greater value to the network. Nodes are thus assigned a priority value, in the form of an integer, with one representing the highest priority. Multiple nodes can be assigned the same priority value.

We define a simple priority function that allows each node to be assigned its own priority, within the model, allowing for fine-grained definition of users' priority during network optimization (for simplification, we limit to integer values). In general, a node with priority 1 will receive ten times the resources as an identical node with priority 10, although there are many other internal algorithmic parameters and conditions that play into resource assignment. To accomplish this, we use the priority function defined in Equation (2), representing an inverse relationship between the magnitude of the priority value and its effect on the user (i.e., smaller values are of higher priority).

$$p(n) = \frac{1}{priority} \tag{2}$$

2.2 Utility

The utility function, $u(n, f, v)$ of Equation (1) is *the probability that a user can successfully identify the object in the scene*. This was defined and studied in (U.S. Department of Homeland Security 2010) through subjective public safety practitioner experiments. This work, along with other previous research (U.S. Department of Homeland Security 2011) (U.S. Department of Homeland Security 2012) (Dumke, Ford and Stange 2011), showed that the content and scene of a video stream has a major effect on how a video is perceived at a given bitrate. Thus, in order to successfully optimize a network's resources, we must take into account the properties of the video streams themselves.

In (U.S. Department of Homeland Security 2010), the researchers used the concept of a hypothetical reference circuit to categorize video clips based on three selected properties:

- **Lighting**: The lighting conditions of the scene, defined as: bright, dim or variable.
- **Motion**: The motion of the object within the scene, defined as: high or low.
- **Target Size**: The relative size of the target of interest in the video: large or small.

From these, they generated generalized use cases and presented public safety practitioners' video clips during subjective testing, asking them to identity a target object in the scene. With the resulting data, they generated a set of recommendations and curves showing the relationship between bitrate and object identification under fixed conditions, such as scene lighting, motion and target size.

We assume that each video stream's properties are known *a priori*, through either real-time analysis at the eNodeB or via some other back-end system. The generalized form of the utility function $u(n, f, v)$ is presented in Equation (3), with $b(n, f)$ representing the bitrate allocated to the node under the current network configuration and $v(n)$ representing the video stream's properties.

$$u(n, f, v) = u(b(n, f), v(n)) \tag{3}$$

We also define the concept of minimum utility. Minimum utility is defined as a lower acceptable limit on utility that the network should provide to users. If this minimum cannot be met with available resources, the algorithm should attempt to assign those resources to another node which can achieve the minimum utility. For example, instead of having two users both with poor utility and both resulting in unusable video streams, the network will favor having one user with high utility and one with low/none (of course, details such as node priority and others will determine which user gets high utility and which gets low).

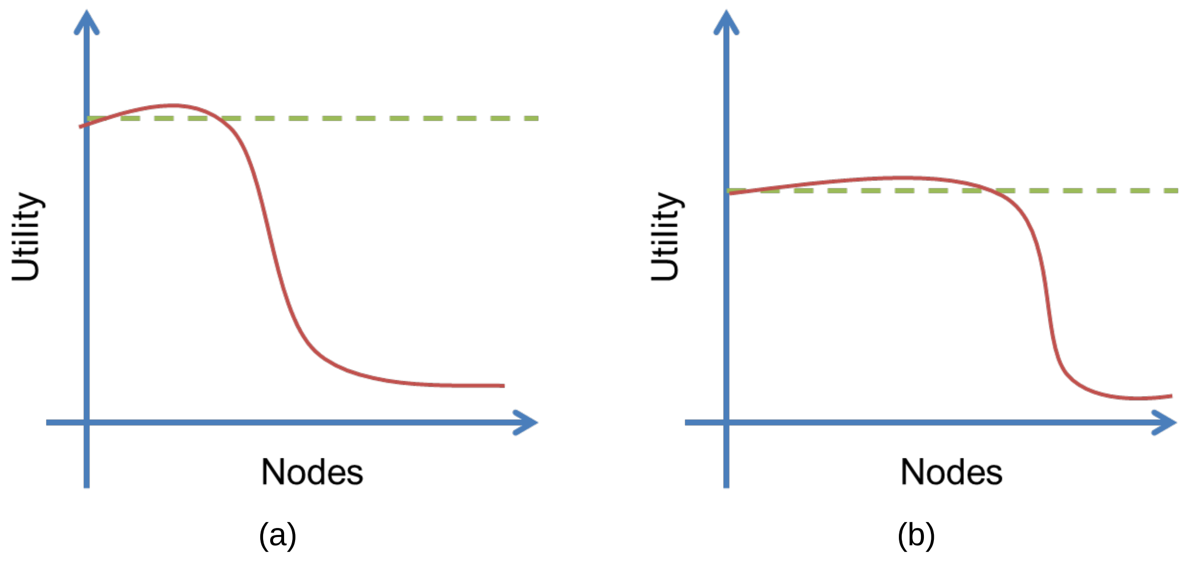


Figure 4: Effects of minimum utility (dashed line) on resource allocation. (a) High minimum utility and (b) Low minimum utility.

Similar to how we visualized priority in Figure 3, we can visualize the effects of minimum utility in the generic case. Assume that we plot Nodes vs Utility (nodes sorted with higher priority to the left). Figures 4(a) and 4(b) show the effects of altering the minimum utility for network.

As can be observed, a higher minimum utility, represented by the dashed line, requires more network resources being assigned to higher priority nodes. In turn, this reduces the number of nodes that are able to achieve this level of utility in a resource constrained environment. Lowering the minimum utility reduces the number of resources required to achieve this level of utility, thus allowing more nodes to achieve it. In essence, the minimum utility input affects the peak level of resource allocation. Lowering the peak level allows the algorithm to allocate resources such that more users can achieve it.

2.2.1 Utility Functions

We define four utility functions for investigation. The first is the experimental utility function, shown in Figure 5(a). This is pulled directly from our previous work (U.S. Department of Homeland Security 2010) corresponding to a video stream with bright lighting, a large target size and low motion (each combination of parameters has its own curve). Technically, this utility function does not meet our desire of accepting a minimum utility value, since these curves are fixed and cannot be customized to meet local public safety needs. However, we include them for completeness since all subsequent utility functions will inherit from this dataset.

The step utility function, shown in Figure 5(b), is a union of the experimental utility function with a step function aligned at the minimum bitrate. Any allocated bitrate that is less than the minimum bitrate returns a utility value of zero. For allocated bitrates greater than the minimum bitrate, the utility value returned is the same as generated from the experimental utility function.

The third utility function is a modification of the step utility function. Referred to as the ramp utility function and shown in Figure 5(c), this utility function removes the presence of the horizontal portion of the curve. We postulate that a requested bitrate within this horizontal range may have no "incentive" to either move towards greater levels of utility, or to relinquish its network allocation. We thus selected an arbitrary value of 10 percent of the minimum bitrate's corresponding utility and replaced the horizontal portion with a ramp function peaking at this value.

Lastly, we define the sigmoid utility function, shown in Figure 5(d). The first three utility functions presented were all piecewise-linear. We investigated a single, continuous utility function, that did not having a constant tangent over any interval. To accomplish this, we used a sigmoid curve, which like the more general logistics curve, has a well-recognized 'S'-shape. The benefits of using a sigmoid curve are that it naturally results in non-negative numbers for all values (in its general form) and has a range of $(0, 1)$, which maps directly to the possible ranges of utility. For our model, we perform a mathematical translation to center the sigmoid curve on the minimum bitrate.

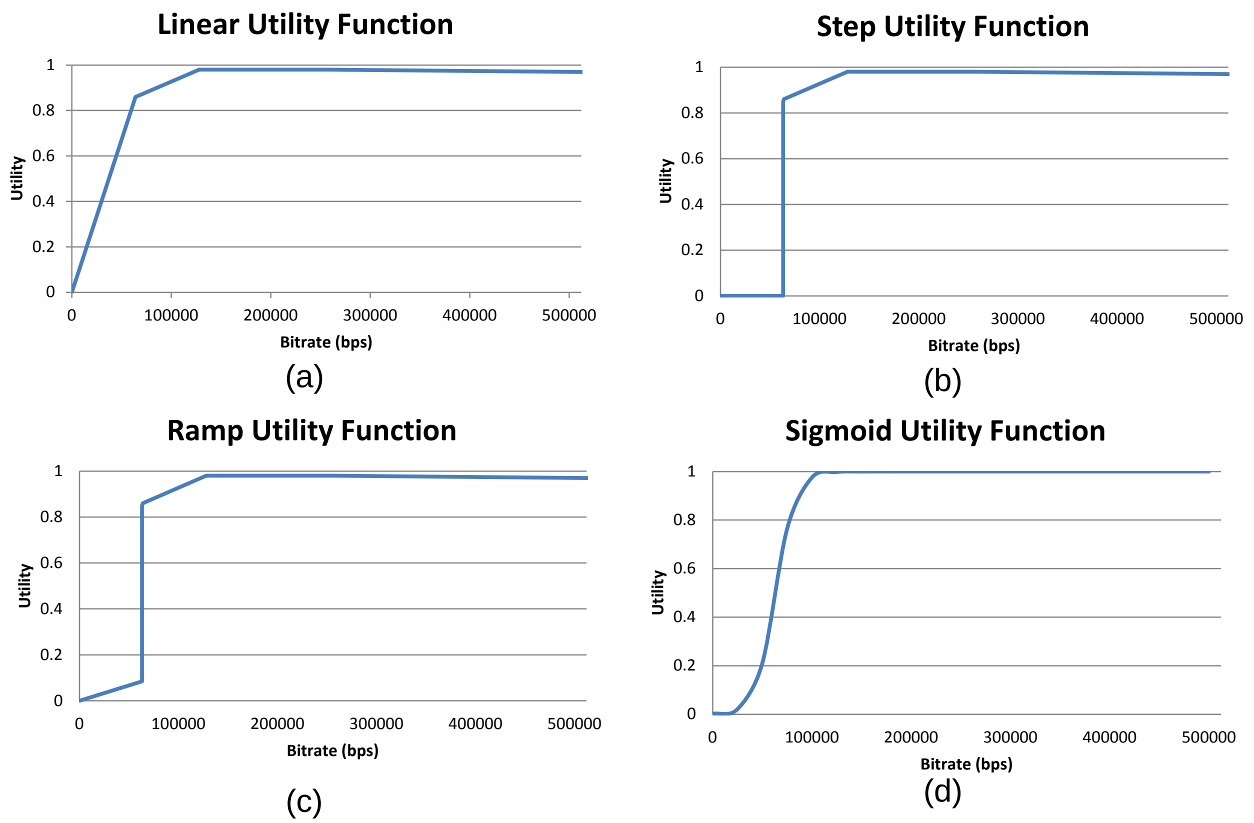


Figure 5: Utility functions- (a) Experimental, (b) Step, (c) Ramp and (d) Sigmoid.

3 Algorithms

With our network model defined, we borrow established optimization algorithms (Iterated Conditional Modes and the Metropolis-Hastings Algorithm) and concepts (Simulated Annealing) from image processing, showing how they can be applied directly to this problem space with minimum alterations. These algorithms allow us to search a highly multi-dimensional solutions space in an efficient manner for an optimal resource allocation for the network scheduler. We present the three algorithms we investigated in the following section.

3.1 Iterated Conditional Modes

Iterated Conditional Modes (ICM) is a deterministic algorithm that guarantees convergence to a local maximum. Proposed by Besag (Besag 1986) for the application of noise reduction in image processing, ICM sequentially maximizes local conditional probabilities through an iterative approach in which each pixel is maximized and conditioned on its neighbors. It uses local conditional probabilities on the assumptions that neighboring pixels tend to have similar values within an image and noise is injected into pixels independently of their neighbors. Let y be the noisy image of the uncorrupted image x, with $x_{i,j}^{k}$ representing the estimation of pixel (i,j) at iteration k. ICM computes $x_{i,j}^{k+1} | x_{i\pm1,j\pm1}^{k}$ until $x_{i,j}^{k+1} = x_{i,j}^{k}$.

We map ICM onto our problem-space in a straight-forward manner. The algorithm iterates over a network of nodes (analogous to a network of image pixels), with the special case that every node in our network is a neighbor of all other nodes. The neighborhood system is an assumption to make image processing feasible. We relax this assumption here as there are relatively few users. From this, we derive Algorithm 1, which continually reallocates network resources, selecting a new resource allocation, until it finds the one that maximizes the network's usefulness.

Algorithm 1 Iterated Conditional Modes

```
1:  Initialize: $f \leftarrow InitialResourceAllocation()$
2:  $U_{max} = U(f)$
3:  Do
4:    $reallocationOccured = FALSE$
5:    for all $n_i \in \mathcal{N}$ do
6:      for all subchannel $s$ in $n_i.subchannels$ do
7:        for all $n_j \in \mathcal{N}, n_i \neq n_j$ do
8:          $f^* \leftarrow n_i[s].Deallocate(), n_j[s].Allocate()$
9:          $U_{temp} = U(f^*)$
10:         if $U_{temp} \leq U_{max}$ then
11:           $f \leftarrow n_i[s].Allocate(), n_j[s].Deallocate()$
12:           Continue
13:         else
14:           $U_{max} = U_{temp}$; $reallocationOccured = TRUE$
15:         end if
16:       end for
17:     end for
18:   end for
19: while ($reallocationOccured$)
```

```
20: return $f$
```

3.2 Metropolis-Hastings Algorithm

The second optimization algorithm we look at is the Metropolis-Hastings Algorithm, first proposed by Metropolis (Metropolis, et al. 1953) and later extended into the general case by Hastings (Hastings 1970). This algorithm is a Markov chain Monte Carlo (MCMC) method which can be useful in obtaining a solution where the solution space consists of a large multi-dimensional distribution. By selecting a sequence of samples, the algorithm can generate a distribution that closely approximates the desired distribution, with increasing numbers of samples improving the desired accuracy. Non-deterministic by nature, the algorithm accomplishes this through a random walk along the multi-dimensional distribution, selecting a new point x_{i+1} based on a probability density $Q(x_{i+1}|x_i)$, in which $Q(x_{i+1}|x_i)$ must be symmetric, i.e., $Q(x|y) = Q(y|x)$, and is usually represented as a Gaussian. The standard deviation σ of the Gaussian used in the sampler is a variable in our model. We will examine the effects it produces when presenting our results.

Whereas ICM is a deterministic algorithm and converged to a local maximum (which is not guaranteed to be the global maximum), the Metropolis-Hastings Algorithm has the ability to reject a new point, even if that point appears to improve the desired solution, through the use of an acceptance ratio α, which is a ratio of the current usefulness to the previous iterations usefulness. Thus, through properly-defined variables, the algorithm can remove itself from a local maximum in search of the global maximum.

Additionally, whereas ICM operates over discrete units (resource blocks), the Metropolis-Hastings Algorithm operates on a continuing spectrum of values. This allows us to work with the more precise duty cycle of the node, providing the added benefit of presenting a more accurate description of the physical world. Algorithm 2 presents the Metropolis-Hastings Algorithm as applied to the problem-space.

Algorithm 2 Metropolis Hastings Algorithm

```
1:  Initialize: $f_0 \leftarrow InitialResourceAllocation()$
2:  $t = 0, U_t = U(f_t)$
3:  while $t < t_{max}$ do
4:      $f_t^* \leftarrow Sample(f_t)$
5:      $U_t^* = U(f_t^*)$
6:      if $U_t^* > U_t$ then
7:          $f_{t+1} \leftarrow f_t^*$
8:      Else
9:          $\alpha_t = \left(\frac{U_t^*}{U_t}\right)$
10:         $f_{t+1} \leftarrow f_t^*$ with probability of $(1 - \alpha_t)$
11:     end if
12:     $t = t + 1$
13: end while
14: return $f_{t_{max}}$
15: Procedure Sample(f)
16: for all $n_i \in f$ do
17:     for all subchannel $s$ in $n_i.subchannels$ do
18:         $f^* \leftarrow n_i[s].Allocate(rand(0, \sigma))$
19:     end for
20: end for
21: return $f^*$
```

3.2.1 Solution Space and Invalid Solutions

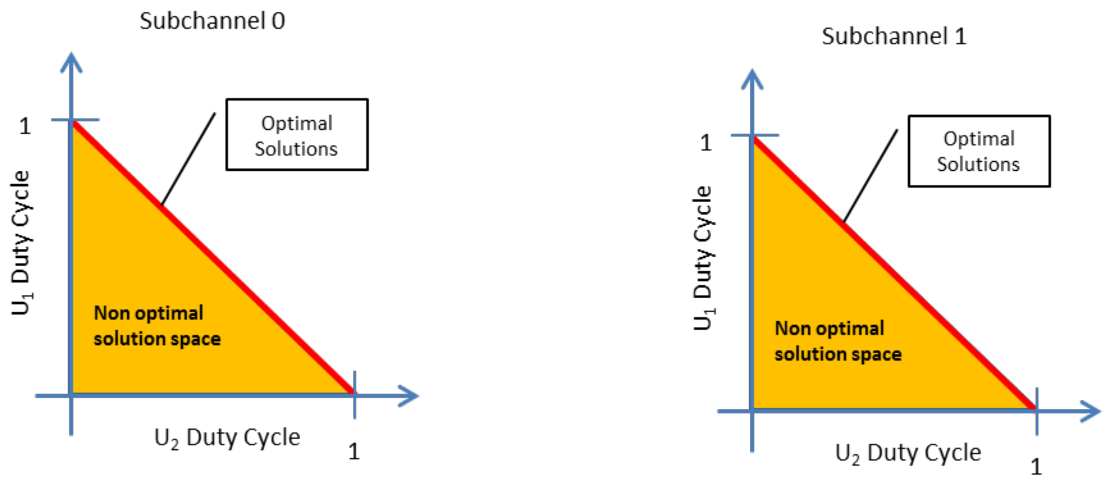


Figure 6: Simplified view of non-convex solution space- (a) Subchannel 0 and (b) Subchannel 1.

Before continuing, it is worth considering the solution space on which the algorithm will operate. While we know that the solution space is highly multi-dimensional, due to the 50 independent subchannels that make up the wireless communication medium, upon simple examination, it can further be shown that the solution space is non-convex. Take a simple example of an environment with two users of equal priority and only two subchannels. Since the total duty cycle allocated between all users in a given subchannel cannot exceed 1, the solution space for a specific subchannel is bounded by the line $d_1 + d_2 = 1$, with d_i being the duty cycle of node n_i. (It is straightforward that duty cycle cannot be negative, and thus the axes serve as the other bounds for the solution space.)

In this example, shown in Figure 6, the region shaded is the complete solutions space. Hence, we have an $|\mathcal{N}|$-dimensional solution space for each subchannel. Additionally, since the allocation of an unallocated resource to a node can never cause the overall utility to decrease, i.e., a resource can never have a negative utility; the line marking $d_1 + d_2 = 1$, can be described as the optimal solution space, in that we expect to see the solution that maximizes the usefulness to reside on that line for each subchannel. This gives an optimum solution space of dimension $|\mathcal{N} - 1|$ for each subchannel.

Since the Metropolis-Hastings Algorithm randomly selects a new point based on a normal distribution, it is possible (and highly probable) that an invalid point will be selected when the current point is near the bounds of the solution space. An invalid point is a point that falls outside of the solution space, such as a subchannel that has total duty cycle greater than one, or a duty cycle for a subchannel for a particular node is negative. While one could have the Metropolis-Hastings Algorithm select another point, this could cause performance issues if the current point sits in a location where a larger majority of the points near it are invalid. To alleviate this, we project the invalid value back onto the normal of the solution surface to arrive at a valid point.

Additionally, we combine the knowledge of the solution space with the ability to project points to gain an additional optimization to our algorithm. Since we know that all maximized solutions must fall on the optimum solution space, we can force the simulator to project all selected points, valid or invalid, to the optimum solution space. We expect that such a behavior may result in performance increases, as the simulator avoids "walking around" within the non-optimal solution area. However, we acknowledge that if local minimums or maximums exist within the surface, forcing the projection may inhibit the algorithm from the possibility of escaping them. We look for such behavior when analyzing the results.

3.3 Simulated Annealing

Simulated annealing (Kirkpatrick, Gelatt Jr and Vecchi 1983) (Cerny 1985) (Geman and Geman 1985) is the idea of slowly reducing the probability of accepting a worse solution given your current state. Inspired by the thermodynamic concept of annealing, simulated annealing can be applied to allow one to identify an approximation of a global maximum without requiring an exhaustive search. The Metropolis-Hastings Algorithm lends itself nicely to this approach, as it allows for a non-zero probability of accepting a worse result at any given sample.

Remember that in the Metropolis-Hastings Algorithm, the probability density function $Q(x|y)$ must be symmetric and is usually a Gaussian distribution. We look to gain efficiencies applying simulated annealing to the sampler within our network optimization problem. To model this, we map the simulated annealing concept of temperature to the standard deviation σ of the sampler's distribution.

During the execution of the Metropolis-Hastings Algorithm, $\sigma \to 0$ as the algorithm iterates. Since standard deviation must be a positive, non-zero number, we define σ such that it reduces asymptotically to zero through an exponential decay model. Thus we define the standard deviation σ_t at a given time iteration t as (4),

$$\sigma_t = \sigma_0 e^{-c\left(\frac{t}{t_{max}}\right)} \tag{4}$$

with σ_0 being the initial standard deviation and t_{max} being the total time allotted for algorithm execution, in iterations. The constant c defines the reduction in σ_0 from t_0 to t_{max} as (5).

$$\frac{\sigma_0}{\sigma_{t_{max}}} = \frac{1}{e^{-c}} = e^{c} \tag{5}$$

We adjust the acceptance probability computation in the original Metropolis-Hastings Algorithm so that the above holds true. This is accomplished by raising α to σ.

Applying this to the Metropolis-Hastings Algorithm now yields Algorithm 3. The initial standard deviation σ_0 and the scaling constant c are both parameters to the algorithm and the effects they have on the final resource allocation are presented in our results. Since Algorithm 3 is derived from Algorithm 2, we again apply the same behavior with regard to the solution space and correcting for the selection of invalid solutions, described in Section Solution Space and Invalid Solutions.

Algorithm 3 Metropolis Hastings Algorithm with Simulated Annealing

```
1:  Initialize: $f_0 \leftarrow InitialResourceAllocation()$
2:  $t = 0, U_t = U(f_t)$
3:  while $t < t_{max}$ do
4:      $\sigma_t = \sigma_o e^{-c\left(\frac{t}{t_{max}}\right)}$
5:      $f_t^* \leftarrow Sample(f_t, \sigma_t)$
6:      $U_t^* = U(f_t^*)$
7:      if $U_t^* > U_t$ then
8:          $f_{t+1} \leftarrow f_t^*$
9:      Else
10:         $\alpha_t = \left(\frac{u_t^*}{u_t}\right)^{\sigma_t}$
11:         $f_{t+1} \leftarrow f_t^*$ with $p(1 - \alpha_t)$
12:     end if
13:     $t = t + 1$
14: end while
15: return $f_{t_{max}}$
16: Procedure Sample(f, $\sigma$)
17: for all $n_i \in f$ do
18:     for all subchannel $s$ in $n_i.subchannels$ do
19:         $f^* \leftarrow n_i[s].Allocate(rand(0, \sigma))$
20:     end for
21: end for
22: return $f^*$
```

4 Simulator

In order to compare our three optimization algorithms, we developed our own discrete time simulator to analyze their performance. For a baseline measurement, and since we do not have knowledge of a current eNodeB's scheduling algorithm, we implemented a fourth scheduler based on a greedy algorithm. This greedy scheduler is a simple algorithm which behaves similar to TCP, in which each node tries to use as many resources as possible with disregard to other nodes in the network. No attempt is made in this greedy scheduler to ensure all resource blocks are assigned.

To maintain consistency between algorithm results in order to perform proper comparisons, we always define our network's initial resource allocation in resource blocks. When optimizing with Metro (Metropolis-Hastings Algorithm) or Anneal (Metropolis-Hasting Algorithm with Simulated Annealing), we insert an initialization step to convert the number of assigned resource blocks to a node into its corresponding duty cycle.

We analyze our scheduling algorithms in two different network environments. We begin by comparing their performance in an ideal network environment in which all nodes have equal access to every subchannel. This provides a look at how the schedulers behave in ideal conditions.

In the second part of our simulations, we analyze how the scheduling algorithms behave in increasingly degraded networks conditions. We run multiple simulations with the percentage of the subchannels available to the UEs, varying from 100 percent of subchannels available to 50 percent. This is performed by generating network environments in which every UE independently has a fixed percentage of their subchannels defined as unavailable, with the available subchannels randomized among the UEs. We generated 25 such network condition files for each data point, from 100 percent to 50 percent, with increments of 10 percent. For deterministic scheduling algorithms, we ran each of the 25 network conditions once and averaged their results. For non-deterministic algorithms, we used the same method as in the ideal network conditions and ran each network definition files 25 times, averaging the results, for a total of 625 measurements (25 network definitions each simulated 25 times).

4.1 Assumptions

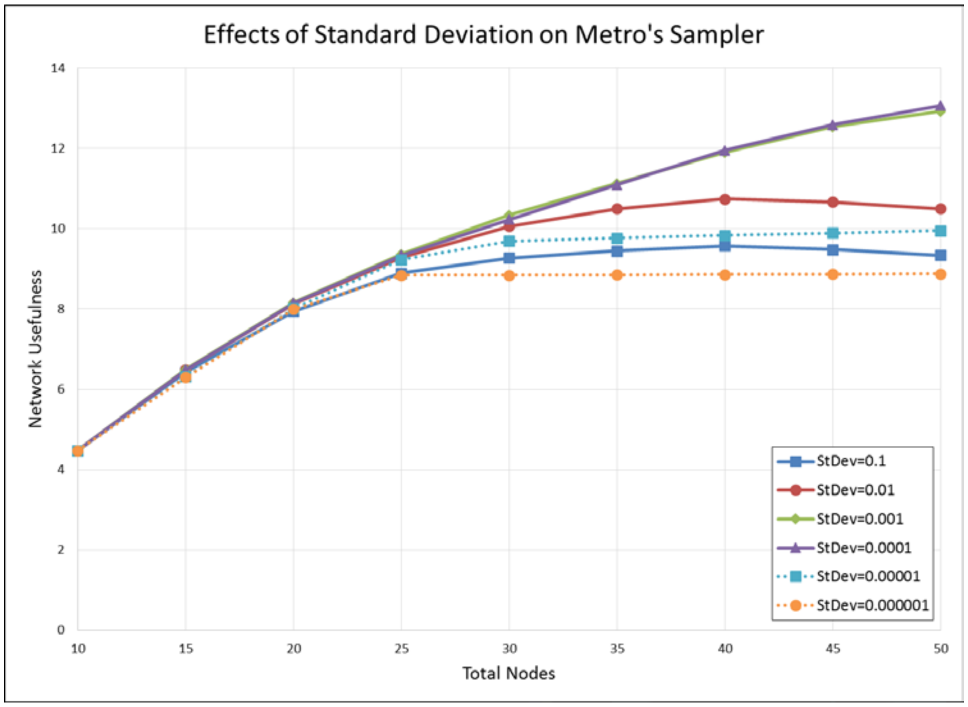
We assumed that a LTE subchannel carries 200 RBs per second, as opposed to 2000 RBs per second. This reduces the number of nodes required to create a congested network environment, while not affecting the relative performance of any of the scheduling algorithms with respect to each other. This allows us to shorten the duration of each simulation, since in implementing our algorithms, we did not focus on computational optimization, such as through implementing paralleling, and thus we shy away from making any statements of the type in our findings. Likewise, we assumed that all nodes modulated their signals using QPSK and with Extended CP Mode - again, to minimize simulation durations.

Lastly, we assumed all video was delivered in VGA format, and that all video stream properties are Bright lighting conditions, Low motion and Large target size, unless otherwise stated. We want to focus on the algorithms themselves and choose not to look at the complex relationships between algorithms, channel availability, and video quality and characteristics. Such investigations are left to future work.

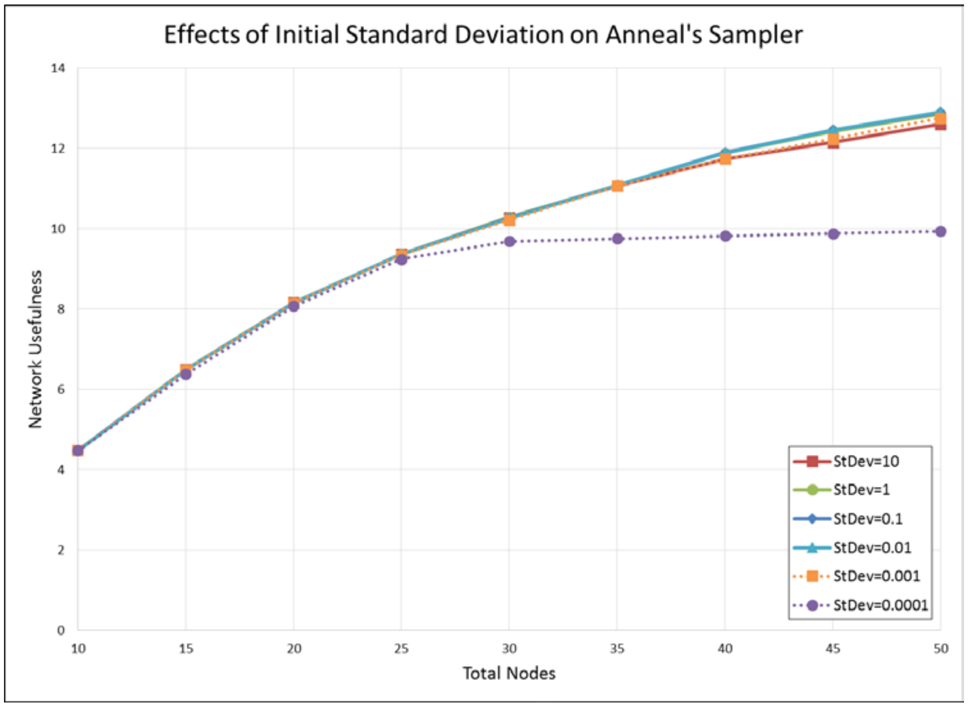
5 Results

5.1 Baseline Analysis

We begin by performing a baseline analysis on our proposed scheduling algorithms. We use an ideal network environment in which every node has all of their subchannels available for transmitting and receiving data.



(a)



(b)

Figure 7: Effects of the Sampler's Standard Deviation on Network Usefulness for (a) Metro and (b) Anneal.

The Metro algorithm requires the selection of a standard deviation for its sampler. We tested values from 0.1 to 0.000001 in logarithmic intervals. The results are shown in Figure 7(a). We see that, initially, as we decrease the standard deviation, we get improved overall performance with respect to network usefulness. As the number of nodes increase in the network, a sampler with a large standard deviation is unable to identify an optimum solution. The large standard deviation reduces the likelihood that the sampler is able to select a solution within the increasingly smaller target area where the optimum solution resides. Decreasing the standard deviation increases the resolution of the sampler.

However, if the sampler's standard deviation is selected too small, it suffers from the opposite problem that a large standard deviation suffers from. In this case, the small standard deviation causes the sampler to become stuck in a local area of the solution space since it cannot move across the solution space quickly. This could be overcome by increasing the number of iterations until the algorithm terminates – but at a (possibly large) performance penalty. We selected a standard deviation value of $\sigma = 0.0001$ for use in the Metro scheduling algorithm for the remainder of our results.

The Anneal algorithm also requires the selection of a standard deviation for its sampler. We tested and plotted values from 10 to 0.0001 in logarithmic intervals. The results are shown in Figure 7(b). The Anneal algorithm is more robust against the value of the standard deviation used due to its simulated annealing property. This allows the standard deviation to decrease over time. However, careful inspection of the data shows that it also exhibits behaviors similar to that of Metro – although in the inverse.

Unlike Metro, in which a smaller standard deviation increases the usefulness, the inverse is true with Anneal. While this may seem counter-intuitive at first, an in-depth look at the trace data produced by the algorithm explains why this is. The Anneal algorithm is based on the Metropolis-Hastings Algorithm, but uses the concept of annealing in its sampler. This means that given a beginning standard deviation of σ, the sampler will reduce $\sigma \rightarrow 0$ as $t \rightarrow \infty$, controlled by the scaling constant ($c = 10$ in our simulations) during the execution of the algorithm by means of an exponential decay model. What we are observing is that while decreasing values of standard deviation allow for better results, as shown in the analysis of Metro, there not only exists diminishing returns. Once the standard deviation becomes small enough, it negatively impacts the algorithm's performance because it essentially gets stuck in a local area, with the sampler too small to ever move out of it. Thus starting with a larger standard deviation, while decreasing its value during execution, allows Anneal to avoid this issue.

Similarly, if the initial standard deviation is too large, there might not exist enough time for the algorithm to converge before it terminates. This is why we see less than optimal results when we tested with large standard deviation values, such as 10. Based on these results, we selected a standard deviation value of $\sigma = 0.1$ for use in the Anneal scheduling algorithm for the remainder of our results.

With the standard deviation selected for the Anneal and Metro, we can now perform a baseline comparison of all the scheduling algorithms, shown in Figure 8. We included a Greedy algorithm for comparison. All algorithms, with respect to total network usefulness, have relatively similar performance. Since all nodes have exactly the same network characteristics, the only thing distinguishing them from each other is their priority. Thus algorithms all default to a priority-based allocation similar to what Greedy does by default.

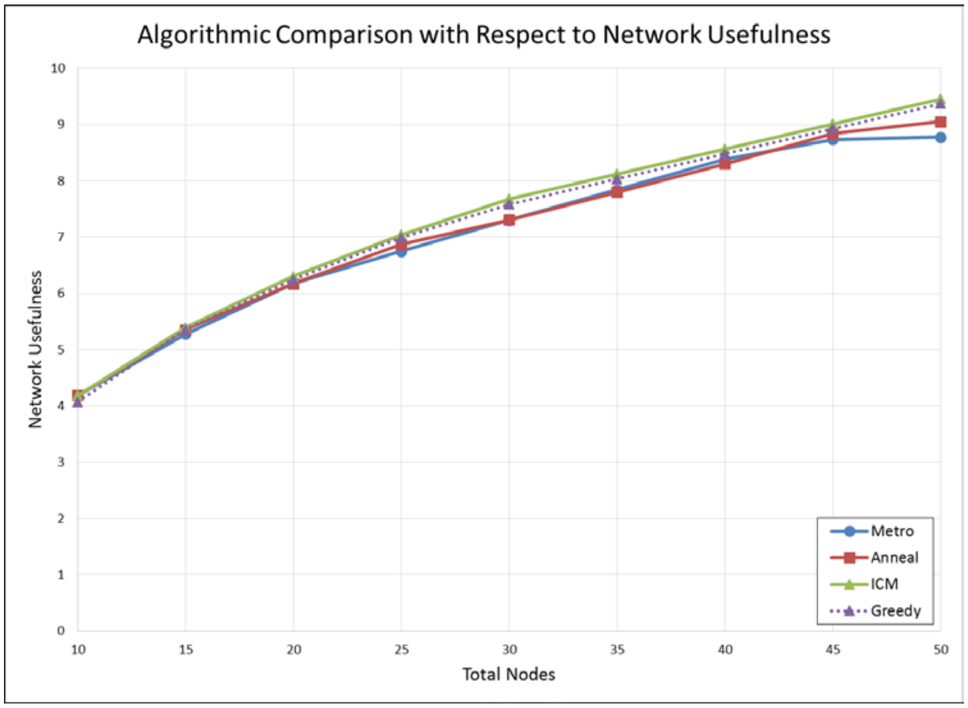


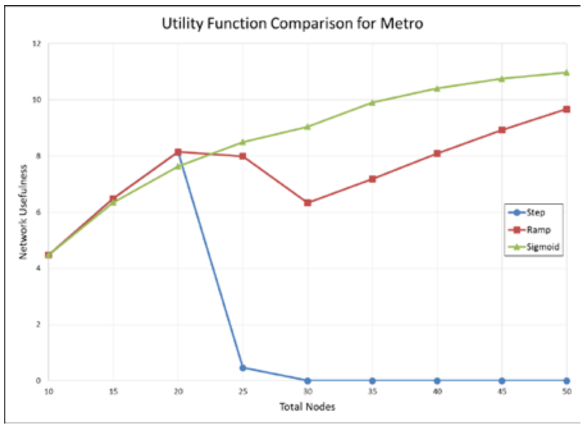
Figure 8: Comparison of Scheduling Algorithms in an Ideal Network Environment

5.1.1 Computational Performance

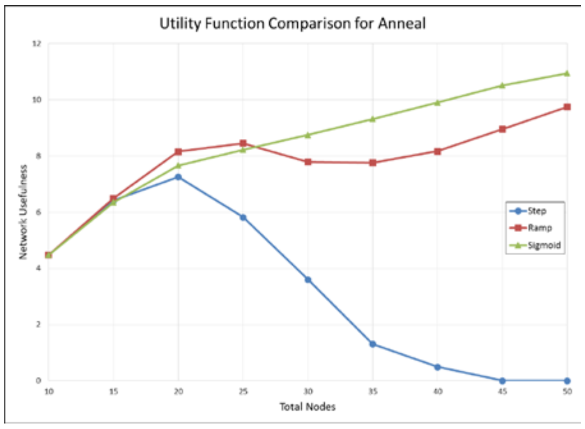
We discussed previously that we shy away from giving time-based results to the computation of each algorithm, as we did not optimize our algorithms for computational performance. However, we can analyze the algorithms themselves to glean information. The ICM scheduler iterates through all nodes, restarting its execution for each re-allocation of a resource block. Thus, the algorithm's performance will scale linearly.

Both Metro and Anneal, however, execute for a fixed duration before terminating. Although increasing the number of nodes increases the computation of the sampler (though increasing the dimensions of the solution space to select from), point selection is independent between solution space dimensions (other than the final step of verifying its validity and mapping back). This means that both Metro and Anneal have constant complexity.

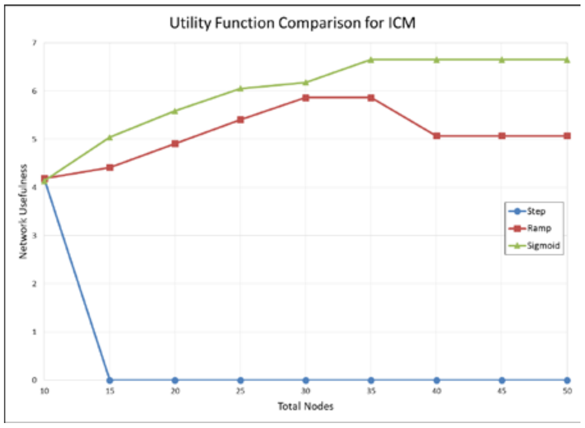
5.2 Utility Functions



(a)



(b)



(c)

Figure 9: Network usefulness performance of utility functions for (a) Metro, (b) Anneal and (c) ICM.

Each of our scheduling algorithms relies on a utility function, which maps the allocated bitrate to the utility of the video the node is receiving. We presented three utility functions (Step, Ramp and Sigmoid) which were based upon the results of subjective tests of public safety practitioners (U.S. Department of Homeland Security 2010). We plot how each utility function maximizes network usefulness in our uniform network environment for Metro, Anneal and ICM – Figure 9(a), Figure 9(b) and Figure 9(c) respectively.

The sigmoid utility function shows the best overall performance for all scheduling algorithms. Both the ramp and step utility functions appear to perform identically to the sigmoid in environments with lower resource constraints (less nodes requesting resources). However, as resource competition increases, the performance of the step utility function falls dramatically, while the ramp function incurs a less dramatic decrease. Additionally, the performance of Anneal is more robust to the step and utility functions.

Through examining the trace files, both of these behaviors can be explained. Remember that the initial environment for this simulation was uniform resource allocation to all nodes in the network. If the environment is large enough, then the initial starting bitrate of each node will be less than the critical bitrate of the utility function, which in these simulations occurred when $|\mathcal{N}| = 25$.

When starting below where the critical bitrate occurs, the outcome is largely dependent on the algorithm used. In Metro, the sampler selects a new point on the overall solutions space using a fixed normal distribution, which corresponds to a new resource allocation for the network (and $f' \in \mathcal{F}, f' \neq f$). Due to the small standard deviation of the distribution (which we previously showed increased the overall performance of Metro), the likelihood of the newly selected distribution containing a node with a resource allocation greater than the critical bitrate is small (and decreases while $|\mathcal{N}|$ increases). With the step utility function being horizontal below the critical bitrate, giving all nodes a fixed utility regardless of any reallocation of network resources, we are left with $U(f') = U(f) = 0$, and thus account for the steep drop-off in utility. The Anneal algorithm is able to mitigate this effect somewhat because the standard deviation of its sampler begins at a much larger value, increasing the probability of it allocating a node with enough network resources to get it over the critical bitrate. This is why we see the gradual decrease in overall network usefulness as the number of nodes increases: the difference between the starting bitrate and the critical bitrate grows larger and outside the range of probabilistically being selected by the sampler (in the Metro case, due to the small standard deviation, this for all practical circumstances occurs immediately). However, the Anneal sampler's standard deviation decreases with time, gradually falling into the same issue that Metro encounters. The ramp utility function's behavior suffers from similar effects as the step function, although not as pronounced due to the non-horizontal natural of the function below the critical bitrate.

5.3 Resource Distribution by Priority

An important feature of optimizing the network resources is the algorithm's ability to, in situations of high resource contention, maximize the usefulness of each individual network resource. Remember that in our ideal network environment, the only feature that differentiates nodes is their priority. Thus, as contention increases, lower priority nodes have their resources allocated to higher priority ones. We examine how each algorithm performs in three different network conditions: low contention ($|\mathcal{N}| = 10$), medium contention ($|\mathcal{N}| = 35$) and high contention ($|\mathcal{N}| = 50$). Figure 11(a) and Figure 11(b) show the allocation by node priority of usefulness and utility respectively. Looking closely, ICM does the best job at cutting off lower priority nodes from being assigned resources as resource competition increase. In fact, ICM successfully drives all lower priority nodes to zero. Metro and Anneal do a good job at removing resources from lower priority nodes as resource contention increases.

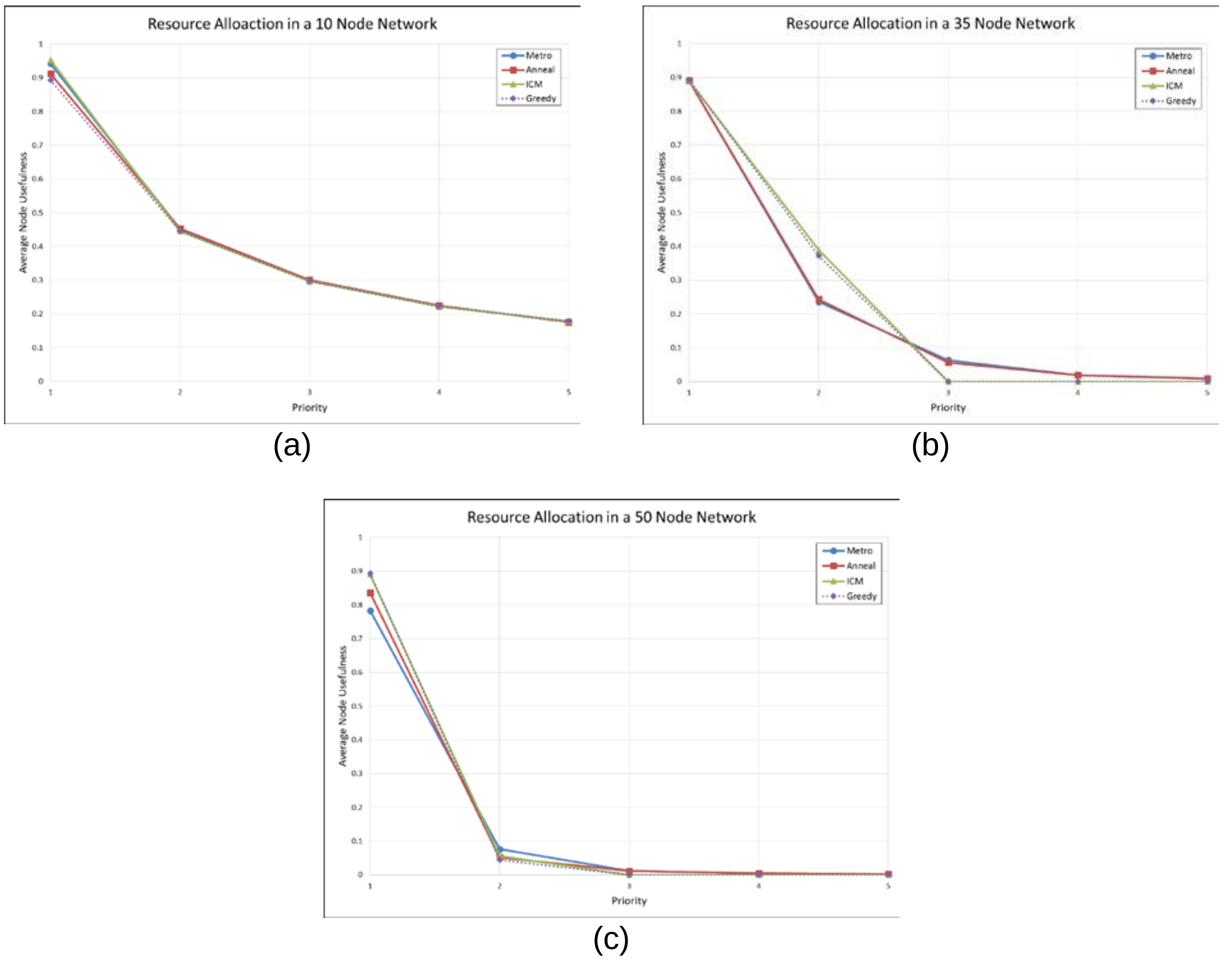


Figure 10: Resource allocation by priority for a network with (a) low resource contention, (b) medium resource contention and (c) high resource contention.

5.3.1 Stopping Criteria

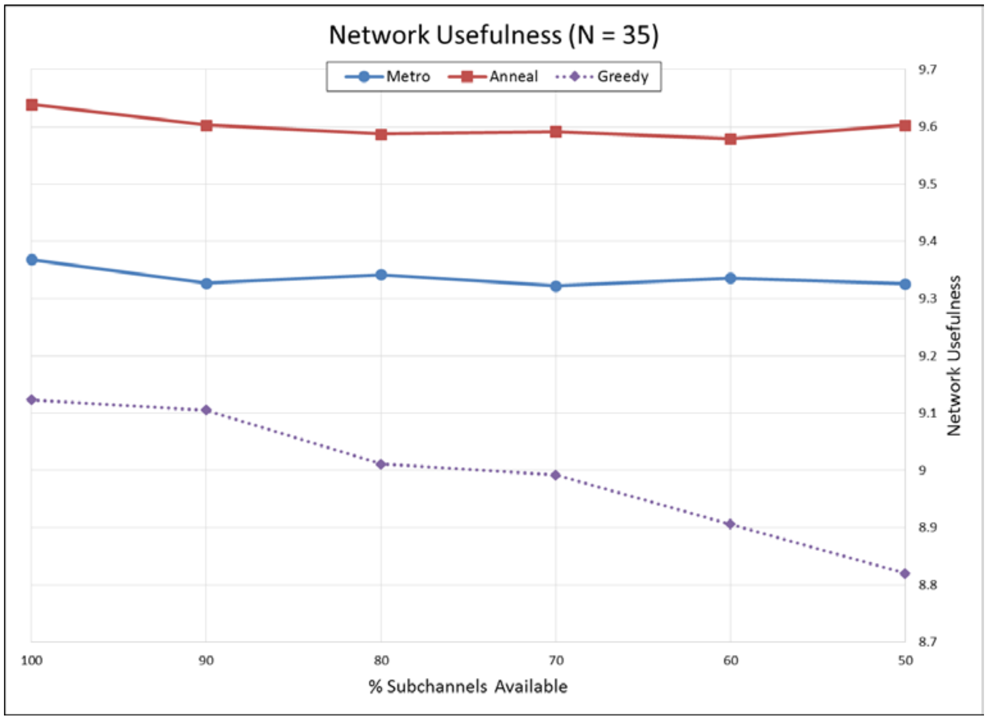
While Anneal and Metro perform slightly worse in higher contention environments, this can be compensated by increasing the number of iterations the algorithms execute before terminating. Remember that Anneal and Metro are iterative algorithms that do not have a defined halting criteria – they execute for a defined number of iterations then halt. ICM, on the other hand, halts once it identifies a solution that cannot be improved by reallocating a resource block from one node to another. This means that increasing the number of iterations for Anneal and Metro improves the results of the solution they identify – albeit at diminishing returns.

5.4 Network Degradation

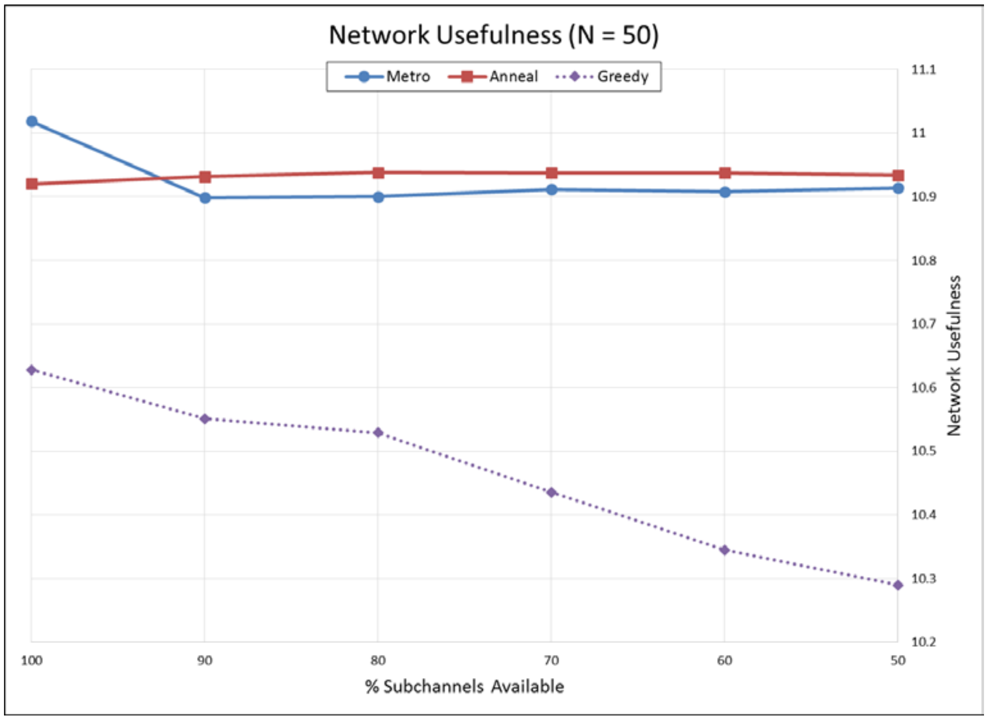
We now look to test the scheduling algorithms' robustness to network degradation. Up until this point, all nodes in our network environment were permitted to receive data on any subchannel (ideal network conditions). In this subsection, we gradually degrade the network conditions by randomly disabling a percentage of each node's subchannels. In addition, we compare two different network sizes: 35 nodes and 50 nodes. Note that even though we used these network sizes in the previous subsection as a way of looking at the effects of network contention, these two subsections are not directly comparable since we are controlling network contention by both network size and subchannel availability (previous section only used network size). As before, the priority of the nodes ranges from 1 to 5 and is uniformly distributed.

Looking at network usefulness, we see that in both the 35 node network (Figure 12) and the 50 node network (Figure 12), Metro and Anneal are robust against network degradation. This is accomplished by driving lower priority nodes to zero resources and allocating them to maximize their use. We saw how in an ideal network, some resources were still being allocated to lower priority nodes. In this test, those resources are being better allocated and providing the scheduling algorithms' robustness. Greedy, for comparison, is unable to make smart choices on resource allocation and as the total number of network resources decreases, so does the network usefulness.

We can also look at the same situation except in terms of the average number of nodes receiving service. We say a node receives service if it is allocated enough resources such that its video stream has a utility of 0.85. As expected, Figure 12 shows that in both a 35 node and 50 node environment, Metro and Anneal are robust against network degradation, with Anneal providing better overall performance.

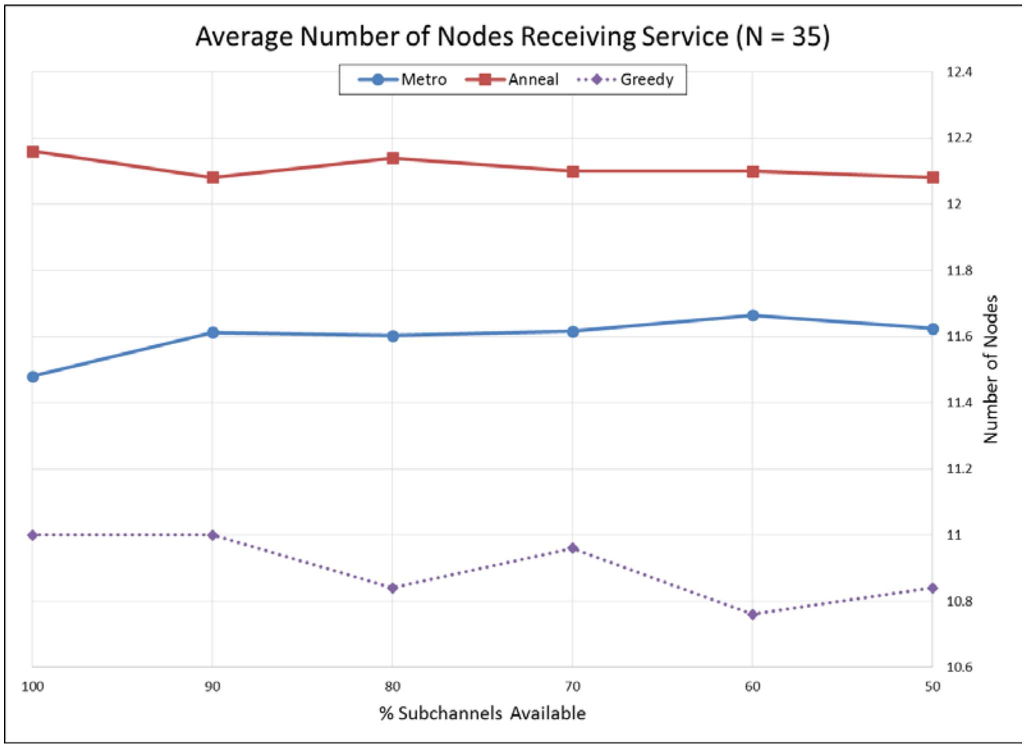


(a)

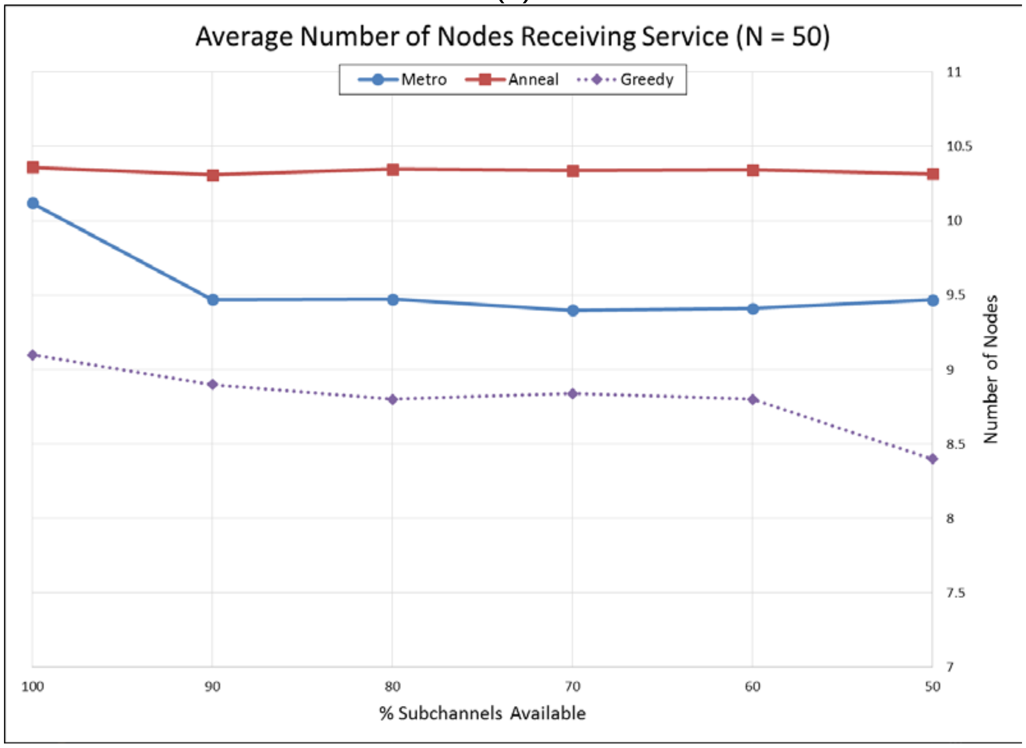


(b)

Figure 11: Network usefulness for varying degraded network environments- (a) 35 node network and (b) 50 node network.



(a)



(b)

Figure 12: Average number of nodes receiving service for varying degraded network environments- (a) 35 node network and (b) 50 node network.

5.5 Balancing Priority and Utility

Lastly, we test how well the scheduling algorithms balance the properties of priority and utility. In this test, we defined a 35 node network environment in which 25 percent of the nodes have Priority = 1 and are requesting a video stream (Dim/Moving/Large Target) in which a high bitrate is needed to provide high utility. The other 75 percent of the users have Priority = 2 and are requesting a video stream (Light/Static/Large Target) in which high utility can be achieved with a much lower bitrate. Thus we are left with a scenario in which a small set of high priority nodes want to consume all network resources and crowd out the much large set of lower priority nodes requesting minimal resources. As before, each data point is an average of 25 simulations.

We begin by looking at network usefulness, shown in Table 1. Metro, Anneal and ICM all greatly outperform the Greedy algorithm. This is because Greedy allocates resources based on priority alone, so the high priority nodes are allowed to crowd out the more numerous lower priority nodes. Metro and Anneal perform best because they compute that greater network usefulness is achieved by allocated resources to the lower priority nodes, thereby allowing more users to utilize the network.

Algorithm	Network Usefulness
Metro	7.68284
Anneal	6.70197
ICM	5.75250
Greedy	1.71347

Table 1: Network usefulness for different scheduling algorithms in a priority vs utility environment test

We can see this in better detail if we look at Table 2. Here, we see that using Greedy only allows two high bandwidth nodes on the network, crowding out all other nodes. Metro, on the other hand, allocates resources to on average 14.24 nodes of priority two. Yet, if one of those high priority nodes must receive network resources at all costs, then this can easily be achieved through the allocation of priorities to nodes, since usefulness is partially computed as a weighted product of utility and the inverse of priority.

Algorithm	Average Nodes
Metro	14.24
Anneal	13.52
ICM	12
Greedy	2

Table 2: Average number of nodes receiving 0.85 utility in a priority vs utility environment test

6 Conclusions

In this work, we looked at defining an algorithmic LTE resource scheduler that optimizes the usefulness of the network configuration. This began by developing a mathematical model of an LTE network and defining a new metric called usefulness. By computing the usefulness of a network's resource allocation, we can compare multiple resource allocations against each other in a quantitative manner. This results in the ability to apply known optimization algorithms to the problem LTE network resource scheduling in order to identify the resource allocation that maximizes the network's usefulness.

We presented three different optimization algorithms: Iterated Conditional Modes (ICM), the Metropolis-Hastings Algorithm (Metro) and the Metropolis-Hastings Algorithm with Simulated Annealing (Anneal). We showed that although ICM appears to perform slightly better in the general case, Metro and Anneal allow for better customization as well as contain scaling and performance benefits.

We incorporated the concept of local control, an important issue for public safety practitioners. Not only does this support their desired ability to control their own networks, we can leverage their insights during incident response by creating a flexible scheduling algorithm that optimizes to the specific situation at hand, instead of a rigid algorithm that can only be optimized for the general case – thus achieving better and more personalized results.

We defined four different utility functions to test with our scheduling algorithms. After simulating each with the scheduling algorithms, we concluded that using a sigmoid function allows for the best performance of the four. Its continuous, non-linear features are successful in allocating resources to the nodes where they are most useful.

Further testing of the scheduling algorithms showed that Metro and Anneal outperformed both ICM and Greedy. From a computational standpoint, both scale efficiently with network size. While most of the time Anneal performed better than Metro due to its annealing ability while searching the solution space, this was not definitive. In our test of priority vs utility, Metro was able to consistently achieve better results in terms of both network usefulness and average number of nodes receiving service. We suspect that further tuning of the multiple variables in Anneal may remove this discrepancy, but we leave this for a future effort.

6.1 Future Work

In the future, we look to further refine the sigmoid utility function to allow for even better customization. We recognize that there are still efficiencies to be gained, such as driving resource allocation to nodes below the defined minimum utility to zero. Additionally, we look to further study the effects of the multiple of parameters within the model and how they can be tuned, along with giving recommendations for certain defined scenarios.

Beyond this, we would like to implement our resource scheduling algorithms into a more robust and capable network simulator, such as ns3. With this, we could study more advanced topics, such as the effects of mobility and the transitions from transient to steady-state when users enter or leave a cell sector.

We have also received strong positive feedback from public safety personnel when we presented initial work at the Fifth Video Quality in Public Safety (VQiPS) Workshop in Houston, Texas. Included in this was allowing feedback from end users into the algorithm, instead of only relying on the backend system operations, utilizing upcoming multicast standards, and enhanced priority functions that contain multiple variables. More advanced suggestions included the ability to make our model network agnostic so that it could be applied to future technologies not based on LTE and defining advanced network metrics and visualization tools – although such requests may be beyond the scope of even future work at this point.

Appendix A References

3GPP. n.d. *www.3gpp.org.* http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core.

3GPP, ETSI. 2012. *Quality of Service (QoS) Concept and Architecture.* Technical Specification, Sophia Antpolis Cedex, France: ETSI 3GPP.

Besag, Julian. 1986. "On the Statistical Analysis of Dirty Pictures." *Journal of the Royal Statistical Society, Series B* 259-302.

Cerny, V. 1985. "Thermodynamical approach to the traveling salesman problem: An efficient simulated algorithm." *Journal of Optimization Theory and Applications* 41-51.

Dumke, Joel, Carolyn G Ford, and Irena W Stange. 2011. "The effects of scene characteristics, resolution, and compression on the ability to recognize objects in video." *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series.*

Geman, S, and D Geman. 1985. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 721-741.

Hastings, W K. 1970. "Monte Carlo Sampling Methods Using Markov Chains." *Biometrika* 97-109.

Kirkpatrick, S, C D Gelatt Jr, and M P Vecchi. 1983. "Optimization by Simulated Annealing." *Science* 671-680.

Metropolis, Nicholas, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. "Equations of State Calculations by Fast Computing Machines." *Journal of Chemical Physics* 1087-1092.

U.S. Department of Homeland Security. 2012. "Assessing Video Quality for Public Safety Applications Using Visual Acuity." Public Safety Communications Technical Report.

U.S. Department of Homeland Security. 2011. "Recorded-Video Quality Tests for Object Recognition Tasks." Public Safety Communications Technical Report.

U.S. Department of Homeland Security. 2010. "Video Quality Tests for Object Recognition Applications." Public Safety Communications Technical Report.

Appendix B Mathematical Details

In this appendix, we present the mathematical details describing the formulation of our network model. These were omitted from the main body of this report for the sake of readability, but provided here for completeness.

Define

- $\mathcal{N} = \{n_1, \dots, n_k\}$ as the set of all nodes in the network
- $\mathcal{R} = \{1, \dots, 50\} \times \{1, \dots, 2000\}$ as the set of all Resouce Blocks (RBs) within all 50 subchannels for a 1 second duration

Let $\mathcal{F}: \mathcal{R} \to \mathcal{N}$, with $f \in \mathcal{F}$ representing a specific network resource allocation.

Define $b(n, f)$ as the downstream bitrate that node n can receive under the network resource allocation f and described by Algorithm B.1

Algorithm B.1 Node Bitrate

```
1: $Given : n, f$
2: $bitrate = 0$
3: for all subchannel $s$ in $n.subchannels$ do
4:   if $s$ is available then
5:     $bitrate += f_{RB}^{s}(n) \cdot s.CPMode \cdot s.Modulation \cdot 12$
6:   end if
7: end for
8: return $bitrate$
```

With

- $f_{RB}^{s}(n)$: The number of resource blocks per second assigned to node n on subchannel s.
- $s.CPMode$: The number of symbols per slot; either 7 (Normal CP) or 6 (Extended CP).
- $s.Modulation$: The number of bits per symbol used in encoding data; either 2 (QPSK), 4 (16QAM), or 6 (64QAM).
- 12: The number of sub-carrier frequencies within a single subchannel, as defined in the LTE standard.

Let $U(f)$ be defined as a unit-less metric called *usefulness* and be composed of the weighted sum of the uility $u(n, f)$ provided by each node n under the current resource allocation f, and weighted against the node's priority **p(n)**.

$$U(f) = \sum_{\mathcal{N}}^{n} u(n, f) \cdot p(n)$$

With

$$p(n) = \frac{1}{priority}$$

Problem statement:

Given a set of all possible network resource allocations $\mathcal{F}:\mathcal{R}\rightarrow\mathcal{N}$, find $f \in F$ such that . $U(f) \geq U(f')\forall f' \in \mathcal{F}, f' \neq f$.

With regard to the utility functions, they can be described mathematically as:

Step Function

$$u_{step}(n,f,v) = \begin{cases} 0, & b_{alloc} < b_{critical} \\ u_{exp}(n,f,v), & b_{alloc} \geq b_{critical} \end{cases}$$

Ramp Function

$$u_{ramp}(n,f,v) = \begin{cases} 0.1\left(\frac{b_{alloc}}{b_{critial}}\right)u_{exp}(n,f,v), & b_{alloc} < b_{critical} \\ u_{exp}(n,f,v), & b_{alloc} \geq b_{critical} \end{cases}$$

Sigmoid Function

$$u_{sigmoid}(n,f,v) = \frac{1}{1 + e^{-\beta(b_{alloc}-b_{critical})}}$$

with $u_{min}(n,f,v)$ being the experimental function from (U.S. Department of Homeland Security 2010).