Approaches to Joint Base Station Selection and Adaptive Slicing in Virtualized Wireless Networks

Kory A. Teague

(ABSTRACT)

Wireless network virtualization is a promising avenue of research for next-generation 5G cellular networks. This work investigates the problem of selecting base stations to construct virtual networks for a set of service providers, and adaptive slicing of the resources between the service providers to satisfy service provider demands. A two-stage stochastic optimization framework is introduced to solve this problem, and two methods are presented for approximating the stochastic model. The first method uses a sampling approach applied to the deterministic equivalent program of the stochastic model. The second method uses a genetic algorithm for base station selection and adaptively slicing via a single-stage linear optimization problem. A number of scenarios are simulated using a log-normal model designed to emulate demand from real world cellular networks. Simulations indicate that the first approach can provide a reasonably tight solution, but is constrained as the time expense grows exponentially with the number of parameters. The second approach provides a significant improvement in run time with the introduction of marginal error.

This work received support in part from the National Science Foundation via work involved with the Wireless @ Virginia Tech research group.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP Third-Generation Partnership Project

5G-NR 5G New Radio

BS Base Station

CapEx Capital Expenditures

DEP Deterministic Equivalent Program

DSA Dynamic Spectrum Access

EB Exabytes

GA Genetic Algorithm

IoT Internet-of-Things

MIMO Multiple-Input Multiple-Output

mmWave Millimeter Wave

MNO Mobile Network Operator

MVNO Mobile Virtual Network Operator

NVS Network Virtualization Substrate

NwoB Networks without Borders

OpEx Operational Expenditures

PPP Poisson Point Process

QoS Quality of Service

RAN Radio Access Network

RP Resource Provider

SAA Sample Average Approximation

sDEP Sampled Deterministic Equivalent Program

SP Service Provider

SSLT Scalable, Spatially-correlated, Log-normal distributed Traffic

VNB Virtual Network Builder

VWN Virtualized Wireless Network

ZB Zetabytes

Chapter 1

Introduction

Mobile carriers have seen explosive growth in both the volume of users and the demands of those users within the networks they operate. New and evolving data-driven applications, such as audio/video streaming, social networking, and the Internet-of-Things (IoT) have also placed increasing demand upon the networks. In 2016, the amount of IP data handled by mobile networks exceeded 86 Exabytes (EB); it is projected to reach almost 200 EB in 2018, and 580 EB in 2021 [1]. Due to this exponential growth, incremental approaches to improve the network will fail to satisfy demand. As this growth continues in the near future, new architectures like 5G and its associated technologies will be needed to keep pace with the demand.

However, deployment of these technologies and networks can be a costly, prohibitive venture. To meet these demands requires a similar increase in capital (CapEx) and operational

expenditures (OpEx). As volumes and costs rise and margins shrink, approaches to reduce these expenditures become increasingly necessary. Resource infrastructure sharing has been a common practice for mobile network operators (MNOs) going back to 2G and 3G networks. First, MNOs needed to offer coverage for their users in regions where they had no infrastructure, leading to the creation of roaming agreements, eliminating the need for deploying new infrastructure in that region and reducing CapEx. Second, by sharing passive elements of the infrastructure, such as physical sites, tower masts, power, and air-conditioning, the CapEx of deploying new backhaul and radio access networks (RANs), such as cellular base stations (BSs), has decreased [2].

CapEx reductions from passive resource sharing drove an interest in resource sharing of the active elements of the network. For example, MNOs might share RANs, core networks, BSs, antenna systems, or backhaul, which leads to reductions in both CapEx and OpEx. The ability to share these active resources removed the necessity for network operators to own and maintain a physical network while providing actual MNO-like mobile services. These mobile virtual network operators (MVNOs) function similarly to MNOs, but operate a virtualized wireless network (VWN) comprised of virtual resources instead of physical resources without the associated CapEx. It has been shown that virtualization in this manner can increase overall demand satisfaction of a set of VWNs while decreasing overall cost (i.e., OpEx) by decreasing the idle capacity of the networks [3].

In order to take advantage of increasing virtualizable resources and competition for those resources, a specific problem must be solved: how to select the set of virtual resources

to form a VWN that meets its demands with necessary or maximum demand satisfaction at minimum cost. The solution to this problem is further complicated in the context of multiple MVNOs, each with one or more VWNs with unique demands, assessing a large pool of available virtual resources that can be adaptively allocated as demands shift.

This thesis addresses the topic of resource selection and adaptive slicing within cellular VWNs through the lens of stochastic optimization and investigates two approaches to efficiently reach a solution.

1.1 Trends in Wireless Networking

IP traffic is increasing across all types of networks, and is trending to become more mobile focused. According to the Cisco Visual Networking Index [1], global IP traffic will increase nearly threefold over the 2016-2021 time period, reaching 3.3 Zetabytes (ZB) annually in 2021 from 1.2 ZB annually in 2016. Traffic across the fixed internet backbone is projected to match this threefold pace, growing from 790 EB to 2.2 ZB. However, mobile data traffic is projected to have twice the growth of fixed internet over the same period, increasing almost sevenfold from 86 EB in 2016 to 580 EB in 2021. Traffic from wireless and mobile devices combined will reach 63% of total IP traffic by 2021, up from 49% in 2016. By 2021, smartphone IP traffic (33% of global IP traffic) will alone outnumber PC IP traffic (25%). In both the consumer and business markets, this demand includes enormous growth in video applications, specifically that of video streaming. By 2021, global IP video traffic

will reach 82% of all consumer internet traffic, up from 73% in 2016. By 2021, internet live video streaming will account for 13% of this video traffic, growing 15-fold over the period. Similarly, virtual and augmented reality uses will see the largest increase, growing at a 82% compound annual growth rate, and expected to reach a 20-fold increase between 2016-2021.

The technology underlying the mobile data network needs to continue to evolve with these changing trends and growth. The primary focus of the 5G cellular standard has been to meet these targets in an effective and robust manner. Of specific interest is that of aggregate data rate (e.g., area capacity, the available amount of data a network can facilitate over a unit area) and edge rate (e.g., 5% rate, the minimum data rate that can be reasonably provided to all but 5% of users) of the network. For 5G, the general consensus is that aggregate data rate and edge rate must be 1000x and 100x that of 4G, respectively [4]. To supply these rates, several strategies are being investigated, with three primary technologies being (1) the continuing of cellular densification and offloading, (2) increased bandwidth by expanding into new spectra like Wi-Fi and millimeter wave, and (3) increasing spectral efficiency through advances such as those in massive multiple-input multiple-output (MIMO).

The first strategy is extreme densification and offloading. By making network cells smaller, the number of active nodes increases for the same unit area. This is a common strategy across cellular generations, and a large impetus behind the use of smaller range RANs like microcells and femtocells [5]. Cell sizes have shrunk, dropping from the order of hundreds of square kilometers to now fractions of a square kilometer. The most important benefit of cell densification is that it increases spectral reuse, which reduces the amount of

users competing for the same resources. Theoretically, since signal-to-interference ratio is maintained as the cell shrinks, such densification can be repeated indefinitely as deployments allow [4, 6].

The second strategy is to increase bandwidth through the use of previously unused spectra such as millimeter wave (mmWave) and Wi-Fi. Cellular networks have utilized microwave frequencies ranging from a few centimeters to about a meter in wavelength; this range has become thoroughly occupied and to generate new bandwidth would require expanding to new frequencies [7, 8]. Up to now, mmWave has been unused and in some cases unlicensed due to very poor propagation properties [9, 10] and high equipment costs [10]. However, equipment costs are falling rapidly due to technological maturation. Further, the propagation qualities are increasingly surmountable as cell sizes shrink [11].

The third strategy involves the use of massive MIMO to increase spectral efficiency. MIMO uses multiple transmit and receive antennas to exploit multipath signal propagation, multiplying the capacity of a given radio link. The technology has been used for over a decade as a component of Wi-Fi before being introduced into the 3G and 4G standards [4]. A new approach to be used in 5G is that of "massive MIMO", where the number of transmit antennas at the BS greatly outnumber the number of active users [12]. For example, a given BS might have hundreds of antennas while maintaining data links for tens of users. This provides several benefits, most importantly vastly improving spectral efficiency.

5G must supply these rates at much higher energy and cost efficiencies, ideally matching or exceeding the capacity increases to avoid increasing overall network energy use and OpEx.

However, technologies that have been investigated to adequately increase the capacity of the network have several major hurdles to meet in order to be implemented at the desired energy and cost efficiencies. Massive MIMO requires the deployment of a vast number of antennas, which requires new BS architectures that have issues with scalability and cost. Millimeter-wave is more expensive than the more mature hardware of typical cellular bands. Decreasing cell size for cellular densification allows for smaller, cheaper BSs, but this decreased cost may not keep pace with the required number of increased deployments. [4]

3GPP (the Third-Generation Partnership Project) is currently working on finalizing the standard for 5G implementations. In December 2017, 3GPP froze the first half of release 15 of the 5G standard, covering 5G New Radio (5G-NR) which establishes specifications for new standalone 5G deployments. It is expected that 3GPP will freeze the second half of release 15, establishing the specifications of non-standalone 5G which utilizes existing LTE networks, in Summer of 2018. Further work on release 16 and beyond is still in progress.

1.2 Virtualization, Virtualized Wireless Networks, and the Networks without Borders Paradigm

One approach towards minimizing CapEx and OpEx of networks has been the utilization of resource sharing. Resource sharing encompases the sharing of resources between multiple networks and can take the form of *passive sharing*—referring to the sharing of physical sites,

tower masts, cabling, power supplies, and other components that are not actively on part of the network architecture—and *active sharing*—referring to the sharing of the active network architecture itself, such as backhaul and RAN. The practice has been utilized since 2G and 3G networks as a tool for reducing CapEx in expanding the network [2]. Since then, resource sharing has become more common; it is now available, standardized [13], and implemented in many major carrier networks. As reported by Costa-Perez et al. [14], a 2010 market survey [15] found that over 65% of European MNOs have deployed mobile infrastructure sharing in some form. It was further reported [14] that 20% of cells carry about 50% of total network traffic, with the remaining 80% of cells still causing OpEx with less utility. Through active resource sharing, networks can reduce or avoid redundant deployments and wasted capacity, reducing overall CapEx and OpEx.

The increasing prominence of active resource sharing challenges the traditional model of ownership of the various network layers and elements. Once it became feasible for network operators to utilize resources owned and maintained by other operators, it became possible for these MNOs to operate networks primarily or only using these shared resources. A given shared resource can be decoupled from a specific physical resource. This enables it to be adaptively associated with any of a given pool of qualifying physical resources as network conditions allow, establishing the shared resource as a virtual resource. Further, virtual networks can adapt to changing network conditions, adding and removing virtual resources as capacity requirements change. For example, an MVNO with a network of virtual resources can add additional virtual resources during peak hours when additional capacity for end

user satisfaction is needed, and removing unneeded resources during times of low demand to reduce OpEx of the network.

Other research has virtualization to improve performance in wireless networks. Panchal and Yates [16] have shown on an LTE testbed that active inter-operator resource sharing improves performance of overloaded networks in terms of decreased drop probability and overloaded sectors. Sharing methods that included virtualization provided further, albeit marginal, performance improvements at an increased complexity. In this capacity, improved performance allows for smaller networks reducing CapEx and OpEx. Costa-Perez et al. [14] found in LTE testbeds that network virtualization substrate (NVS), a suggested virtualization technique, provides improved overall throughput compared to networks without resource sharing.

1.2.1 Virtualization and the Network Value Chain

This concept of virtualization partitions the classical wireless networking value chain, allowing for specialization of segments of the value chain into new entities such as resource providers and service providers [2]. Traditional MNOs control every segment of the typical mobile network value chain (Figure 1.1 [17]), from spectrum to the end user. With the introduction of virtualization techniques, MVNOs can obtain access to bulk network services available from an MNO. This allows for MVNOs to specialize without the significant CapEx or responsibility to deploy and maintain the radio infrastructure [17]. For example,

an MVNO could focus on marketing, working solely within the distribution channel to the network's customers, or the MVNO could establish itself earlier in the value chain, focusing on operating the network from the core network.

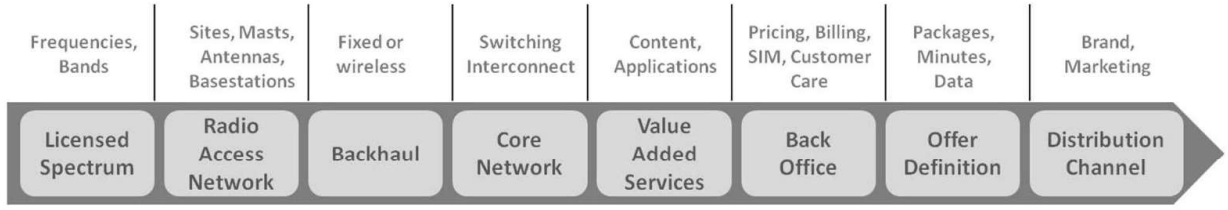


Figure 1.1: Typical MNO value chain [17]

Specialization of networks and the entities involved in the network can improve the cost efficiency of the network. According to Beckman and Smith [2]: "Extensive vertical integration is a characteristic of an immature product. As the product increases in complexity, it is no longer possible to [provide] an end-to-end solution." In both examples, the MVNO adds value to the traditional value chain by specializing in segments (e.g., marketing or service creation) that are different from the segments (e.g., network maintenance) still handled by the owner and operator of the network resources.

By focusing on the strengths provided by virtualization, more value can be generated through specialization. Doyle et al. [17] investigates the value chain with this segmentation in mind and introduces the Networks without Borders (NwoB) approach as a new service-oriented network with a proposed new value chain (Figure 1.2 [17]). The network under the NwoB approach is entirely service-oriented, where the network responds to services and connectivity is tailored for the service. Services have a wider meaning than the voice, text, and data of a typical MNO. Services also include that of Netflix-like or real-time video

streaming, IoT applications, or various types of over-the-top services. Each service would be provided by a service provider that compensates the virtual network operator operating a virtual network constructed specifically for the purpose of that service; the virtual network is the service. Unlike an MVNO which manages resources provided to it by agreement, the virtual network operator manages slices of virtual resources from a pool of all resources as provided through resource aggregating services.

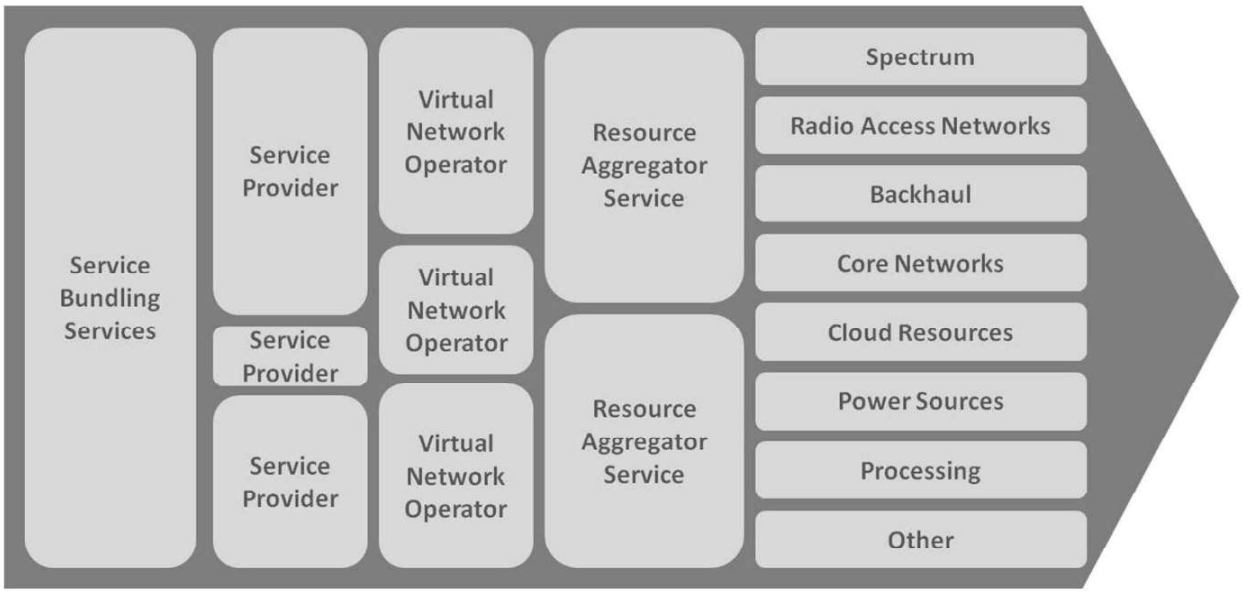


Figure 1.2: Proposed network value chain under the NwoB paradigm [17]

The benefits of this paradigm as proposed by Doyle et al. [17] are four-fold. First, it provides specialization and independence for each stage, allowing service providers to focus on generating value from services provided. Second, networks can be specialized for a service, reducing OpEx through extensive resource sharing. Third, as resources are virtualized and pooled together, any resource (e.g., typical RAN, Wi-Fi, mmWave, raw spectrum) could be added with the pool and utilized for a network as its properties fit the network's needs.

Fourth, it lowers the barrier for entry and establishes services for new entities to fulfill.

1.2.2 Virtualization Architecture in this Work

Recognizing the critical nature of virtualization and resource allocation, this thesis develops and analyzes two methods for constructing virtualized wireless networks built on a virtualization architecture [3, 18] inspired by the NwoB paradigm presented by Doyle et al. [17]. Figure 1.3 [3, 18] illustrates the three primary roles in this architecture: (1) the Resource Providers (RPs), (2) the Virtual Network Builders (VNBs), and (3) the Service Providers (SPs).

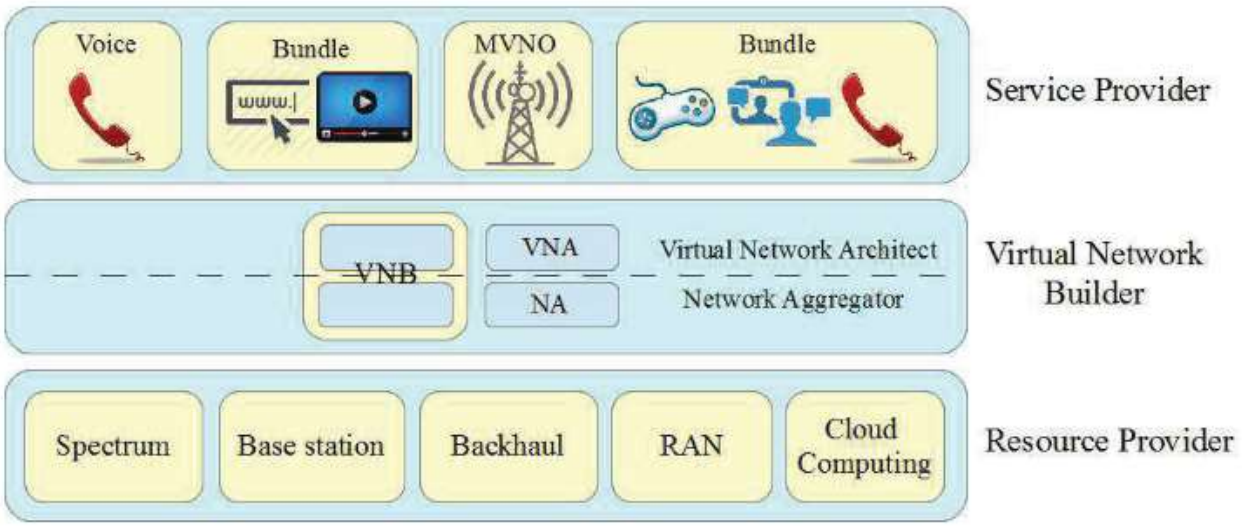


Figure 1.3: VWN architecture as used in this work [3, 18]

RPs deploy and maintain the physical resources that are to be virtualized and offered for use within the virtualization framework and are the various entities that occupy the rightmost column of segments (i.e., resources) in the NwoB value chain (Figure 1.2 [17]). These

resources can be in the form of any network-capable resource. For example, the resources could be BSs as provided by a traditional MNO, a company- or individual-owned WLAN, femtocell access points, available licensed or unlicensed spectrum, or cloud computing. An RP is then any entity that offers a virtualizable resource, such as a traditional MNO, company, or individual. RPs maintain the resources, but also determine how the resource would be sliced and shared.

The VNB acts as resource aggregator, VWN constructor, and as intermediary between SPs and RPs. Therefore, the VNB acts as a combination virtual network operator and resource aggregator in the NwoB value chain (Figure 1.2 [17]). The VNB aggregates the resources maintained by individual RPs to establish the pool of available virtual resources. The VNB also coordinates with SPs to understand the demands of their services and constructs VWNs tuned specifically to these demands. By understanding the needs of the services provided by the SPs, the VNB will evaluate which virtual and virtualizable resources available from the RPs are needed to construct the optimal[1] network for the SPs' needs, coordinate with the necessary RPs to obtain access to these resources for a given wholesale (OpEx) cost, and construct the network for the SPs to operate. Multiple VNBs can coexist, each with their potentially overlapping set of RPs from which to aggregate resources.

SPs operate similarly to the service providers in the NwoB approach. Primarily, an SP determines a service that they wish to provide, understands and enumerates the demands

[1]In this network context, "optimal" is loosely defined to mean a network that provides the maximum demand satisfaction for the SP at the minimum cost to be paid to the RP. These two requirements—maximum demand satisfaction and minimum cost—are frequently contradictory and need to be balanced by the VNB.

that are to be satisfied for that service, and provides the service over the VWN to their end users. SPs can provide a wide range of services over the network. The service could be a traditional MNO or be providing MNO-like services, such as voice calling and texting. Services could cover specific applications, such as IoT, teleconferencing, augmented or virtual reality, or emergency services. Other examples include traditional over-the-top services, such as Netflix-like or real-time (live) video streaming, social media (Facebook, Twitter, etc.), messaging (Skype, Groupme, etc.), or news/content feeds. Further, an SP could also bundle several services, either through a single VWN built for the bundle, or by bundling services provided by several SPs.

Between these three entity roles, various interactions become possible. The most common interactions are illustrated in Figure 1.4 [18]. The interactions between the various entity roles are: (A) among SPs; (B) between the SPs and the VNBs; (C) among VNBs; (D) between the VNBs and the RPs; and (E) among RPs. It should be apparent that across each of these interactions is the imposition of costs as exchange for the transfer of services, networks, and resources.

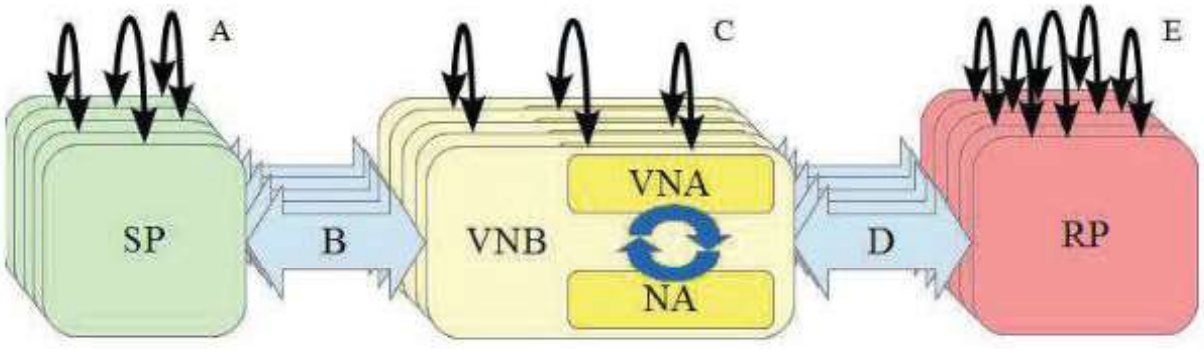


Figure 1.4: Interactions between roles in the VWN architecture [18]

Interaction (A) describes associations among various SPs. This would typically occur in situations where a SP desires to bundle the services of several SPs, or when a SP wishes to utilize a specialized network operation from another SP. Generally, this interaction would be performed manually over timescales of weeks or months.

Interaction (B) describes associations directly between SPs and VNBs. This would be one of the most common interactions within this framework. This interaction is bidirectional. In the first direction, SPs would provide the VNB they are coordinating with the specific demands and needs for the service they are providing. In the opposite direction, the VNB utilizes these conveyed needs and demands to construct a VWN and provide it for use to the service provider. Ideally, this interaction is highly or entirely automated, with the interactions varying from minutes or hours to weeks or months based on the level of automation and the specifics of the interaction. It will require optimization techniques and/or machine learning to achieve satisfactory results in this interaction.

Interaction (C) describes associations among various VNBs. Generally, such interactions may occur when a VNB does not have access to the appropriate virtual resources to satisfy interaction (B) interactions. Obvious examples include not having the necessary resources to provide adequate coverage over geographical areas or capacity in high-density environments. These interactions would generally be performed manually over timescales of weeks or months.

Interaction (D) describes associations between VNBs and RPs. Similar to interaction (B), this would be the other of the most common interactions within this framework. It is

also very important, as it establishes the mapping between the virtual and physical resources and builds the substrate that the framework is built upon. VNBs interact with the RPs by making requests for new resources and releasing unneeded resources. RPs interact with the VNBs by issuing updates, such as any changes to the resources in the VNBs' available pool of resources. Updates such as these are potentially highly disruptive to the VNBs as the updates can impact a large number of VWNs managed by the VNBs. With further similarity to interaction (B), this interaction is highly dependent on automation; based on the level of automation, this interaction may occur over timescales of minutes or hours to weeks or months.

Interaction (E) describes associations among various RPs. In this interaction, various RPs establish connections with each other to facilitate proper mapping of physical resources to virtual through the use of quality of service (QoS) parameters that define the abstracted resources. For example, a small-scale RP containing only an individual-owned femtocell could connect with a larger RP via this interaction so that the resource within the small-scale RP is visible for association with a VNB over interaction (D) as handled by the larger RP. These interactions could take seconds to weeks depending on the complexities of the RPs, their resources, and the amount of human involvement.

Other work has been completed using this architecture. Abdel-Rahman et al. [3] constructed several resource allocation models, including one-stage programs, two-stage programs, and a one-stage stochastic program, to investigate the efficacy of this virtualization architecture upon a preexisting set of resources. The implementation focused on interaction

(B) from the perspective of the VNB, and showed that virtualization decreased the cost and idle capacity of the networks and increased demand satisfaction of the networks.

Cardoso et al. [18] expanded on this work by introducing a two-stage stochastic program to optimize interaction (B). The two-stage stochastic resource allocation similarly reduces cost and idle capacity of the VWN compared to the network without sharing. However, no direct comparisons are made with the non-stochastic programs tested by Abdel-Rahman et al. [3].

Gomez et al. [19] utilized this architecture from an economics perspective. Using a matching markets framework, they investigated the interaction of association between SPs and VNBs, such as the methods for how SPs indicate their needs and how VNBs indicate their VWN capabilities, and the fees that SPs will pay to partner for a VNB. Gomez expanded on this work in her Ph.D. dissertation [20].

The focus of this thesis is on optimization approaches largely in the context of interaction (B). This problem involves establishing how SPs convey the demands needed by the VNB to construct an optimal VWN for the service provided by the SP. Further, the construction of the optimal VWN is sought within a short amount of time so that interaction (B) can be completed over shorter timescales (e.g., minutes or hours) instead of longer (e.g., days, weeks, months). With an optimal VWN in mind, construction of the VWN is inherently an optimization problem, and the search of expedient solutions lays within the study of optimization.

1.3 Review of Optimization Methods

In this thesis, I address the problem of the creation of optimal networks by a VNB that satisfy the specific demands of SPs using a pool of resources provided by a set of RPs. This is naturally an optimization problem, in which some objective function is either minimized or maximized. At it's most basic, optimization techniques (e.g., linear programming, integer programming) will find the set of input parameters that minimize or maximize a single decision variable—the value of the objective function—in context of a set of constraints.

1.3.1 Stochastic Programming

Standard linear and integer programming requires complete, certain knowledge of all parameters that affect the functions or model being optimized (i.e., the model's parameters and functions must be deterministic). Communications, especially wireless communications, can be highly non-deterministic as the communication channel introduces a large amount of uncertainty. Stochastic programming provides a powerful mathematical tool to handle optimization under such uncertainties.

Stochastic programming has been recently exploited to optimize resource allocation in various types of wireless communications operating under uncertainties. Abdel-Rahman et al. [3] exploit stochastic optimization within the framework of the virtualization architecture presented in Section 1.2.2 to minimize the cost of resource allocation by introducing probabalistic QoS guarantees. Cardoso et al. [18] expand on that work by introducing a second stage

to balance maximizing demand satisfaction while minimizing cost. Further examples include resource allocation in dynamic spectrum access (DSA) networks [21], optimal orchestration of LTE-U networks utilizing Wi-Fi access points [22], resource allocation in opportunistic LTE-A networks considering end user rate demand satisfaction [23], resource allocation for OFDMA-based cognitive radios considering primary user system interference [24], and predictive resource allocation for energy-efficient video streaming to mobile end users [25].

Introducing stochastic parameters and constraints allows the optimization model to consider probabilities within the optimization. In the case of resource allocation in networks, it may be possible to allocate enough resources to satisfy all end-user demand. Such an optimization may require too many resources to be economical considering the law of diminishing returns, with the solution being cost prohibitive. It is much cheaper to solve such that 95% or 99% of demand is satisfied, leaving some demand unsatisfied.

However, standard linear programming techniques cannot solve models with stochastic parameters. Stochastic programming therefore requires converting the stochastic program into its deterministic equivalent program (DEP) which replaces all stochastic variables with deterministic variables [26]. The process of forming a DEP from a stochastic program involves converting each stochastic variable into a set of all possible scenarios and scenario probabilities. These scenarios and scenario probabilities are present within the model as a new dimension and weight for the now-deterministic variable. To fully encapsulate the stochastic variable, the deterministic equivalent variable may be composed of an infinite set.

Resource allocation problems are typically some form of integer programming—in which

all decision variables (unknowns) are integers—or mixed integer programming—in which some decision variables (unknowns) are integers. Both integer and mixed integer programs are generally[2] considered NP-hard[3]. As the programs increase in scope, they become more computationally complex to solve; accounting for the scenarios of the previously stochastic variables further increases this complexity. Finding the optimal solution may require more time than is feasible; in the worst case, these problems run in exponential time complexity.

1.3.2 Metaheuristic Approaches

The use of heuristic or metaheuristic algorithms can provide close-to-optimal solutions in much better time. Examples include hill climbing, simulated annealing, ant colony optimization, and particle swarm optimization. Each of these approaches are iterative techniques.

Hill climbing starts with an arbitrary solution and makes incremental changes to variables, finding a new solution. If the new solution is better than the previous, the new solution is iterated upon. This continues until no further improvements can be made. Hill climbing will only find the local maximum close to the initial arbitrary solution, and is best in convex problems where the only local maximum is guaranteed to be the global maximum.

[2]Some subclasses of integer and mixed integer programs are efficiently solvable, but these are the exception. Several classic NP-Complete (a subset of NP-Hard) problems [27] are integer programs and mixed integer programs.

[3]Finding the minimum resource allocation that provides coverage over a geographic area falls within a category of problems referred to as *minimum set cover problems*. It is apparent that the problem considered in this thesis—specifically the stochastic program proposed in Section 2.2—is some form of minimum set cover problem; specifically, it might be referred to as a capacitated set cover problem. Minimum set cover problems are provably NP-hard and typically rely on approximation solutions to solve in a feasible amount of time [28].

Simulated annealing is inspired by the process of annealing found in metallurgy, where metal is heated to the point where atoms can migrate, reducing defects in the crystalline structure. In simulated annealing, the model has some notion of temperature, which represents the internal energy of the system, and states, which represent possible solutions to the system being optimized. The system has an initial state, each state has an associated energy, and the system is attempting to reach the state of lowest energy. On each iteration, the heuristic considers a neighboring state, and chooses to transition to the new state with a probability dependent on the energy of the current state, the energy of the neighboring state, and the temperature. This transition can lead from a lower energy state (better) to a higher energy state (worse), and will do so more often while it has a higher temperature. Gradually, the system will cool and decrease the temperature, which causes the system to tend to select states with lower energy; as the temperature drops, the systems overall energy drops. When the temperature reaches zero, the system will only transition to states of lower energy (i.e., that are more optimal), reducing to the hill climbing algorithm.

Ant colony optimization is inspired by the behavior of ants. A colony of ants move around independently trying to find food, laying pheromones on the taken path. Upon crossing paths, ants have a probabilistic chance to follow the new path based on the strength of the pheromones of the new and old paths. Over time, pheromones evaporate, and paths less taken will weaken. Longer paths, since they take longer to traverse and will be reinforced less often, will also weaken. This has benefits over approaches like simulated annealing because it adapts in real time.

In particle swarm optimization, a number of candidate solutions, called particles, are created that move semi-chaotically. In each iteration, every particle will move according to its velocity. Each particle has it's best known position, and it's velocity updates in a way that is guided by their own best known position and the swarm's best known position. This allows a large portion of the search space to be investigated, with candidate solutions exploring regions containing local maxima until it settles to exploit and find the best found local maxima.

In this thesis, I utilize a genetic algorithm as an approach for optimization. A genetic algorithm is a form of evolutionary algorithm, a set of algorithms which are inspired by biological evolution and natural selection. Each iteration is called a *generation* and is composed of a number of candidate solutions called *individuals*. Each individual is defined by a *chromosome* which details the specific candidate solution. During each generation, every individual is evaluated on its *fitness*, a function dependent on the individual's chromosome; the higher the individual's fitness, the more optimal the individual. Individuals called *parents* are then randomly selected to pass their chromosome onto the next generation in a process called *selection*; in selection, more fit individuals are more likely to be selected. With a certain probability, groups of parents will undergo *crossover* and exchange the data contained within their chromosomes to form new *children* that are a mixture of the parents; if mixing does not occur, the parents are cloned into the next generation as children. Then, individual bits within the children's chromosomes have a chance to flip, or *mutate*. The resulting children from crossover and mutation form the entire next generation.

Since chromosomes from fitter individuals are more likely to pass on to subsequent generations, generations gradually become fitter. Through crossover, fit chromosomes may combine to form fitter children that proliferate; less fit children are often also formed, but are generally not selected for later generations. Mutation introduces diversity into the generations, which expand the exploration of the search space. More details, including that of implementation and variants, will be expanded upon in Section 3.2.

Genetic algorithms have been used previously as approaches for simplifying the search spaces of large, complex stochastic optimization problems. For example, Cui et al. [29] used a genetic algorithm where each chromosome defined a subproblem of larger resource allocation optimization problem, and the fitness was evaluated by solving the subproblems with linear programming optimization methods. One approach investigated in this thesis coordinates a genetic algorithm with an optimization program wherein the genetic algorithm solves for and fixes a decision variable to simplify the larger optimization program. Hybrid approaches (e.g., Cui et al.) and other effective metaheuristic algorithms (e.g., ant colony optimization, particle swarm optimization, neural networks, and machine learning) are worth investigating in the context of the posed VWN architecture, but beyond the scope of this thesis.

1.4 Thesis Objective

The objective of this thesis is to develop two approaches to joint resource allocation to construct a set of a VWNs and adaptively slice the selected resources to allocate to the

individual VWNs. A model will be presented as the context for these approaches, expanding upon the VWN architecture proposed in Section 1.2.2. The validity of this model will be restricted to the scope of cellular networks using generic base stations as its resources. The two proposed approaches will be performed within the VNB, and evaluated in four cases that differ in the resources provided by the RPs and service demands to be satisfied by the SPs. Accordingly, the efficacy of these approaches will be measured primarily by the optimality of the solutions, such as cost and network service demand satisfaction, and the run time, providing the VNB with a sufficient solution in a reasonable amount of time.

1.5 Thesis Outline

This thesis is organized as follows. Chapter 2 defines the model used for the resource allocation methods explored in this thesis. Further, Chapter 2 also details the two-stage stochastic optimization problem which optimally performs resource selection and slicing as a basis of approaches presented within this work. Chapter 3 establishes the two approaches investigated to provide solutions to the optimization problem posed in Chapter 2: a sampled deterministic equivalent program which solves the problem as a whole and a genetic algorithm that simplifies the problem by providing an estimated optimal resource selection. Chapter 4 tests these two approaches by presenting four data sets that mimic real world cellular networks and evaluates the results. Chapter 5 contains the conclusions and proposed future work in this area.

Chapter 2

Virtual Network Builder Model

This chapter establishes the mathematical foundation for this thesis. First, a geographic model is presented defining an area of interest, the pool of resources maintained by the RPs for use by the VNB, and a characterization of the service demand of each SP. Second, a specific demand model, based on the empirical analysis of collected cellular network traces, is presented. This is the demand model that will be used throughout this thesis. Third, a two-stage stochastic program is developed to model the problem of resource selection and adaptive slicing within the VNB.

Throughout this thesis, stochastic variables will be differentiated from deterministic variables with a tilde ($\sim$) placed above the symbol.

2.1 Network Area Definitions

Consider a geographic region, $\mathcal{R}$, of width X meters and length Y meters that contains a VNB and a set $\mathcal{S} \stackrel{\text{def}}{=} \{1, 2, \ldots, S\}$ of virtualized resources (i.e., BSs) the VNB has aggregated for use in the construction of VWNs. The pool of BSs, $\mathcal{S}$, is mapped to physical BSs owned and maintained by RPs and made available for use through contracts between the RPs and the VNB. The contract-negotiated cost for the VNB to lease BS $s \in \mathcal{S}$ is denoted by c_s. The costs for the BSs used within a constructed VWN are passed to the SPs as part of the overall cost of the network. The rate capacity of BS $s \in \mathcal{S}$ is denoted by r_s and its coverage radius is denoted by b_s.

Let $\mathcal{N} \stackrel{\text{def}}{=} \{1, 2, \ldots, N\}$ be the set of SPs seeking a VWN with coverage within the region $\mathcal{R}$. An SP $n \in \mathcal{N}$ associates with the VNB to create the desired VWN. Through this association, SP n must coordinate with the VNB to indicate the demands of the intended service the VWN would need to satisfy. SP n must know and communicate to the VNB the estimated geographic distribution of the service's traffic demand density (or demand *intensity*) as a function, $\lambda_n(x, y)$, $n \in \mathcal{N}$, $x \in [0, X]$, $y \in [0, Y]$, in terms of $^{\text{bits}}/_{\text{km}^2}$. The demand intensity function could be in the form of a continuous function or as discrete pixels (e.g., a bitmap), and indicates the locations of necessary coverage and the desired capacity within specific regions of the area. Examples of possible maps could be for services such as localized video streaming (specific, localized coverage with high regional capacities) or MNO-like voice lines (broad coverage with comparatively low capacity). For an example of

λ_n, see Section 2.1.1.

Further, the SP would also provide the desired or needed percent demand satisfaction rate[1] for the service. Some services, such as those related to emergency services, must have nearly 100% demand satisfaction. Others, such as generic voice lines or video streaming, may be able to tolerate some unsatisfied demand as a tradeoff for decreased network leasing or operational costs.

λ_n describes the geographic distribution of SP $n \in \mathcal{N}$ within $\mathcal{R}$. At any instantaneous point in time, λ_n is realized as a set of discrete demand points (each representing e.g., user equipment). Let $\mathcal{M}_n \stackrel{\text{def}}{=} \{1, 2, \ldots, M_n\}$ be this set of demand points SP n is attempting to satisfy with its service at a given point in time. Each demand point $m \in \mathcal{M}_n$ is seeking to connect to the VWN operated by SP $n \in \mathcal{N}$ and is located within the domain of $\mathcal{R}$ (i.e., $(\tilde{x}_{m_n}, \tilde{y}_{m_n})$, $\tilde{x}_{m_n} \in [0, X]$, $\tilde{y}_{m_n} \in [0, Y]$). Demand point locations are stochastically determined by the distribution of traffic as described by the demand intensity function λ_n. A realization of these demand points can be found as a two-dimensional non-stationary (or inhomogeneous) Poisson point process (PPP) using λ_n as it's spatial intensity function.

For purposes of visualization or compuation, this non-stationary PPP is generatable using an accept-reject method [30]. A stationary PPP is generated relative to the maximum value of the traffic demand density function within $\mathcal{R}$. That is, a number of points generated within the region is selected from a Poisson random variable with parameter (or mean) $\lambda_{n,\max} * X * Y$,

[1]Percent demand satisfaction rate refers to the percent of the demand described by λ_n that is satisfied by VWN constructed by the VNB. In this context, it is the percent of total requested demand that is sliced resources by the VNB. This will be more formally described later in Section 2.2 (eq. (2.21)) and evaluated during simulations in Chapter 4.

where $\lambda_{n,\max}$ is the maximum value of λ_n within $\mathcal{R}$. Each point is then independently and uniformly distributed (i.e., each point has a location $(x,\, y)$ with $x \sim \mathcal{U}\,(0,\, X)^2$ and $y \sim \mathcal{U}\,(0,\, Y)$) over $\mathcal{R}$. Then, each point undergoes the accept-reject procedure to inhomogenize the stationary PPP. Each point is kept with a probability of the ratio of the value of the intensity function at that point's location to the maximum value of the intensity function. That is, for each point in the PPP a uniform random variable, $P \sim \mathcal{U}\,(0,\, 1)$, is generated and the point is either *accepted* and kept or *rejected* and discarded according to

$$\begin{cases} \text{the } i^{\text{th}} \text{ point is kept}, & \text{if } P \leq \frac{\lambda(x_i, y_i)}{\lambda_{n,\max}}; \\ \text{the } i^{\text{th}} \text{ point is discarded}, & \text{otherwise}, \end{cases} \tag{2.1}$$

where x_i and y_i are the x- and y-coordinates of the i^{th} point of the stationary PPP.

Each demand point $m \in \mathcal{M}_n$ loads the VWN of SP $n \in \mathcal{N}$ with point traffic demand denoted by d_n. So that the total demand described by λ_n is allocated by the points in $\mathcal{M}_n$, the overall demand

$$D_n = \int_0^X \int_0^Y \lambda_n\,(x,\, y)\, dy\, dx, n \in \mathcal{N}, \tag{2.2}$$

of the demand density distribution is evenly distributed such that

[2] $\mathcal{U}\,(a,\, b)$ refers to a random variable uniformly distributed over the domain $[a,\, b]$, where $a < b$.

$$d_n = \frac{D_n}{M_n},\ n \in \mathcal{N}. \tag{2.3}$$

Let $\tilde{u}_{mns} \in [0,\ 1]$ represent the normalized capacity (with respect to r_s) of BS $s \in \mathcal{S}$ at point $m \in \mathcal{M}_n, n \in \mathcal{N}$, associated with SP $n \in \mathcal{N}$ (i.e., the normalized maximum rate that a user can receive at point m from BS s). Specifically,

$$\tilde{u}_{mns} \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if demand point } m \text{ of SP } n \text{ is located more than } b_s \text{ meters from} \\ & \text{from BS } s; \\ 1, & \text{if demand point } m \text{ of SP } n \text{ is located within a small distance of} \\ & \text{BS } s; \\ (0,\ 1), & \text{otherwise.} \end{cases} \tag{2.4}$$

It is apparent that $\tilde{u}_{mns}$ will vary according to the path-loss characteristics of the environment, the locations of the demand points, and other various factors. In some instances, it can be beneficial to simplify this definition such that

$$\tilde{u}_{mns} \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if demand point } m \text{ of SP } n \text{ is located more than } b_s \text{ meters from} \\ & \text{BS } s; \\ 1, & \text{otherwise (i.e., if demand point } m \text{ of SP } n \text{ is located less than} \\ & \text{or equal to } b_s \text{ meters from BS } s). \end{cases} \tag{2.5}$$

Under this definition, $\tilde{u}_{mns}$ is binary and wholly determined by the stochastic locations of

the demand points and the fixed locations and ranges of the BSs.

From the perspective of the VNB, each SP is only distinguishable by its overall demand characteristics. These demand characteristics are defined by the SPs' demand density distributions. The VNB must construct a VWN for each SP, but for optimal VWNs to be created, the VNB must consider the demands of all SPs simultaneously and in context of each other. For the VNB, all SP demand points are indistinguishable and independent. Therefore, the VNB considers a single set of demand points

$$\mathcal{M} \stackrel{\text{def}}{=} \bigcup_{n \in \mathcal{N}} \mathcal{M}_n = \{1, 2, \ldots, M\}, \tag{2.6}$$

where $M = \sum_{n \in \mathcal{N}} M_n$, with demands d and stochastic normalized capacities $\tilde{u}_{ms}$, $m \in \mathcal{M}$, $s \in \mathcal{S}$. $\mathcal{M}$ can be considered a set of demand points associated with a superposition of PPPs with intensity $\lambda = \sum_{n \in \mathcal{N}} \lambda_n$.

2.1.1 Example Demand Distribution; The SSLT Model

One major assumption made in Section 2.1 is that SPs must communicate the demand characteristics (i.e., λ_n) of their service to the VNB to properly facilitate VWN construction. In this subsection, I establish an example model demonstrating the demand characteristics to be communicated by generating an example hypothetical demand intensity function. For testing the approaches for VWN construction presented in Chapter 3, this example is the fundamental model used for simulating SP demand in cellular network-based services.

Gotzner et al. [31] have shown that a log-normal distribution[3] can approximate traffic demand in real-world cellular networks. It has also been shown that traffic distributions are spatially correlated [33,35]. Lee et al. [32,36] presented the Scalable, Spatially-correlated, and Log-normally distributed Traffic (SSLT) model to emulate the characteristics of real world cellular data networks. This model is flexible and can be adjusted to simulate numerous cellular networks, and can characterize the demand of a supposed SP in a way that mimics real-world data. I use a variant of the SSLT demand model presented by Lee et al., which I altered to be a continuous function serving as a continuous or pixelated demand density map.

To generate this spatial SSLT model distribution over the area of consideration, an initial Gaussian field, $\lambda^G = \lambda^G(x,\, y)\,,\, x \in [0,\, X]\,,\, y \in [0,\, Y]$, is generated by

$$\lambda^G(x,\, y) = \frac{1}{L} \sum_{l=1}^{L} \cos\left(i_l x + \phi_l\right)\, \cos\left(j_l y + \psi_l\right) \tag{2.7}$$

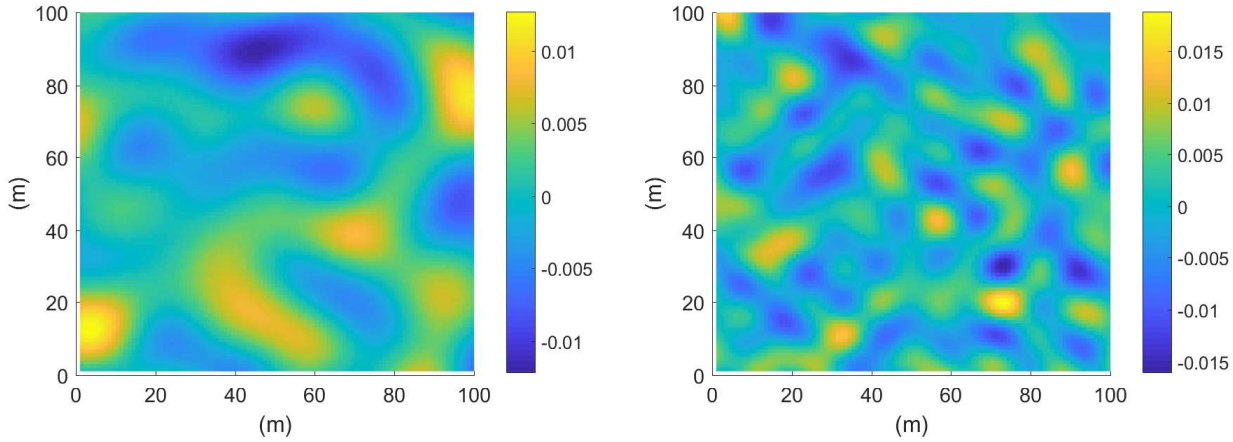
where $\mathcal{L} \stackrel{\text{def}}{=} \{1,\, 2,\, \ldots,\, L\}$ is a set of the products of two cosines with stochastic angular frequencies $i_l,\, j_l \sim \mathcal{U}(0,\, \omega_{\max})\,,\, l \in \mathcal{L}$ and phases $\phi_l,\, \psi_l \sim \mathcal{U}(0,\, 2\pi)\,,\, l \in \mathcal{L}$. As L increases, it is expected that λ^G approaches a Gaussian random field according to the central limit theorem.

[3]It has also been shown that traffic distributions in cellular networks can be approximated by a Weibull distribution [32], by mixtures of log-normal distributions [32, 33], or by an α-stable distribution [34].

According to Lee et al. [36], λ^G is spatially correlated with autocorrelation function

$$R\left(dx,\, dy\right) = \mathbb{E}\left[\lambda^G\left(x,\, y\right)\ \lambda^G\left(x+dx,\, y+dy\right)\right] = \frac{1}{4L}\ \mathrm{sinc}\left(\omega_{\max} dx\right)\ \mathrm{sinc}\left(\omega_{\max} dy\right). \quad (2.8)$$

The autocorrelation function is notably dependent on the maximum angular frequency defining ρ^G, $\omega_{\max}$. As $\omega_{\max}$ increases, the demand of adjacent regions become less correlated. $\omega_{\max}$ is effectively a measure of the inhomogeneity of ρ^G. This effect of $\omega_{\max}$ on the inhomogeneity of λ^G is shown in Figure 2.1; Figure 2.1b fluctuates more rapidly than Figure 2.1a, which is generated from a smaller $\omega_{\max}$.



(a) $\omega_{\max} = {}^{2\pi}/_{30}$ (b) $\omega_{\max} = {}^{4\pi}/_{30}$

Figure 2.1: Two example Gaussian random fields (i.e., λ^G) for generating the SSLT demand model, varying by $\omega_{\max}$. $L = 100000$, $X = 100$, $Y = 100$

The approximate Gaussian distribution λ^G is then normalized to a standard normal distribution[4], $\lambda^S = \lambda^S\left(x,\, y\right),\ x \in [0,\, X],\ y \in [0,\, Y]$,

[4]A standard normal distribution is a Gaussian distribution with mean $\mu = 0$ and variance $\sigma = 1$.

$$\lambda^S(x,\, y) = \frac{\lambda^G(x,\, y) - \overline{\lambda^G}}{\sqrt{\mathrm{Var}\left(\lambda^G\right)}}, \tag{2.9}$$

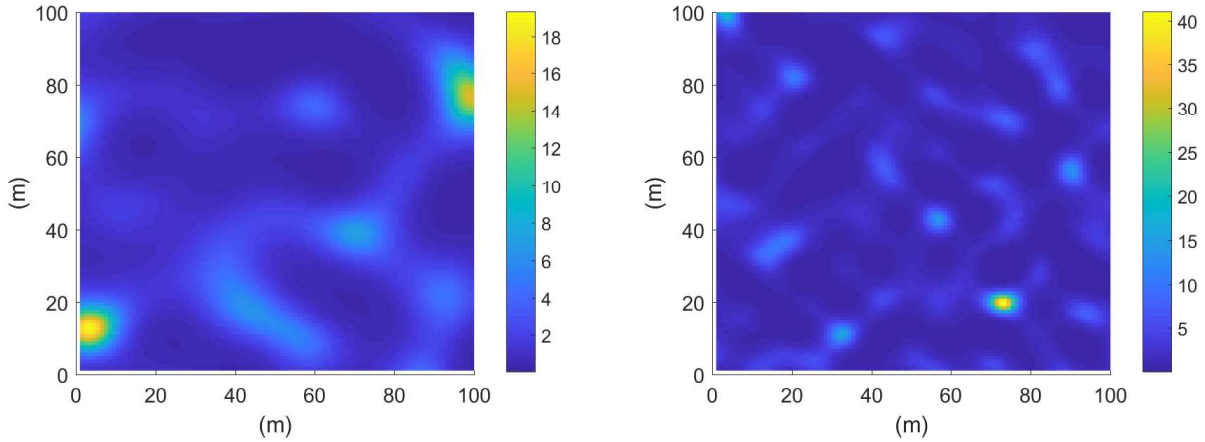
where $\mathrm{Var}\left(\lambda^G\right) = \mathbb{E}\left[\left(\lambda^G\right)^2\right] - \mathbb{E}\left[\lambda^G\right]^2$ is the variance of λ^G and $\overline{\lambda^G} = \mathbb{E}\left[\lambda^G\right]$ is the mean of λ^G. While this could be mathematically derived, in practice $\mathrm{Var}\left(\lambda^G\right)$ and $\overline{\lambda^G}$ are the sample variance and mean which are empirically found from a set of uniformly distributed sampled points (i.e., a simple random sample of λ^G).

The final log-normal distribution, $\lambda = \lambda(x,\, y), x \in [0,\, X], y \in [0,\, Y]$, is determined by assigning location (μ) and scale (σ) parameters to λ^S according to

$$\lambda(x,\, y) = \exp\left(\sigma\, \lambda^S(x,\, y) + \mu\right). \tag{2.10}$$

Figure 2.2 shows the resulting SSLT demand density distribution, λ, generated from the λ^G fields displayed in Figure 2.1 with default location and scale parameters (i.e., $\mu = 0$, $\sigma = 1$). By controlling the maximum angular frequency of the originating Gaussian random field, $\omega_{\max}$, and the log-normal location and scale parameters, the SSLT model can be used to simulate the demand characterization of a hypothetical SP service.

Lee et al. [32, 36] implement their proposed SSLT demand model as a discrete pixelated set of rectangular cells, the value of which indicates the overall demand located within that cell's region. Each demand point located within the SSLT area has an identical amount of demand associated with it. The value of each cell represents the number of homogeneous



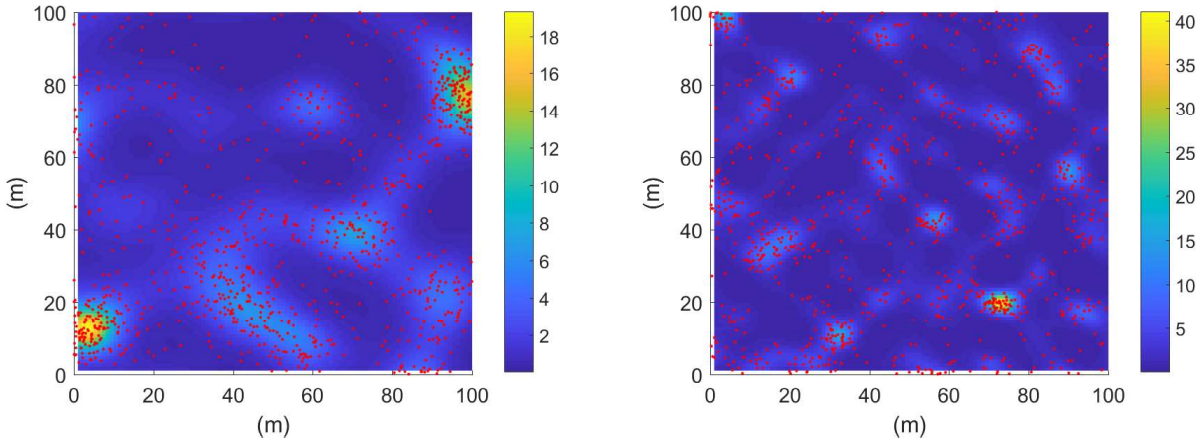
(a) $\omega_{\max} = 2\pi/30$ (b) $\omega_{\max} = 4\pi/30$

Figure 2.2: Two SSLT demand fields generated from the λ^G demand fields displayed in Figure 2.1. $\mu = 0$, $\sigma = 1$

demand points located within that cell, and these demand points are uniformly distributed within that cell.

I deviate from their implementation by leaving the SSLT distribution as a continuous function representing the overall demand of the region, and only pixelate it for visualization purposes. To generate discrete demand points, I generate a non-stationary PPP using λ as the spatial intensity function as described by the accept-reject method (eq. (2.1)) described previously. This allows the specific number of demand points to be controlled and accommodates for the assumption that the SSLT demand model is an overall distribution of demand points rather than each cell operating as independent PPP. Figure 2.3 shows a realization of demand points distributed as a non-stationary PPP.

This SSLT model is used in Chapter 4 to generate SP service demand and end-user demand points while testing the approaches described in Chapter 3.



(a) $\omega_{\max} = 2\pi/30$ (b) $\omega_{\max} = 4\pi/30$

Figure 2.3: Example demand point realizations distributed according to the described non-stationary PPP with 400 demand points.

2.2 Stochastic Optimization

As presented in Section 2.1, the VNB present within $\mathcal{R}$ must construct a set of VWNs to satisfy the needs of the services the various SPs seek to provide. Each service provider provides a characterization of its demand, such as the SSLT model presented in Section 2.1.1. With this information, the VNB must select the subset of virtual resources available in $\mathcal{S}$ that minimizes the cost to the VNB, and associate slices of these resources to VWNs that optimally satisfy the SPs' demand.

I formulate the presented problem of resource selection as a two-stage stochastic optimization program (eqs. (2.12)–(2.18)). Let z_s, $s \in \mathcal{S}$, be a binary decision variable defined as

$$z_s \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if resource } s \text{ is selected to be sliced into a VWN;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

which establishes VNB resource selection. Further let $\delta_{ms} \in [0, r_s]$, $m \in \mathcal{M}$, $s \in \mathcal{S}$, be a decision variable defined as the rate of resource s that is allocated to demand point m. δ_{ms} is said to be the *slice* of BS s allocated to the VWN service that is associated with demand point m.

To balance the interest of maximizing demand satisfaction against minimizing cost, I introduce the positive real number α as a weighting coefficient between the two stages (eq. (2.14)). The SP indicates the desired amount of demand satisfaction necessary for the service the SP provides; α, as set by the VNB, realizes this degree of demand satisfaction of the constructed VWNs relative to their cost.

The first stage objective function (eq. (2.12)) minimizes the total cost of the selected network with respect to that formed network's expected ability to satisfy the demand contained within $\mathcal{R}$. It characterizes this demand satisfaction as the expectation of $h(z, u)$, which is the optimal value of the second stage (eq. (2.14)) given a fixed z_s from the first stage. The second stage is normalized relative to D—the maximum value of the second stage if all demand is allocated slices—and weighted according to α. The optimal value of the second stage maximizes demand satisfaction by maximally slicing the BSs comprising the network selected by the first stage to the SPs' demands. The first stage handles the

Two-Stage Stochastic Optimization Program for BS Selection and Adaptive Slicing

$$\underset{\{z_s\}}{\text{minimize}} \left\{ \sum_{s \in \mathcal{S}} c_s \, z_s - \frac{\alpha}{D} \, \mathbb{E}\left[h\left(\boldsymbol{z}, \, \tilde{\boldsymbol{u}} \right) \right] \right\} \tag{2.12}$$

subject to:

$$z_s \in \{0, \, 1\}, \, \forall s \in \mathcal{S} \tag{2.13}$$

where $h(\boldsymbol{z}, \tilde{\boldsymbol{u}})$ is the optimal value of the second-stage problem, which is given by:

$$\underset{\{\delta_{ms}\}}{\text{maximize}} \left\{ \sum_{m \in \mathcal{M}} \sum_{s \in \mathcal{S}} \delta_{ms} \, \tilde{u}_{ms} \right\} \tag{2.14}$$

subject to:

$$z_s = \mathbb{1}_{\left\{ \sum_{m \in \mathcal{M}} \delta_{ms} > 0 \right\}}, \, \forall s \in \mathcal{S} \tag{2.15}$$

$$\sum_{s \in \mathcal{S}} \delta_{ms} \, \tilde{u}_{ms} \leq d_m, \, \forall m \in \mathcal{M} \tag{2.16}$$

$$\sum_{m \in \mathcal{M}} \delta_{ms} \leq r_s, \, \forall s \in \mathcal{S} \tag{2.17}$$

$$\delta_{ms} \in [0, \, d_m], \, \forall m \in \mathcal{M}, \, \forall s \in \mathcal{S} \tag{2.18}$$

interaction between the RPs and the VNB where the VNB selects resources to use, and the second stage handles the interaction between the VNB and the SPs where the VNB slices the selected resources to the SPs' VWNs (i.e., interactions D and B as described in Section 1.2.2, respectively).

Equations (2.13) and (2.18) are constraints that implement the defined ranges of the decision variables z_s and δ_{ms}. Equation (2.16) is a constraint that reinforces eq. (2.18) and asserts that a demand point is not overallocated by slicing it more resources than it demands. Similarly, eq. (2.17) ensures that a given BS is not over allocated to demand points.

Equation (2.15) ensures that demand is only allocated from available, selected resources. For this constraint, $\mathbb{1}_{\{*\}}$ is the indicator function, which is defined by

$$\mathbb{1}_{\{*\}} \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if condition } * \text{ is true;} \\ \\ 0, & \text{otherwise.} \end{cases} \tag{2.19}$$

δ_{ms} is the slice of BS $s \in \mathcal{S}$ that is allocated to demand point $m \in \mathcal{M}$. The VWN constructed for SP $n \in \mathcal{N}$ is comprised of all of the slices allocated to the SP's demand points. That is, the VWN constructed for SP $n \in \mathcal{N}$ is

$$\bigcup_{m \in \mathcal{M}_n} \delta_{ms},\ s \in \mathcal{S}, \tag{2.20}$$

where $\mathcal{M}_n \subseteq \mathcal{M}$ is the set of demand points associated with SP n. It should be noted that

$\delta_{ms} = 0, \forall m \notin \mathcal{M}_n \subseteq \mathcal{M}$ and $\delta_{ms} > 0, \forall m \in \mathcal{M}_n \subseteq \mathcal{M}$. From this, the percent demand satisfaction rate can be determined for each SP $n \in \mathcal{N}$ as

$$\text{VWN Demand Satisfaction Rate for SP } n = \sum_{m \in \mathcal{M}_n} \sum_{s \in \mathcal{S}} \frac{\delta_{ms}}{d_n \ M_n} \tag{2.21}$$

The stochastic optimization program (eqs. (2.12)–(2.18)) models the problem faced by a VNB that wishes to construct VWNs for and balance the needs of the various SPs. However, this program is not directly solvable, as mixed integer programming optimization tools cannot be used to solve stochastic optimization programs. Chapter 3 poses two approaches for solving this program.

Chapter 3

Approximation Approaches

Chapter 2 established a mathematical model for analyzing the relationship between SPs and a VNB and proposed a two-stage stochastic optimization program (eqs. (2.12)–(2.18)) for constructing VWNs in the context of that relationship. In this chapter, I present two approaches that arrive at a solution for this program. First, I modify the stochastic program to convert it into a deterministic form, the deterministic equivalent program (DEP), and sample it such that it is solvable using typical mixed integer programming optimization tools. From this, I further derive a model to adaptively slice BSs into new VWNs when resources are already selected. I then present a heuristic approach via genetic algorithm (GA) to handle resource selection with lower computational complexity.

3.1 Approach I: Deterministic Equivalent Program

In order to directly solve the two-stage stochastic optimization program (eqs. (2.12)–(2.18)), it must be converted into a deterministic equivalent program (DEP). The DEP is equivalent to the original stochastic optimization program, but does not contain any stochastic variables (only deterministic variables) [26]. This is accomplished by converting stochastic variables into sets which contain every scenario or realization in the scope of the stochastic variables.

Let Ω be defined as the sample space (i.e., the set of all scenarios) of $\tilde{u}_{ms}$. Since $\tilde{u}_{ms}$ is binary for all values of $m \in \mathcal{M}$ and $s \in \mathcal{S}$, Ω is finite and with a cardinality of $2^{M \cdot S}$ (i.e., $\Omega = \left\{1, 2, \ldots, 2^{M \cdot S}\right\}$). Each scenario $\omega \in \Omega$ is distinct and defined by a realization of $\tilde{u}_{ms}$; variables that are dependent on the scenario are shown with a superscript (ω) with the specific scenario it is dependent on indicated by ω. The probability of scenario ω (i.e., the probability of a specific realization of $\tilde{u}_{ms}$ occurring), is $p^{(\omega)}$, where $\sum_{\omega \in \Omega} p^{(\omega)} = 1$. By including Ω and $p^{(\omega)}$ and substituting $u_{ms}^{(\omega)}$ and $\delta_{ms}^{(\omega)}$ for $\tilde{u}_{ms}$ and δ_{ms}, the DEP (eqs. (3.1)–(3.5)) of the two-stage stochastic optimization program modeled in Section 2.2 is established.

The objective function (eq. (3.1)) combines both objective functions of the initial stochastic optimization program (eqs. (2.12) and (2.14)) into a single deterministic objective. The goal modeled by eq. (3.1) is unchanged from that modeled by eqs. (2.12) and (2.14); the first half handles resource selection by finding the minimal cost network relative to the second half, which handles adaptive slicing of those selected BSs by allocating slices of the BSs to demand for maximum demand satisfaction. Whereas in the stochastic program slicing is

Deterministic Equivalent Program (DEP) of Equations (2.12)–(2.18)

$$\underset{\left\{z_s,\, \delta_{ms}^{(\omega)}\right\}}{\text{minimize}} \left\{ \sum_{s \in \mathcal{S}} c_s \; z_s - \frac{\alpha}{D} \sum_{\omega \in \Omega} p^{(\omega)} \left(\sum_{m \in \mathcal{M}} \sum_{s \in \mathcal{S}} \delta_{ms}^{(\omega)} \; u_{ms}^{(\omega)} \right) \right\} \tag{3.1}$$

subject to:

$$\sum_{s \in \mathcal{S}} \delta_{ms}^{(\omega)} \; u_{ms}^{(\omega)} \leq d_m, \; \forall m \in \mathcal{M}, \; \forall \omega \in \Omega \tag{3.2}$$

$$\sum_{m \in \mathcal{M}} \delta_{ms}^{(\omega)} \leq r_s \; z_s, \; \forall s \in \mathcal{S}, \; \forall \omega \in \Omega \tag{3.3}$$

$$z_s \in \{0,\, 1\}, \; \forall s \in \mathcal{S} \tag{3.4}$$

$$\delta_{ms}^{(\omega)} \in [0,\, d_m], \; \forall m \in \mathcal{M}, \; \forall s \in \mathcal{S}, \; \forall \omega \in \Omega \tag{3.5}$$

performed relative to $\tilde{u}_{ms}$ as a whole in the second stage, in the DEP each scenario receives an independent slicing that maximally allocates demand for that scenario. By considering all scenarios in the sample space and weighting each scenario's impact on the optimization according to its probability of occurrence, the DEP is equivalent to the original program. The competing goals of minimizing VWN cost and maximizing VWN resource allocation to demand are controlled by α. Further, the second half is normalized relative to D—the maximum value of the second stage summation—so that the summations of the second half is restricted to the range [0, 1]; this constrains α to approximate the expected cost of the VWN.

Equation (3.2) is a constraint that ensures that demand is allocated to BSs that are within

range and that demand is not overallocated resources, adapting eq. (2.16) for all demand scenarios. Equation (3.3) is a constraint that ensures only selected BSs are allocated demand, and that BSs are allocated within capacity, combining and adapting eqs. (2.15) and (2.17) for all demand scenarios. Equations (3.4) and (3.5) define bounds on the decision variables according to eqs. (2.13) and (2.18).

As the DEP reformulation is equivalent to the original stochastic optimization program, the solution of the DEP is also a solution of the original. However, while the DEP removes the original's stochastic variables, it introduces a new set, Ω, as an additional dimension to the problem. As Ω has a cardinality of $2^{M \cdot S}$, it represents a significant computational complexity, especially relative to $\mathcal{M}$ and $\mathcal{S}$ (with cardinalities M and S, respectively). Further, $p^{(\omega)}$ is nontrivial to find and varies according to λ_n and the BS locations. While it is possible to solve the DEP directly, it is highly time consuming and scales very poorly with an increase in the number of demand points and BSs.

3.1.1 Sampling the DEP; Sample Average Approximation

As presented, the DEP, while solvable, is ineffective. However, an alternative to the DEP can be found without needing to consider the entire scope of the sample space. By generating a number of random scenarios, Ω is sampled into a set of smaller size, trading accuracy of the solution for manageability.

Let $\hat{\Omega} \stackrel{\text{def}}{=} \{1, 2, \ldots, O\} \subset \Omega$ be this finite, discrete set containing O sampled scenarios

of the sample space. Each scenario in $\hat{\Omega}$ is generated as a single, independent realization of demand points according to the demand intensity field λ_n (i.e., a non-stationary PPP with intensity function λ_n; see Section 2.1 and eq. (2.1)). As a result of this unweighted random sampling, each scenario in $\hat{\Omega}$ has equal probability of occurring. That is, $p^{(\omega)} = 1/o, \forall \omega \in \hat{\Omega}$. By making this change, the sampled deterministic equivalent program (sDEP) (eqs. (3.6)–(3.10)), or sample average approximation (SAA), of the original stochastic optimization program is found.

Sampled Deterministic Equivalent Program (sDEP) of Equations (2.12)–(2.18)

$$\underset{\left\{z_s,\, \delta_{ms}^{(\omega)}\right\}}{\text{minimize}} \left\{ \sum_{s \in \mathcal{S}} c_s \, z_s - \frac{\alpha}{D \cdot O} \sum_{\omega \in \hat{\Omega}} \sum_{m \in \mathcal{M}} \sum_{s \in \mathcal{S}} \delta_{ms}^{(\omega)} \, u_{ms}^{(\omega)} \right\} \tag{3.6}$$

subject to:

$$\sum_{s \in \mathcal{S}} \delta_{ms}^{(\omega)} \, u_{ms}^{(\omega)} \leq d_m, \, \forall m \in \mathcal{M}, \, \forall \omega \in \hat{\Omega} \tag{3.7}$$

$$\sum_{m \in \mathcal{M}} \delta_{ms}^{(\omega)} \leq r_s \, z_s, \, \forall s \in \mathcal{S}, \, \forall \omega \in \hat{\Omega} \tag{3.8}$$

$$z_s \in \{0, \, 1\}, \, \forall s \in \mathcal{S}. \tag{3.9}$$

$$\delta_{ms}^{(\omega)} \in [0, \, d_m], \, \forall m \in \mathcal{M}, \, \forall s \in \mathcal{S}, \, \forall \omega \in \hat{\Omega} \tag{3.10}$$

The differences between the DEP and sDEP formulations are seemingly cosmetic. The objective (eq. (3.6)) and constraints (eqs. (3.7)–(3.10)) are effectively unchanged from the equivalent objective (eq. (3.1) and constraints (eqs. (3.2)–(3.5)) of the DEP. The distinction lies in that finding a solution of the sDEP optimizes for the specific scenarios in $\hat{\Omega}$. By

sampling Ω to form $\hat{\Omega}$, the sDEP formulation introduces solution error as a tradeoff for manageability. The more accurately $\hat{\Omega}$ estimates Ω, the more accurate the optimal solution of the sDEP estimates the solution of the original stochastic program. For a sufficiently large O, $\hat{\Omega}$ contains enough scenarios to represent an arbitrarily tight approximation of Ω. As O increases, the computational complexity of the solution increases exponentially, causing the manageability of $\hat{\Omega}$ and the sDEP to decrease.

3.1.2 Adaptive Slicing

After the solution to the sDEP model has been found, the VNB has determined the joint BS selection that supports the VWNs. It further provides a proposed resource slicing of the BSs to the SPs' demand points, which determine the VWN constructions for those scenarios considered in $\hat{\Omega}$. Each specific realization of demand points requires a proper resource slicing to construct the VWNs.

As a condition for the sDEP model, $\hat{\Omega}$ does not comprehensively cover all scenarios in Ω. For an actual implementation of the sDEP, any observed scenario of demand points is highly unlikely to be in $\hat{\Omega}$, and the sDEP solution would not provide a slicing of the selected resources to construct a VWN for the new scenario. Further, demand point realizations change on a much shorter time scale than the underlying statistics that inform the appropriate resource selection. To find a resource slicing with a new realization in mind using the sDEP model would require resolving the sDEP with the new realization added to $\hat{\Omega}$. This solution would

likely be irrelevant as it could take longer to solve than the realization would exist and provide a resource selection that would be unchanged from the selection provided in the original solution.

Instead, it is valuable to have a model which assigns only an adaptive slicing of the selected resources so the VWNs can adapt according to specific realizations of demand without needing to determine a new joint resource selection. This new adaptive slicing model (eqs. (3.11)–(3.14)) is found by fixing the selected resources, z_s, and only considering the specific demand realization $\omega \notin \hat{\Omega}$ to be sliced to and satisfied.

Adaptive Slicing Model

$$\underset{\{\delta_{ms}\}}{\text{minimize}} \left\{ \sum_{m \in \mathcal{M}} \sum_{s \in \mathcal{S}} \delta_{ms}\, u_{ms} \right\} \tag{3.11}$$

subject to:

$$\sum_{s \in \mathcal{S}} \delta_{ms}\, u_{ms} \leq d_m,\ \forall m \in \mathcal{M} \tag{3.12}$$

$$\sum_{m \in \mathcal{M}} \delta_{ms} \leq r_s\, z_s,\ \forall s \in \mathcal{S}. \tag{3.13}$$

$$\delta_{ms} \in [0,\, d_m]\,,\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S} \tag{3.14}$$

The objective function (eq. (3.11)) is a simplified version of the sDEP objective function (eq. (3.6)). The goal is to choose a slicing of the selected resources to maximize the satisfied demand in the given scenario with fixed resources. The constraints (eqs. (3.12)–(3.14)) are unchanged from the sDEP (eqs. (3.7), (3.8), and (3.10)), except that z_s is a constant and

without $\hat{\Omega}$.

This new formulation is far more tractable than the sDEP formulation, as there is only a single decision variable, δ_{ms}, to solve. Further, as δ_{ms} is not discrete, the adaptive slicing model is a linear programming problem and has better time order classification compared to the sDEP, which is a mixed integer linear programming problem.

With the adaptive slicing model, the first approach to find a solution to the original two-stage stochastic program (eqs. (2.12)–(2.18)) is established. First, as with the stochastic program, an SP coordinates with a VNB to construct it a VWN by communicating the expected demands of its service in the form of a demand intensity function, $\lambda(x, y)$. The VNB concatenates the new SP's demand with the other SPs into $\mathcal{N}$. Second, the VNB finds O, the number scenarios that are required for the sDEP solution to be a tight approximation of the stochastic program. Third, the VNB generates one scenario of demand points with corresponding demands (eqs. (2.1)–(2.3)) for each SP $n \in \mathcal{N}$ according to λ_n and aggregates them into a single set $\mathcal{M}$ (eq. (2.6)). This is repeated until O distinct scenarios are generated, establishing $\hat{\Omega}$ and $u_{ms}^{(\omega)}, m \in \mathcal{M}$, $s \in \mathcal{S}$, $\omega \in \hat{\Omega}$. Fourth, the sDEP model (eqs. (3.6)–(3.10)) is solved using this set of generated scenarios of $u_{ms}^{(\omega)}$, providing a resource selection and optimal slicing for any scenarios in $\hat{\Omega}$. This informs which resources the VNB leases from the RPs. Finally, for any observed realization of demand points, the adaptive slicing model (eqs. (3.11)–(3.14)) is solved with that fixed resource selection, handling VWN construction in the short term. The sDEP is only later used according to longer term changes in the demand statistics, indicating a new resource selection.

3.2 Approach II: The Genetic Algorithm

A major limitation of the approach in Section 3.1 is that the sDEP is increasingly unmanageable as O, S, or M increases. Most importantly, the accuracy of the sampled DEP is directly dependent on the size of $\hat{\Omega}$, O, directly causing a trade off between the accuracy of the sDEP and its computability in a reasonable amount of time. While diminishing returns can be avoided by determining the minimum necessary O for $\hat{\Omega}$ to provide a sufficiently tight estimation, the manageability of the sDEP is still dependent on the size of S and M. In this section, I reformulate the problem of joint BS selection for the VWN as a genetic algorithm (GA), circumventing the need to discretize demand or to sample Ω, thereby simplifying the original problem into a more scalable form. Further, by using the adaptive slicing model of Section 3.1.2, this GA approach can provide an approximate solution of the original stochastic program while only needing to solve for resource selection.

A genetic algorithm is an iterative metaheuristic algorithm inspired by the concept of natural selection in which an approximate solution to a given optimization problem is arrived at via a series of progressive generations. Each generation contains a number of candidate solutions, called individuals, each of which is defined by a chromosome. During a given generation, a fitness heuristic is assessed for each individual based on its chromosome. Then individuals are *selected* at random to become parents, with more fit individuals being selected with higher probability. Parents are then paired off, and each pair of parents may, with probability p_{xov}, undergo a process called *crossover*, a process similar to genetic recom-

bination, in which the parents' chromosomes are mixed to form two individuals (children) for the next generation. If crossover does not occur, the parents are cloned to be their own children for the next generation. The chromosomes of the resulting children then undergo *mutation*, altering the chromosome slightly. Once enough new children have been generated, this new set of individuals forms the next generation to repeat the process. Figure 3.1 is a block diagram that illustrates this general process. Each of these steps is described in further detail below.

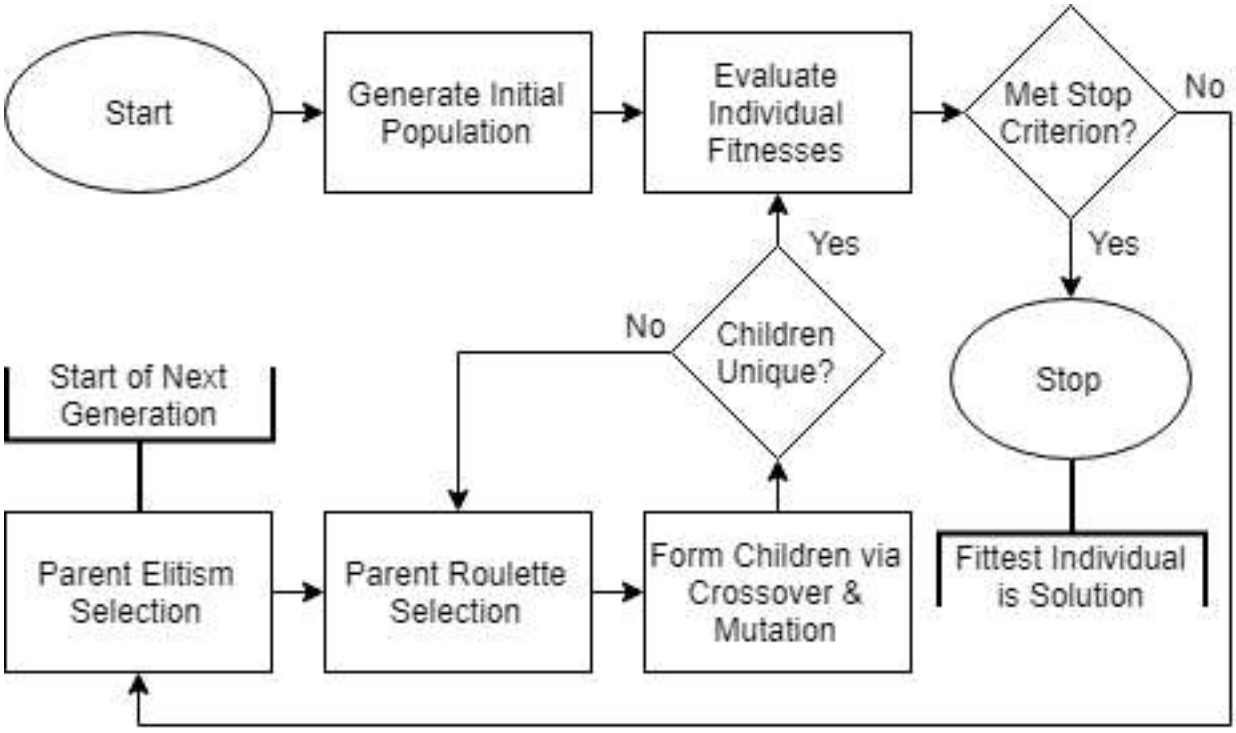


Figure 3.1: Block diagram of the GA approach. This is for a typical genetic algorithm utilizing elitism and enforcing generational uniqueness.

3.2.1 The GA Chromosome

Let $\mathcal{G} \stackrel{\text{def}}{=} \{1, 2, \ldots, G\}$ be the set of generations used in the genetic algorithm and $\mathcal{I}_g \stackrel{\text{def}}{=} \{1, 2, \ldots, I\}, g \in \mathcal{G}$ be the set of individuals within generation g. Each individual $i \in \mathcal{I}_g$ has a binary chromosome $\mathbf{z}^{\{ig\}}$ of length S. $z_s^{\{ig\}}$, $s \in \mathcal{S}$, denoting each individual bit, or gene, of the chromosome, is defined as

$$z_s^{\{ig\}} \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if BS } s \text{ is selected by the VNB for individual } i \text{ in generation } g; \\ 0, & \text{otherwise.} \end{cases} \tag{3.15}$$

Chromosome Fitness

To evaluate the fitness of each chromosome in the GA approach, I assume that all demand over $\mathcal{R}$ is allocated to the closest resource relative to that resource's coverage radius. That is, the demand allocated to a BS $s \in \mathcal{S}$ is all the demand located in the region V_s such that

$$V_s = \left\{ p \in \mathcal{R} \mid \frac{d(p,\, s)}{b_s} \leq \frac{d(p,\, t)}{b_t} \; \forall s \neq t \right\} \tag{3.16}$$

where $d(p,\, s)$ is the euclidean distance between an arbitrary point p and BS s. Let $\mathcal{S}' \subseteq \mathcal{S}$ be the set of resources that have been selected by an arbitrary chromosome (i.e., $z_s^{\{ig\}} = 1$, $\forall s \in \mathcal{S}'$ and $z_s^{\{ig\}} = 0$, $\forall s \notin \mathcal{S}'$ for an arbitrary chromosome $\{ig\}$). By making this assumption,

$\mathcal{S}'$ partitions $\mathcal{R}$ into a multiplicatively weighted Voronoi tessellation [37, 38] using the point locations of the selected BSs as the defining sites and BS coverage radii, b_s, $s \in \mathcal{S}'$, as weights; if all BSs in $\mathcal{S}'$ have homogeneous coverage radii (i.e., $b_s = b$, $\forall s \in \mathcal{S}'$), this partition is a standard unweighted Voronoi tessellation.

The region within the coverage radius of BS s is denoted by B_s. It is desired that V_s be wholly contained within B_s (i.e., $V_s \subseteq B_s$), but this is not guaranteed by the Voronoi tessellation assumption. If V_s is not wholly contained within B_s, BS s is considered to be *overcoverage*. The total demand allocated to a selected resource $s \in \mathcal{S}'$ is $\sum_{n \in \mathcal{N}} \iint_{V_s} \lambda_n (x,\, y)\, dx\, dy$. If the total demand allocated to BS s exceeds B_s, BS s is considered to be *overcapacity*.

The fitness heuristic of each individual chromosome, $z^{\{ig\}}$, is assessed as the reciprocal of the chromosome's cost

$$\text{fitness}\left(z^{\{ig\}}\right) = \frac{1}{\text{cost}\left(z^{\{ig\}}\right)} \tag{3.17}$$

The cost of a given chromosome is the sum of the direct costs of the individual selected BSs and their associated overcoverage (c_{cov}) and overcapacity (c_{cap}) costs

$$\text{cost}\left(\mathbf{z}^{\{ig\}}\right) = \sum_{s \in \mathcal{S}} \Bigg(c_s\, z_s^{\{ig\}} + c_{\text{cov}}\, \mathbb{1}_{\{V_s \not\subseteq B_s\}} + \left(c_{\text{cap}}^{g} - 1\right)\, \max\left(0,\, \sum_{n \in \mathcal{N}} \left(\iint_{V_s} \lambda_n (x,\, y)\, dx\, dy \right) - r_s \right) \Bigg) \tag{3.18}$$

where $\mathbb{1}_{\{*\}}$ is the indicator function (eq. (2.19)).

Resources that are overcoverage are not desired; solutions that assume some demand is unallocatable by being out of range should be consider less-than-feasible or completely infeasible from the outset. In contrast, the overcapacity cost starts at zero and grows exponentially with each successive generation. For early generations, this allows for inefficient or infeasible solutions to temporarily exist to seed later generations and improve generational diversity. This increases the probability that the final solution (a local maxima) is the global maximum or closer to the global maximum of the fitness function.

It should be explicitly noted that the fitness (cost) of a given genome is monotonically non-increasing (non-decreasing) over $g \in \mathcal{G}$; a genome's fitness is constant over g if the genome is not overcapacity, and strictly decreases every generation if the genome is overcapacity.

3.2.2 Forming the Next Generation

Once the fitness calculation of all of the individuals of the current generation has been performed, the next generation begins to form.

Elitism and Selection

Generating the next generation begins with selection. First, I use elitism selection. Under elitism selection, a defined number of the fittest individuals from each generation is selected as that generation's elitist members. These elitist members, the n fittest individuals of the

generation, are cloned into the next generation without undergoing the genetic operators of crossover or mutation. Through elitism, I ensure that the fittest individual of a given generation is not less fit than the previous generation's fittest individual. The only exception comes from if the fittest individual's fitness decreases from inefficiency cost, but these are eventually removed as the overcapcity cost increases.

After elitism, standard selection occurs using roulette-wheel selection. In roulette-wheel selection, the fitness of each individual to normalized to the total fitness of the generation and organized into a list. From this list, a new list of the cumulative fitnesses is generated such that the cumulative fitness of individual i is $\sum_{j=1}^{i} \text{fitness}\left(z^{\{jg\}}\right)$; the cumulative fitness of individual I (i.e., the last individual in the generation) is thereby 1. A random number, $P \sim \mathcal{U}(0,\, 1)$, is then generated. The selected individual (or parent) is the first individual in the list that has a cumulative fitness value larger than P.

Under roulette-wheel selection, parents are each randomly selected with a probability given by

$$\text{P}\left(\text{individual } i \text{ of gen. } g \text{ is selected for gen. } g+1\right) = \frac{\text{fitness}\left(z^{\{ig\}}\right)}{\sum_{i \in \mathcal{I}} \text{fitness}\left(z^{\{ig\}}\right)} \tag{3.19}$$

Crossover

Once selection has occurred, the individuals selected via roulette-wheel selection are paired off. With a probability of p_{xov}, the pair undergoes crossover to form their two children which go on to the next generation; otherwise, the parents are cloned as their own children. During crossover, the chromosomes of each pair of parents are mixed. Typical forms of crossover include single-point, k-point, and uniform crossover, though there are other more exotic forms that are application specific.

I use uniform crossover as it has been suggested [39] to be one of the best crossover operators in terms of general performance. With uniform crossover, each gene in the chromosomes of the children are independently chosen from the parents' genomes with equal probability (i.e., 50%) [40]. A random binary mask of length S is generated. The first child inherits genes from the first parent where there is a 1 in the mask and from the second parent where there is a 0; the second child inherits the inverse.

Compared to single-point crossover, uniform crossover is also suggested to increase exploration of the search space as it severs the positional bias (linkage) of the bits in the genome [41]; bits with possible spatial significance to one another no longer undergo crossover in segments like they would in single or k-point crossover.

Mutation

Once each child has been generated, each bit within the child's genome has an independent chance, p_{mut}, to mutate. When a bit mutates, the bit flips from 1 to 0, or from 0 to 1. Typically, the chance of each bit to mutate is $p_{\text{mut}} = 1/l$, where l is the length of the chromosome, which provides an average of one bit mutation per child [42, 43]. For $\mathbf{z}^{\{ig\}}$, $p_{\text{mut}} = 1/s$.

Uniqueness

In a traditional genetic algorithm, once mutation has been completed, the next generation of children has been completely generated. However, it is possible for several children to be identical, reducing the overall genetic diversity of the generation and decreasing the algorithm's exploration of the search space. To counter this possibility I enforce the "uniqueness condition", which is the assurance that each generation must be comprised of unique individuals. If there are any individuals that are identical to others in the generation, all but one of the identical individuals are discarded from the generation. The vacancies are then filled by a new set of generated children

3.2.3 Stopping the GA

The GA iterates for a specified maximum number of generations G. However, once the GA converges to a stable solution, it is worthwhile to halt the GA early to conserve time.

There are two halting conditions that control this. First, the *fitness halt* condition halts the GA if the fitness of the fittest individual remains unchanged for a number of sequential generations, G_{fit}. Due to the fitness function's monotonicity, a solution under this condition is not overcapacity. This halting condition is expected to provide a solution that is feasible to provide nearly 100% demand satisfaction; a solution provided under fitness halt can be overcoverage, but being overcoverage should be unlikely if c_{cov} is sufficiently high and if $\mathcal{S}$ is not too sparse to satisfy $\mathcal{R}$.

The second halting condition is the *genome halt*. Under this condition, the GA halts early if the genome of the fittest individual remains unchanged for a number of sequential generations, $G_{\text{gen}} \geq G_{\text{fit}}$. Due to the fitness function's monotonicity, a solution under this condition is overcapacity; unlike the fitness halt condition, this allows for an infeasible (i.e., overcapacity) solution to evolve, especially when no solution is feasible. The genome halt primarily contrasts with the fitness halt by forcing more generations to occur. This allows for enough generations to pass so the overcapacity cost increases sufficiently that a hypothetical more feasible (i.e., less overcapacity) solution becomes more fit than the less feasible solution, and for enough generations to pass that it is statistically probable that the more feasible solution evolves. Otherwise, an individual that satisfies the fitness halt condition also satisfies the genome halt condition; the fitness halt condition is a more strict form of convergence.

For both of these halting conditions, a minimum number of generations, $G_{\text{min}} < G$, must first pass. Further, the fitness' of all potential solutions one Hamming distance[1] from the

[1]That is, all chromosomes that differ from the genome of the fittest individual by one bit.

individual that caused the halt are computed; if any of these solutions is more fit, the genetic algorithm continues with the new fittest solution passed to the next generation through elitism. Once any halting condition occurs, the fittest individual of the final processed generation is the solution, with $\mathbf{z}^{\{ig\}}$ of the solution determining z_s, $s \in \mathcal{S}$.

In each case, the chromosome of the fittest individual when the GA halts is an approximate solution for the VNB's resource selection, z_s. As the GA only handles the resource selection, it only represents a solution of the first stage (eqs. (2.12) and (2.13)) of the original stochastic optimization program (eqs. (2.12)–(2.18)). That is, while joint resource selection has occurred and the process was informed by the demand to be satisfied (i.e., λ_n), the selected resources still need to be sliced to determine the allocations of those resources to specific VWNs. This process of solving the second stage of the original optimization program can be adaptively handled to a hypothetical or real set of demand points using the adaptive slicing model (eqs. (3.11)–(3.14)) of Section 3.1.2.

The process of applying the GA approach is similar to the process used to apply the DEP approach of Section 3.1. An SP coordinates with a VNB, communicates its demands in the form of a demand intensity function, $\lambda(x, y)$, and the VNB concatenates the new demand into $\mathcal{N}$. Then, the GA processes over λ_n, $n \in \mathcal{N}$ until either halting condition occurs or the maximum number of generations is reached. The solution provided by the GA indicates the VNB's joint resource selection. Finally, for any observed realization of demand points, the adaptive slicing model (eqs. (3.11)–(3.14)) is solved with that fixed resource selection, handling VWN construction in the short term. The GA is rerun later as λ_n changes for long

term VWN construction.

Chapter 4

Simulations and Results

In this chapter I evaluate the effectiveness of the sDEP (Section 3.1) and GA (Section 3.2) approaches as approximations of the two-stage stochastic optimization program presented in Section 2.2. These approaches are set up as a series of simulations and compare the costs, demand satisfaction, computation time, and general trends. First, a set of preliminary results to evaluate the appropriateness of the simulations are presented. Two cases that represent the needs of an urban cellular network SP and the available resources to satisfy those needs are simulated; one case considers RPs with homogeneous resources while the other case considers RPs with heterogeneous resources.

General Setup

Unless stated otherwise, simulations are configured with the following assumptions. BS locations are modeled as a stationary PPP with a fixed number of BSs. Demand point locations are generated independently for each SP $n \in \mathcal{N}$ in each scenario as a non-stationary PPP using $\lambda_n(x, y)$, $x \in [0, X]$, $y \in [0, Y]$, as the spatial intensity function, as described in Section 2.1 (eq. (2.1)). Demand points have homogeneous demand (eqs. (2.2) and (2.3)). To compute $u_{ms}^{(\omega)}$, it is assumed that there are perfect links between demand points and BSs within the BSs' coverage radii (eq. (2.5)). To compute the fitness function for the GA approach (eqs. (3.17) and (3.18)), λ_n is discretized into a grid of equal-sized square pixels with a given side length. The demand of a given pixel is the product of λ_n at the center of the pixel and the area of the pixel. The demand allocated to a selected resource $s \in \mathcal{S}' \subseteq \mathcal{S}$ is the sum of the demand of all pixels within V_s, the resource's Voronoi cell.

Simulations were run on an Intel Core i7-4790K 4.00 GHz 4 real/8 virtual core CPU with 16 GB of DDR3 RAM. CPLEX [44] was used to solve the sDEP model (eqs. (3.6)–(3.10)) and the adaptive slicing model (eqs. (3.11)–(3.14)), and MATLAB was used to implement the GA and to generate the simulated region. During the simulations, extraneous processes were culled to allow maximal use of computer resources. For a given comparison to the sDEP, the GA results were provided from 50 independent runs using an identical data set except for the set of initial individuals; 95% confidence intervals were computed via the bootstrap method. The sDEP approach was solved across multiple values of α.

4.1 Preliminary Simulations

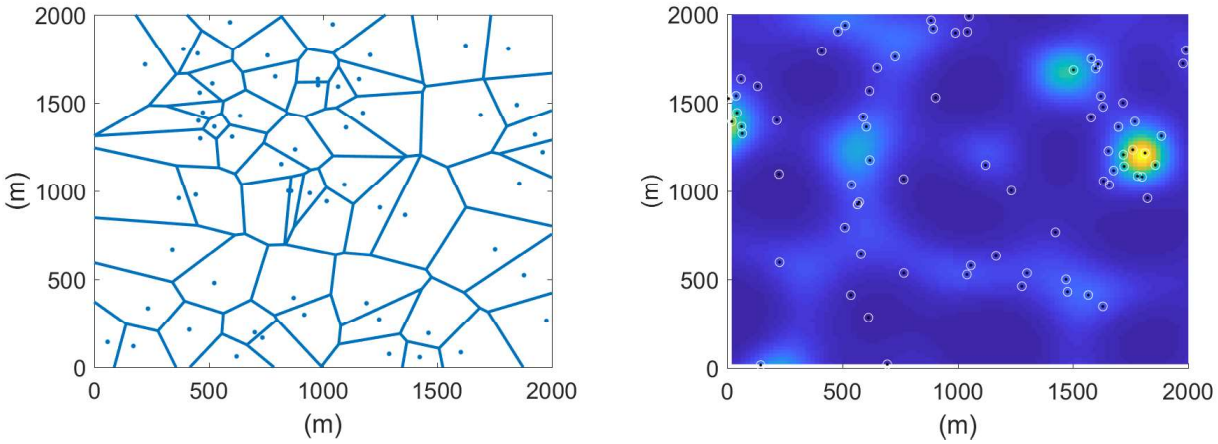
This first considered problem is an initial simulation testing the performance of the sDEP and GA approaches for a general problem intended to be solved in less than 15 minutes. This test was intended to show that the two approaches work correctly and present preliminary results to inform later test cases. Initial results were compiled and collated in a forthcoming paper accepted to the 2019 International Conference on Computing, Networking and Communications (ICNC) [45] and expounded on here.

4.1.1 Setup

For these preliminary simulations, I use the parameter values shown in Table 4.1. There is only one SP, and its demand intensity function, λ, is generated using default location and scale parameters for the SSLT model ($\sigma = 0$, $\mu = 1$). λ is pixelated into a 100 by 100 grid of square pixels with 20 m side length; $\mathcal{R}$ is a region 2 km by 2 km in size illustrated in Figure 4.1. The first listed maximum angular frequency, $\omega_{\max}$, is expressed in units of cycles per pixel; in terms of absolute distance, $\omega_{\max}$ is scaled according to the side length (i.e., absolute $\omega_{\max} = \pi/300$ cycles per meter) and parenthetically stated. Resources are homogeneous and constructed to satisfy the region's demands both in terms of coverage and capacity.

Table 4.1: Numerical Values of Relevant Parameters for Preliminary Simulations

Width, Length of Geographic Area (X)	2 km x 2 km
Number of BSs (S)	60
Number of Demand Points (M)	75
Number of SPs (N)	1
Number of Sampled Scenarios (O)	$\{5, 10, \ldots, 50\}$ and 25
sDEP Objective Weights (α)	$\{5, 10, \ldots, 100\}$ $\{2.2e2, 4.6e2, 1.0e3, \ldots, 1.0e9\}$
BS Cost (c_s, $\forall s \in \mathcal{S}$)	1
BS Capacity (r_s, $\forall s \in \mathcal{S}$)	1.50 Mbps
BS Range (b_s, $\forall s \in \mathcal{S}$)	500 m
Point Traffic Demand (d)	0.178 Mbps
SSLT Approximation Depth (L)	50
SSLT Maximum Angular Frequency ($\omega_{\max}$)	$2\pi/30$ $(\pi/300)$
SSLT Location Parameter (σ)	0
SSLT Scale Parameter (μ)	1
Maximum Number of Generations (G)	3000
Minimum Number of Generations ($G_{\min}$)	300
Unchanged Generations Before Fitness Halt (G_{fit})	150
Unchanged Generations Before Genome Halt (G_{gen})	400
Number of Individuals per Generation (I)	80
Number of Elite Individuals per Generation	4
Probability of Crossover (p_{xov})	0.7
Probability of Mutation per bit (p_{mut})	$1/s = 0.0167$
Overcoverage Cost (c_{cov})	3
Overcapacity Cost (c_{cap})	1.015
Pixel Grid Size	100 x 100, 20 m
Number of Additional Evaluation Scenarios ($\notin \hat{\Omega}$)	50
Number of Additional Scenario Demand Points (M)	200
Additional Demand Point Demand (d)	66.7 kbps



(a) Voronoi tesselation of BS locations (b) λ with one realization of demand points ($\mathcal{M}$)

Figure 4.1: Illustration of resources and demand in the considered region, $\mathcal{R}$, used for the preliminary simulations.

4.1.2 Results

In this section I present the results of these preliminary simulations. First, run time results of the two approaches will be presented and comparisons made. Second, the network cost results will be presented and compared, followed by, third, the network demand satisfaction results. Finally, this section will end with my conclusions of these simulations

Run Time

Figure 4.2 shows the run time trend of the GA approach as a histogram PDF. Across 50 independent simulations, the GA approach converges to a solution in 94 to 209 seconds, averaging 125.9 seconds (CPU time) with a 95% confidence interval of (118.4, 135.4). These results are heavily weighted towards the low end, with more than 20% of simulations terminating in under 100 seconds.

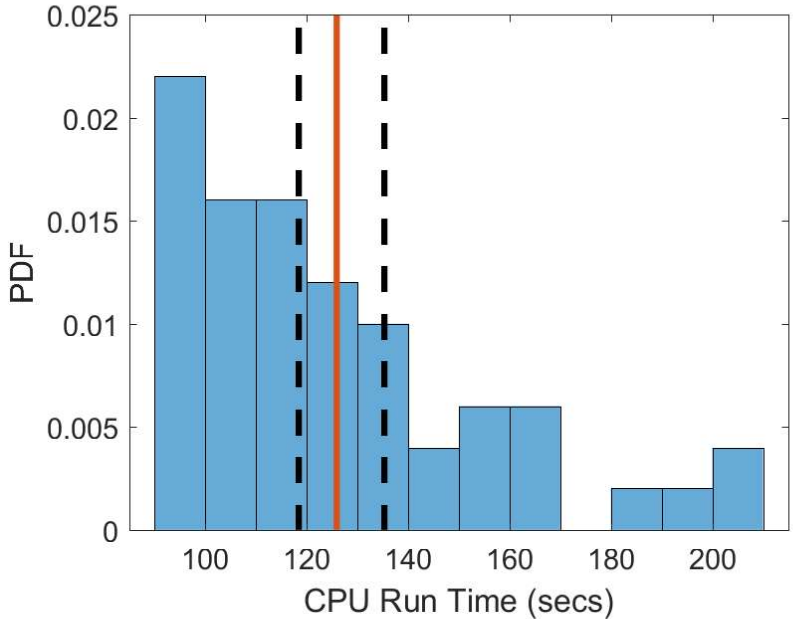


Figure 4.2: GA approach CPU run time PDF over 50 runs. Mean (125.9 secs) shown in orange, 95% confidence interval (118.4, 135.4) shown in dashed black.

Figure 4.3 compares the sDEP run time with the GA run times shown in Figure 4.2 in terms of absolute CPU and approximate wall clock run times and the GA approach's mean speedup ratio. For each set of scenarios, the sDEP approach was performed five times to find mean run times and associated confidence intervals. As the number of scenarios sampled increased, the CPU time of the sDEP increased exponentially. The sDEP failed to converge to a final solution with 50 scenarios within the time limit of 15 minutes (wall clock time), and is not plotted in Figure 4.3. Except for the trivial five scenario case—which allocated zero BSs to the VWN—the GA approach converged to a solution in less CPU time than the sDEP. When 20 scenarios are sampled, the GA provided a 4.4x speedup over the sDEP approach; at 45 scenarios, this increased to 46.3x.

CPLEX is capable of parallelizing across the PC's 8 CPU cores, allowing for the sDEP

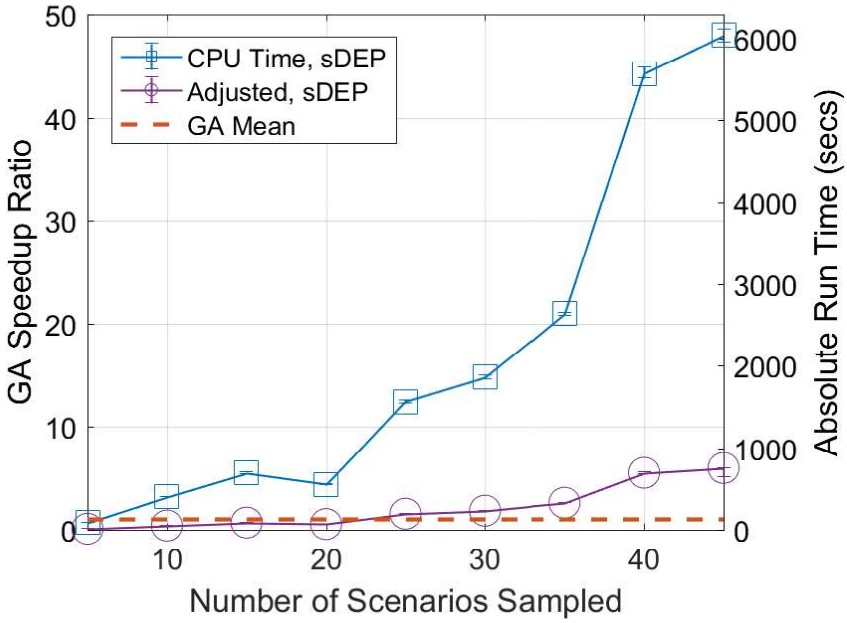


Figure 4.3: Run time comparison of the GA and sDEP approaches with $\alpha = 20$ and varying numbers of sampled scenarios; sDEP confidence intervals shown as errorbars, GA confidence intervals too small to display

approach's wall clock run time to be, at minimum, one-eighth the CPU run time[1]. This theoretical wall clock limit is displayed as the sDEP's adjusted time. In terms of this adjusted wall clock time, the sDEP is more competitive to the GA. However, the GA converges in less wall clock time for 25 or more sampled scenarios.

Figure 4.4 provides a comparison of the run time trend for the sDEP approach for varying α and a fixed ($O = 25$) number of scenarios. Except for the range $\alpha \in [25, 45]$, the simulations show a fairly consistent run time requirement for the sDEP approach. Due to this, α is not a primary concern to compare the GA run time to the sDEP; for this setup

[1]Experimentally, the CPU time was approximately 7.1x that of the wall clock time, not 8x. The 50 scenario case (not plotted) did not complete in 15 minutes wall clock time (ideally 7200 seconds CPU time; 8 threads of 900 seconds each), but occupied only 6,367 seconds CPU time before the time limit expired.

the GA provided a speedup ratio of approximately 10.1x. However, the range $\alpha \in [25, 45]$ is significant, as, within the region, CPLEX approaches a critical point in the optimization of the sDEP model. Within this range, the optimal resource selection is less immediately apparent to the branch-and-cut algorithm CPLEX uses.

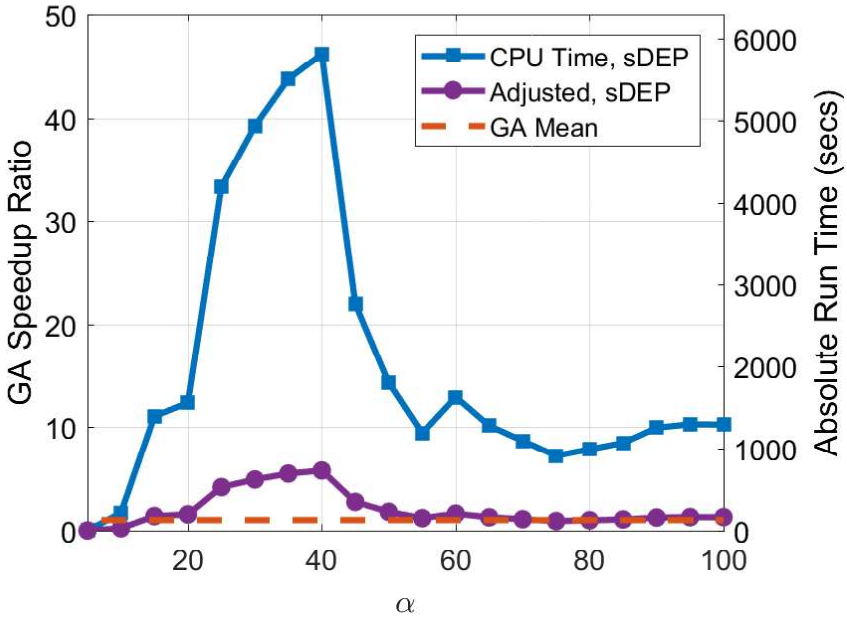


Figure 4.4: Run time comparison of the GA and sDEP approaches with 25 scenarios and varying values of α

Network Cost

The spike of CPU run times corresponds to the regions of α that the sDEP solution is in transition, where the optimal number of resources selected is more difficult for the CPLEX branch-and-cut algorithm to determine. This can be seen in Figure 4.5, where the run time spike of $\alpha \in [25, 45]$ surrounds the transition between the selection of 9 BSs and 10 BSs. CPLEX converges to a solution ($\alpha < 25$, 9 BSs; $\alpha > 45$, 10 BSs) within approximately

1000 seconds. However, within this critical region, the CPLEX branch-and-cut algorithm is unable to cut many close-to-feasible solutions, increasing run time up to 6000 seconds.

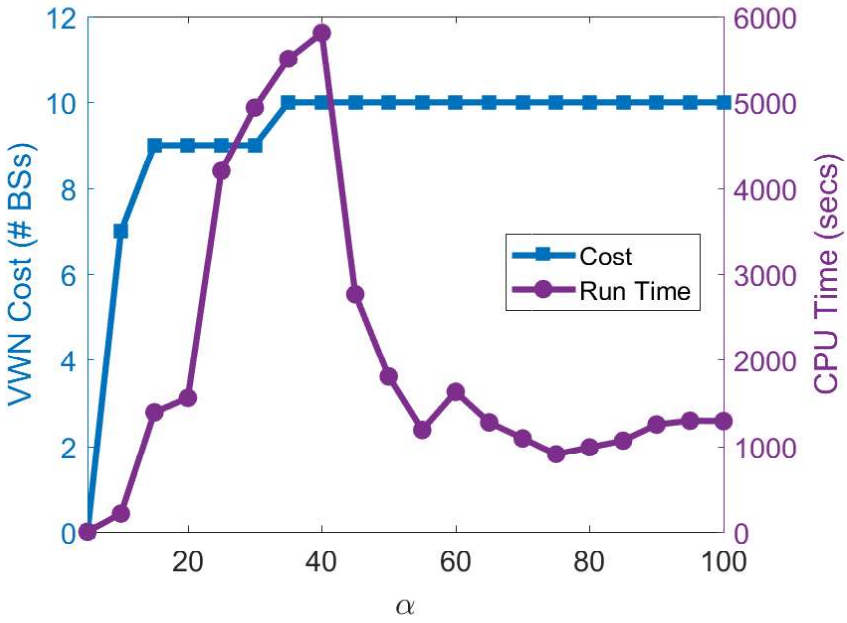


Figure 4.5: Comparison of sDEP CPU run time and VWN construction cost

This trend continues for higher values of α. As shown in Figure 4.6, the region surrounding the transition between selecting 11 and 12 BSs ($\alpha \in$ [2.2e2, 4.6e3]) has a wider spike before settling down to easier solutions, including the trivial, all-BS case for $\alpha \geq$ 1.0e6. It should be noted that for $\alpha \in \{$4.6e2, 1.0e3$\}$ the sDEP model did not complete within the 15 minute (wall clock) time limit, and the solution presented for those values is not necessarily the most optimal solution for the problem, only the most optimal solution found.

The GA approach settles on solutions with four different values of network cost. Figure 4.7 visualizes these four types of solutions as a histogram PDF of the 50 GA simulations. Half of the simulations ended with a selection of 13 BSs. However, almost half ended with 14

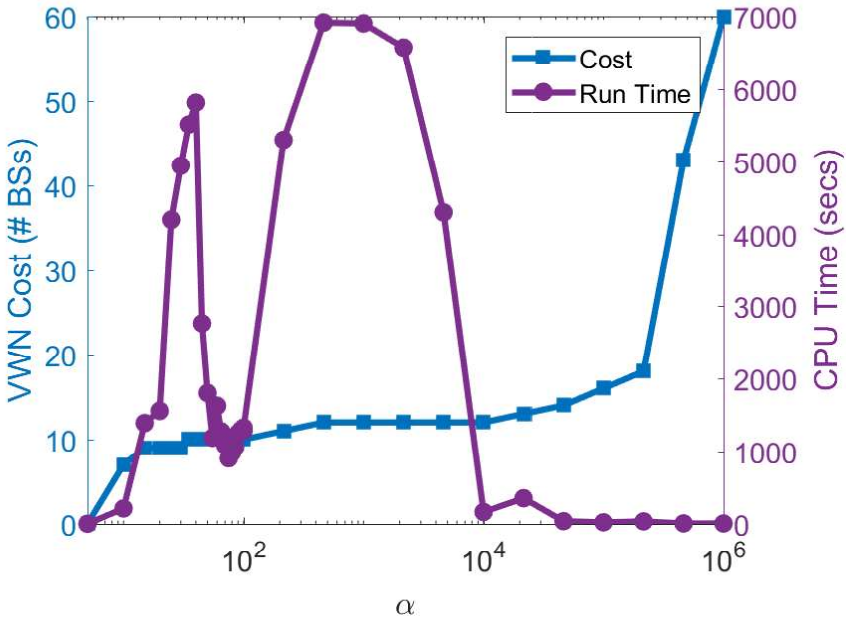


Figure 4.6: Comparison of sDEP CPU run time and VWN construction cost for further values of α.

or 15 selected BSs as the optimal selection; this implies that the GA is terminating too early for the solutions to properly converge to the fitter 13-BS solution. This is further implied by the single 12-BS solution, which, like all of the other solutions, is neither overcoverage nor overcapacity. The 12-BS solution is completely feasible and more fit than all 49 other solutions.

Figure 4.8 plots the costs of the various GA solutions against their CPU run times. There is no apparent trend in the plot; a linear line of best fit would be sloping upwards, implying that more time would allocate more resources. However, the 12-BS solution (189.9 seconds) is far above the mean run time (125.9 seconds), while most 13-BS solutions are below. The GA discovers an easy, weak solution that selects 13 BSs fairly quickly and halts before it has a reasonable chance to discover the fitter but more difficult-to-find 12-BS solution. The GA

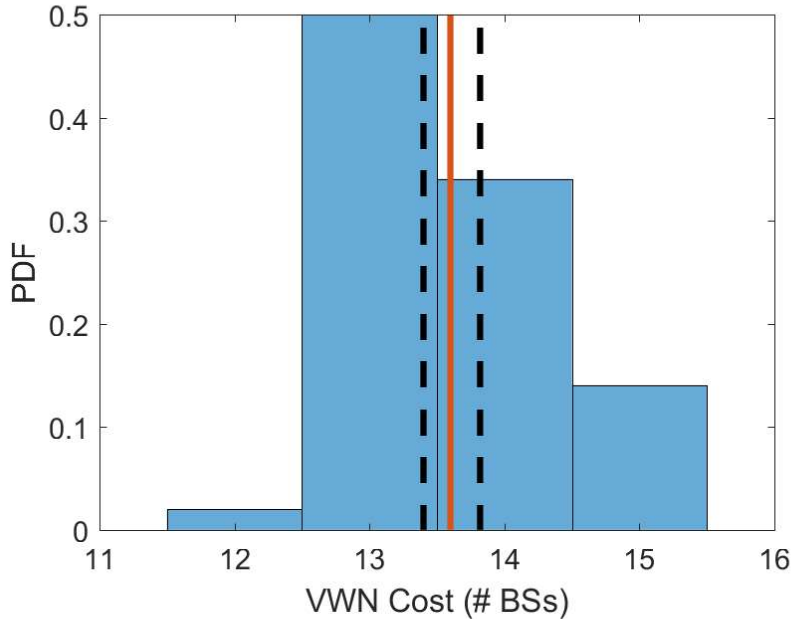


Figure 4.7: PDF of constructed VWN costs of the GA approach. Mean (13.6 BSs) shown in orange, 95% confidence interval (13.4, 13.82) shown in dashed black.

run time will be extended in the proper test cases (Sections 4.2 and 4.3) to avoid premature halting.

The trade off for the GA's improved run time is that the solution provided is expected to be less optimal than the sDEP, as indicated by decreased demand satisfaction for the constructed VWN compared to the sDEP for similar cost networks. Figure 4.9 compares the cost of the sDEP as α increases with the cost of the GA solutions. It should be noted that, since the constructed networks are on the order of ten BSs, each addition or removal of a BS incurs a large amount of cost error.

Differing ranges of α provide distinct solutions to compare against the GA, notably: $\alpha \leq$ 2.2e2, where the sDEP provides a solution of less cost than the GA; $\alpha \in$ [4.6e2, 1.0e4],

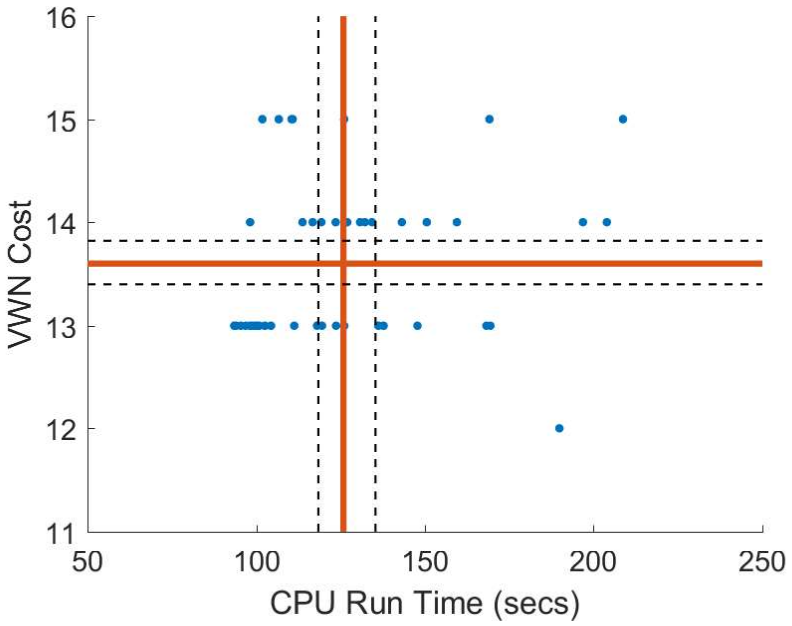


Figure 4.8: Various GA simulation solution costs and associate CPU run times. Means shown in orange, 95% confidence intervals shown in dashed black.

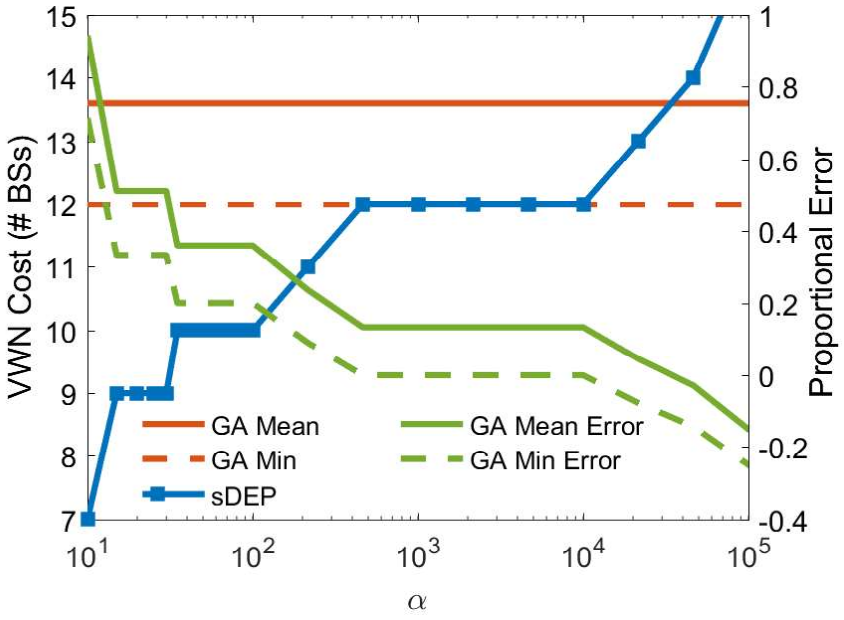


Figure 4.9: Constructed VWN Costs for sDEP and GA approaches.

where the sDEP provides a solution of equal cost to the fittest GA solution; $\alpha \in [2.2e4, 4.6e4]$, where the GA provides solutions around the mean of the GA; and $\alpha \geq 1.0e5$, where the sDEP provides solutions that are more costly than the GA. Of these, the first two ranges are of the most interest. The first range generates networks with lower cost than the GA and is expected to provide less demand satisfaction. Alternatively, the second range generates a network with equal cost to the GA and should provide equal or greater demand satisfaction.

Demand Satisfaction

There is a direct correlation between the number of BSs in the VWN and its capability for satisfying demand. As the cost increases for a given solution, the average demand satisfaction trends towards 100%. This trend for the sDEP is depicted in Figure 4.10 and Table 4.2.

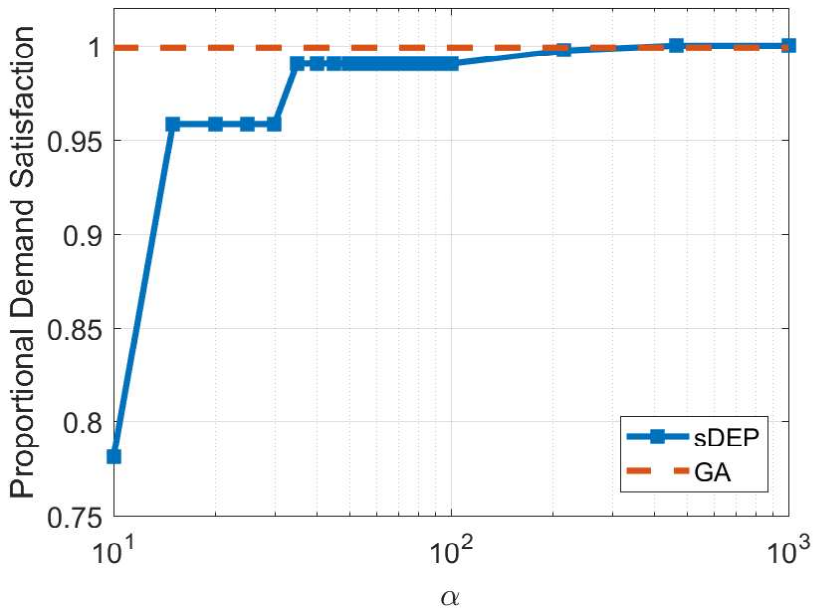


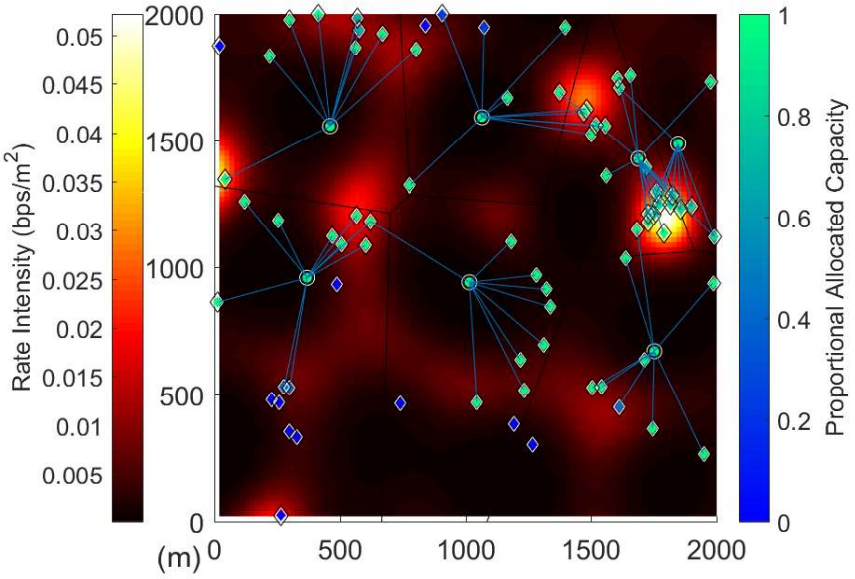
Figure 4.10: Demand satisfaction of various sDEP constructed VWns.

Table 4.2: Demand Satisfaction Results for Various sDEP-Constructed VWNs

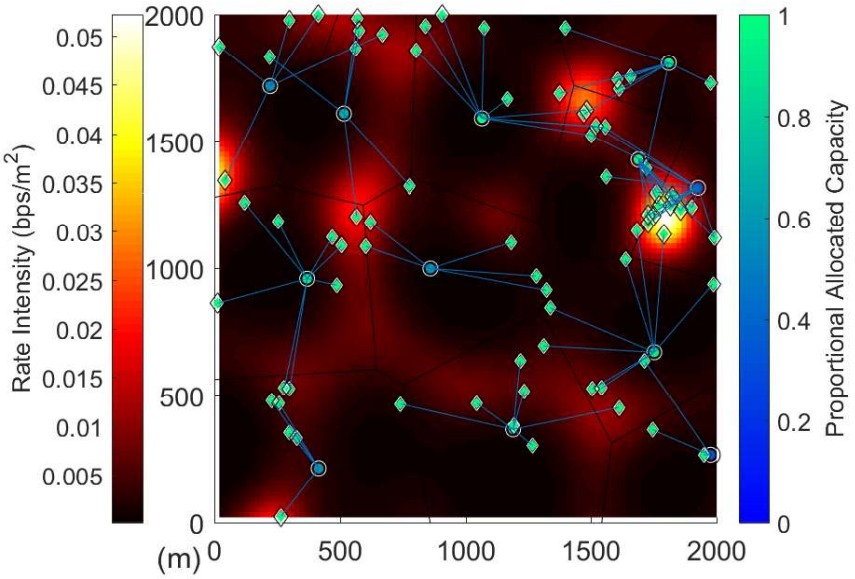
Number of Selected BSs	**Demand Satisfaction**
0	0.00%
7	78.1%
9	95.8%
10	99.0%
11	99.7%
12+	100%

Demand satisfaction approaches 100% until reaching it for all sDEP solutions containing 12 or more BSs. This covers the latter three ranges described while discussing VWN cost (i.e., $\alpha \in [4.6e2,\ 1.0e4]$, $\alpha \in [2.2e4,\ 4.6e4]$, $\alpha \geq 1.0e5$), which generate 100% demand satisfaction for the scenarios that they were optimized for. That is, VWNs constructed by the sDEP that have construction costs greater than or equal to the GA solutions have perfect demand satisfaction. Figure 4.11b is an example (12-BS) solution from this category, which shows that all demand points for the shown demand point scenario are sliced to and completely satisfied. In this solution, several individual BSs, such as the one in the middle or in the bottom right, have additional available capacity, and several BSs, such as the ones in the upper right near the highest demand, provide highly overlapping coverage.

The first range described (i.e., $\alpha \leq 2.2e2$) is wholly described by solutions that provide less than 100% demand satisfaction, but quickly approach that value with diminishing returns. What this range most importantly shows is that there does not exist, at least as far as the sDEP is an approximation of the original two-stage stochastic optimization program, any solutions that give equal demand satisfaction for less cost than the GA's solutions. The most extreme of these solutions (7 BSs at $\alpha = 10$ with 78.1% demand satisfaction) is shown



(a) 7-BS Solution (78.1% Demand Satisfaction; $\alpha = 10$)



(b) 12-BS Solution (100% Demand Satisfaction; $\alpha \in$ [4.6e2, 1.0e4])

Figure 4.11: sDEP resource selection and slicing for one scenario in $\hat{\Omega}$. Circles indicate BSs and diamonds indicate demand points. Lines connecting BSs to demand points indicate that the BS is sliced to the demand point (i.e., that a link between the BS and demand point exists). The color filling a BS indicates the proportion the BS's capacity is sliced, and the color filling a demand point indicates the proportion of the demand point's demands that have been satisfied.

in Figure 4.11a. Here, all BSs are fully allocated, with several demand points being completely unsatisfied without a link to any BS. Not immediately apparent is that the unlinked demand points along the bottom of the region are unlinked because they are not in range of any BS.

One notable observation is that the sDEP provided a non-trivial solution at approximately $\alpha = 10$, but not for $\alpha = 5$. The second summation of the sDEP, which handles slicing and demand satisfaction, is normalized according to the total expected demand of the field, D. As a result, if a hypothetical constructed network provided 100% demand satisfaction, the value of the second half of the sDEP objective function would be α. For a non-trivial solution to be optimal, the satisfaction of the demand within the scenarios comprising Ω must be larger than the cost of the network divided by α. That is, for $\alpha = 5$, there must be a BS that provides 20% demand satisfaction per unit cost for a non-trivial solution to exist. Each BS has a rate capacity of approximately 11% of the total demand contained in the field; $\alpha = {}^{1.5e6}/_{75 \cdot 1.78e5} = {}^{15}/_{13.35} = 8.9$ is the lowest possible value of α that provides a non-trivial solution.

The GA approach performs fairly well, exceeding 99% demand satisfaction for all solutions. Figure 4.12 visualizes the distribution of the various GA solutions. Forty-five of the fifty GA solutions have at least 99.8% demand satisfaction, averaging 99.90% with a 95% confidence interval of (99.83%, 99.94%). Demand satisfaction for the 13-, 14-, and 15-BS solutions is dominated by solutions giving 100% demand satisfaction. See Table 4.3 for specific ranges of the GA solutions.

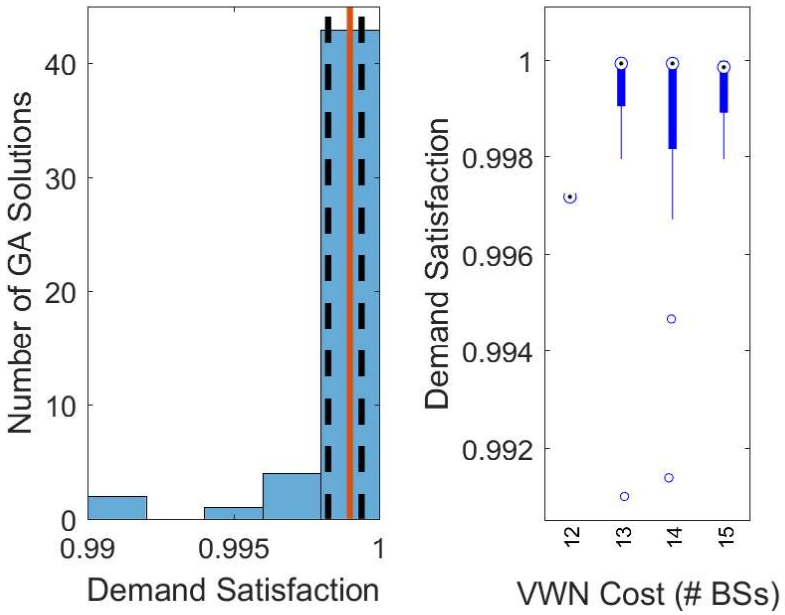


Figure 4.12: Proportional demand satisfaction of the GA approaches as a histogram and box-and-wisker plot. Histogram mean shown in orange, 95% confidence intervals shown in dashed black. Black dots indicate medians, blue bars the inner quartiles (25%–50% and 50%–75%), blue lines the outer quartiles (0%–25% and 75%–100%), and empty circles the outliers.

The sDEP approach is able to provide a 12-BS solution with 100% demand satisfaction. This solution is of equal cost to the cheapest GA solution, which had a loss or error of 0.3% demand satisfaction but was solved in 190 seconds for a 36.4x speedup. More expensive and common GA solutions provide a solution averaging within 0.1% demand satisfaction with the addition of 1 additional resource (an error of 8.33% additional cost); half of these solutions have 100% demand satisfaction, equal to the 12-BS sDEP solutions. 11-BS sDEP solutions provide similar demand satisfaction (i.e., 99.7%) to the 12-BS GA solution; comparing similar demand satisfactions shows the GA introducing 1 BS of error, or 9.09% additional cost.

Each sDEP solution is an optimization for the specific scenarios in $\hat{\Omega}$ that were used to

Table 4.3: Demand Satisfaction Results for Various GA-Constructed VWNs

Number of BSs	**Number of Solutions**	**Min**	**Avg**	**Max**
12	1	99.72%	99.72%	99.72%
13	25	99.10%	99.92%	100.0%
14	17	99.14%	99.86%	100.0%
15	7	99.79%	99.94%	100.0%

generate the solution. This is inherently an unfair comparison for the GA, as the GA is intended to be optimized for λ_n, not the scenarios of $\hat{\Omega}$. By considering new scenarios that are not in $\hat{\Omega}$, it is expected for the sDEP solutions to no longer perform as optimally, and the two approaches can be more appropriately compared. Figure 4.13 visualizes the proportional demand satisfaction for both the sDEP and GA approaches when sliced to a new set of 50 scenarios as evaluated by the adaptive slicing model of Section 3.1.2. The new scenarios are independently generated using the same method as the original scenarios, but with 200 demand points per scenario, each with 66.7 kbps rate demand.

For the range of $\alpha \leq 100$, the sDEP solutions perform slightly better for the new scenario sets compared to the scenarios they were optimized for. These larger scenarios distribute demand more thoroughly, allowing the sDEP solutions to be capable of marginally higher demand satisfaction. However, for all points $\alpha > 100$, these new scenarios cause demand satisfaction to slightly decrease; a closer look over this range is shown in Figure 4.14.

For the range of $\alpha > 100$, this indicates that near the boundary of 100% demand satisfaction, the scenarios are not a complete representation of the original demand field, λ_n. Each scenario has more demand points, which more evenly spread the demand over $\mathcal{R}$. The several 12-BS sDEP solutions which provided 100% demand satisfaction for the scenarios in

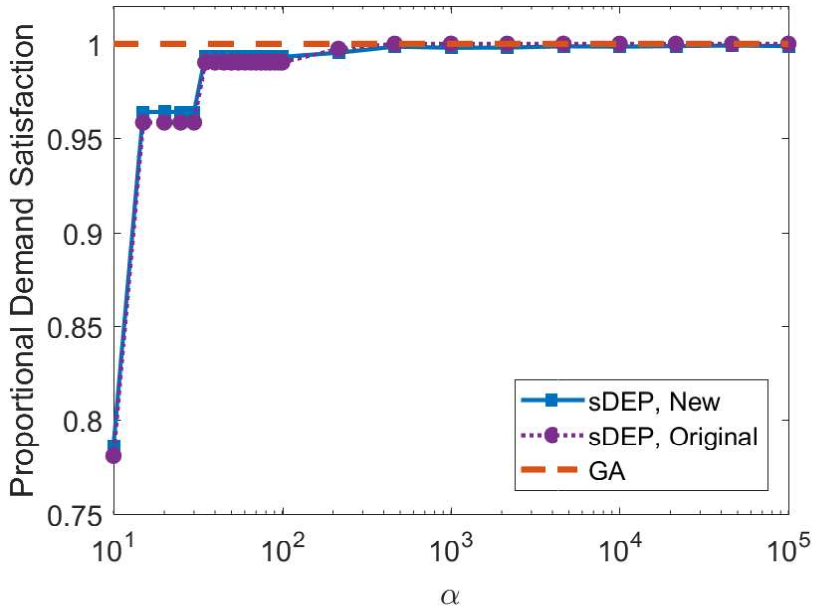


Figure 4.13: Proportional demand satisfaction of the sDEP and GA approaches sliced to 50 new scenarios using the adaptive slicing model. Shows the demand satisfaction for both the new scenarios not in $\hat{\Omega}$ (blue squares) and the original scenarios comprising $\hat{\Omega}$ (purple dotted circles).

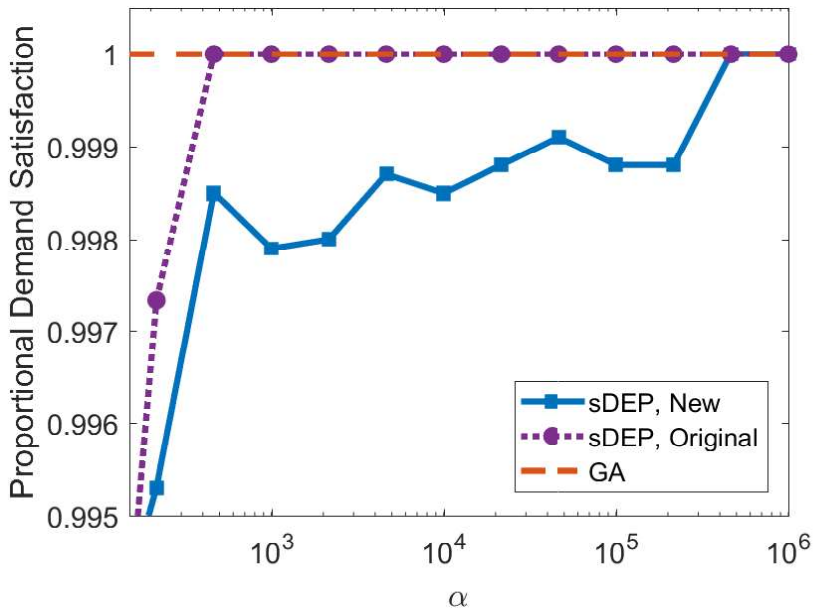


Figure 4.14: Proportional demand satisfaction of the sDEP and GA approaches sliced to 50 new scenarios. Finer detail of Figure 4.13 for the scope around 100% demand satisfaction.

$\hat{\Omega}$ no longer provide 100% demand satisfaction, but instead fluctuate over 99.8% to 99.9% demand satisfaction. This shows that the 12-BS solutions are not identical, despite being the same cost, and highlight the subtle differences between them. Further, the sDEP is unable to provide 100% demand satisfaction up to $\alpha < 4.6e5$, which includes solutions containing up to 18 BSs. Above this range of α, the sDEP is heavily overallocating resources; at $\alpha = 4.6e5$ the sDEP is selecting 43 BSs, jumping from 18 at the previous point of $\alpha = 2.2e5$. This implies that a key part of the original demand field is not demonstrated adequately by the scenarios of $\hat{\Omega}$, and that the sDEP needs to consider more demand points and include more scenarios in $\hat{\Omega}$ to more accurately represent the full sample space of λ_n.

The GA instead performs far better with the new scenarios due their improved distribution of demand. All but 2 GA solutions reach 100% demand satisfaction; the two exceptions instead reach 99.99%. This is tabulated in Table 4.4.

Table 4.4: Demand Satisfaction Results for Various GA-Constructed VWNs Over 50 Non-$\hat{\Omega}$ Scenarios

Number of BSs	**Number of Solutions**	**Proportional Demand Satisfaction**
12	1	100.0%
13	1	99.99%
13	24	100.00%
14	1	99.99%
14	16	100.0%
15	7	100.0%

Notably, the 12-BS solution, shown in Figure 4.15, reaches 100% demand satisfaction, which is higher than the same-cost VWNs constructed by the sDEP. With the number of scenarios and demand points tested here, the 12-BS GA solution provides a better resource

selection than the sDEP, and did so with a overall speedup factor of 36.4x; for the sDEP to improve resource selection would require additional scenarios and demand points, which would increase the approach's run time exponentially, as shown in Figure 4.3.

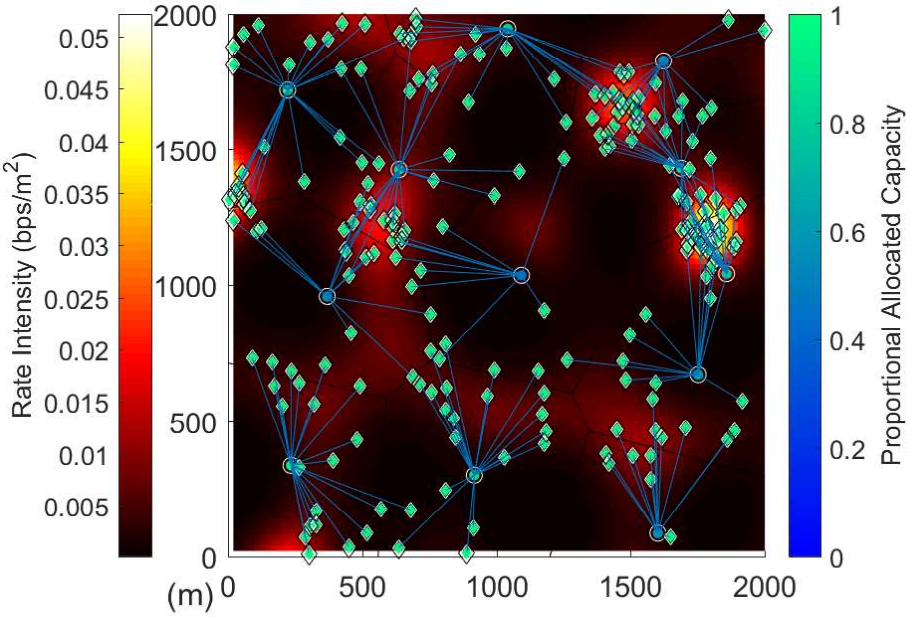


Figure 4.15: GA resource selection and slicing for one scenario *not* in $\hat{\Omega}$. Circles indicate BSs and diamonds indicate demand points. Lines connecting BSs to demand points indicate that the BS is sliced to the demand point (i.e., that a link between the BS and demand point exists). The color filling a BS indicates the proportion the BS's capacity is sliced, and the color filling a demand point indicates the proportion of the demand point's demands that have been satisfied.

The two following test cases (Sections 4.2 and 4.3) will investigate a modification to λ_n that allows for the GA to consider networks not aiming to provide 100% demand satisfaction. This will allow for better comparisons between constructed VWNs that are not aiming for 100% demand satisfaction.

Takeaway Conclusions

These preliminary simulations provide several optimistic findings. First, the GA approach, as expected, does provide a solution to the original two-stage stochastic optimization program in much less time than the sDEP approach, reaching up to a speedup factor of 46.3x. The run time of the GA is only dependent on the number of BSs being considered for selection and the number of pixels for integration within the fitness heuristic. In contrast the sDEP is dependent on the number of resources, the number of demand points, and the number of scenarios sampled forming $\hat{\Omega}$. The GA as designed generates a solution targeting 100% demand satisfaction—which might not be reached when considering scenarios with few, high-demand demand points—whereas the sDEP is tunable via α. For equal-cost VWNs, the GA approach introduces a small amount of error on the order of 0.3% demand satisfaction when considering scenarios the sDEP is optimized for. For scenarios not in $\hat{\Omega}$, the sDEP provides solutions that are less optimal whereas the GA's solutions accuracy is directly related to how accurately the scenarios reflect the initial demand intensity field, λ_n.

The following two sections will expand on the results from these preliminary simulations. They will attempt to improve on the simulations by tackling the limitations presented by these results.

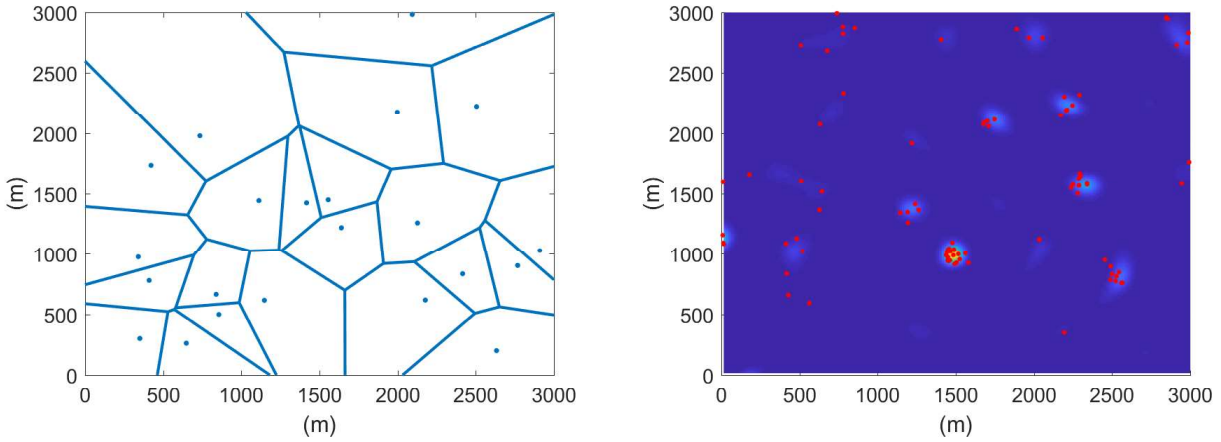
4.2 Case I: Single SP with Homogeneous Resources

In this section I present the first test case that builds on the work of the previous section comparing the sDEP and GA approaches. These simulations are designed to more accurately reflect real world resources and demand as suggested by empirical data. Further, these simulations present a modification to the demand intensity field, λ_n, to adjust the GA's target level of demand satisfaction. Specifically, this test case considers a situation where a VNB with access to purely homogeneous resources constructs a VWN for a single SP.

4.2.1 Case I Setup

This test case is constructed considering the following considerations and parameters. There is only one SP, and its demand intensity function, λ, is generated using parameters (i.e., $\omega_{\max}$, σ, μ) that were empirically found to match a real world urban cellular network in China [32]. λ is pixelated into a 200 by 200 grid of square pixels with 15 m side length; $\mathcal{R}$, illustrated in Figure 4.16, is a region 3 km by 3 km in size. Resources are homogeneous and constructed to represent small macrocells. The size of the region and the number of BSs, demand points, and scenarios were chosen such that the sDEP would converge to a solution in just under a new time limit of one hour wall clock time. Resources have a large coverage radius relative to the region's size; a larger region introduces more demand and requires more BSs for a solution to be feasible, increasing run times, and a smaller coverage radius would cause the resources to have sub-macrocell coverage area. This data set is *smaller* than

the set used in the preliminary simulations of Section 4.1. Due to the decreased number of demand points, each demand point contains a larger amount of the overall field's demand. This will be more properly handled with additional scenarios and the adaptive slicing model (Section 3.1.2). GA generation limits (i.e., G, $G_{\min}$, G_{fit}, and G_{gen}) were increased to avoid the possibly premature convergence seen in the last section. The parameter values used in the simulations are shown in Table 4.5.



(a) Voronoi tesselation of BS locations (b) λ with one realization of demand points ($\mathcal{M}$)

Figure 4.16: Illustration of resources and demand in the region used for Case I.

Modifications to the Demand Intensity Function for the GA Approach

In the previous section, the GA approach gave results that targeted 100% demand satisfaction for the SP. This locked the GA solutions such that they can only be directly compared to sDEP solutions which also achieve 100% demand satisfaction. Further, this forced a limitation of the GA approach that does not directly impact the sDEP approach.

To consider other demand satisfaction targets, the total demand of the region can be

Table 4.5: Numerical Values of Relevant Parameters for Case I

Width, Length of Geographic Area (X)	3 km x 3 km
Number of BSs (S)	22
Number of Demand Points (M)	90
Number of SPs (N)	1
Number of Sampled Scenarios (O)	30
sDEP Objective Weights (α)	$\{6, 8, \ldots, 30\}$ $\{40, 50, \ldots, 100\}$
BS Cost (c_s, $\forall s \in \mathcal{S}$)	1
BS Capacity (r_s, $\forall s \in \mathcal{S}$)	2 Gbps
BS Range (b_s, $\forall s \in \mathcal{S}$)	1500 m
Point Traffic Demand (d)	221.89 Mbps
SSLT Approximation Depth (L)	50
SSLT Maximum Angular Frequency ($\omega_{\max}$)	0.19010 (0.012673)
SSLT Location Parameter (σ)	18.93
SSLT Scale Parameter (μ)	2.3991
Maximum Number of Generations (G)	4000
Minimum Number of Generations ($G_{\min}$)	450
Unchanged Generations Before Fitness Halt (G_{fit})	300
Unchanged Generations Before Genome Halt (G_{gen})	600
Number of Individuals per Generation (I)	80
Number of Elite Individuals per Generation	4
Probability of Crossover (p_{xov})	0.7
Probability of Mutation per bit (p_{mut})	$1/s = 0.0455$
Overcoverage Cost (c_{cov})	3
Overcapacity Cost (c_{cap})	1.015
Pixel Grid Size	200 x 200, 15 m
GA Demand Weights (β)	$\{0.5, 0.6, 0.7, 0.8, 0.9,$ $1.0, 1.5, 2.0, 2.5\}$
Number of Additional Evaluation Scenarios ($\notin \hat{\Omega}$)	100
Number of Additional Scenario Demand Points (M)	300
Additional Demand Point Demand (d)	66.547 Mbps

scaled. Let the positive real number β be a weighting coefficient of the demand considered by the GA. Further, let the *effective* demand allocated to a selected resource $s \in \mathcal{S}' \subseteq \mathcal{S}$ for purposes of selection by the GA be $\sum_{n\in\mathcal{N}} \iint_{V_s} \lambda'_n(x,\, y)\, dxdy = \beta \sum_{n\in\mathcal{N}} \iint_{V_s} \lambda_n(x,\, y)\, dxdy$ where $\lambda'_n(x,\, y) = \beta \cdot \lambda_n(x,\, y)$. Resource s is now considered overcapacity if this effective demand allocation exceeds r_s.

This affects the fitness heuristic (eq. (3.17)) of the GA by changing the cost of a given individual, $\mathbf{z}^{\{ig\}}$, to be

$$\text{cost}\left(\mathbf{z}^{\{ig\}}\right) = \sum_{s\in\mathcal{S}} \Bigg(c_s\, z_s^{\{ig\}} + c_{\text{cov}}\, \mathbb{1}_{\{V_s \not\subseteq B_s\}} + \left(c_{\text{cap}}^{g} - 1\right)\, \max\left(0,\, \sum_{n\in\mathcal{N}} \left(\iint_{V_s} \lambda_n(x,\, y)\; dx\, dy \right) - \frac{r_s}{\beta} \right) \Bigg) \tag{4.1}$$

where $\mathbb{1}_{\{*\}}$ is the indicator function (eq. (2.19)). As indicated by the modified cost function, resource s is also considered overcapacity if the *actual* demand allocated to the BS exceeds r_s/β.

This only impacts the resource selection performed by the GA and only in the context shown. This affects neither the demand associated with individual demand points of a scenario nor the slicing of the selected resources to individual demand points in the adaptive slicing model (Section 3.1.2).

4.2.2 Case I Results

In this section I present the results of the Case I simulations. The results of the sDEP approach are presented, followed by the results of the GA approach, including the effects of introducing the weighting factor β. Finally, comparisons of the two approaches for these simulations are provided, and conclusions made.

Approach 1: The Sampled Deterministic Equivalent Program

Figure 4.17 visualizes the run time and cost of the sDEP approach, which behaves very similarly under this data set compared to the preliminary simulations. The run time averages at approximately 15000 seconds, spiking to nearly 25000 seconds for $\alpha \in \{10, 12\}$. For this region, the sDEP is in transition; the sDEP allocates 9 BSs for $\alpha = 10$ and 10 BSs for $\alpha \geq 12$.

This rapid transition from the trivial no-allocation solution ($\alpha < 10$) to 9 and 10 BSs indicates that the data set is highly polarized. Further simulations were conducted between $\alpha = 8$ and $\alpha = 10$ which demonstrates that the allocation transitions suddenly at approximately $\alpha = 9.88$ from 0 to 9 BSs. As Figure 4.18 shows, these allocations are the maximum necessary; at 9 BSs the sDEP reaches 90.14%, and at 10 BSs the sDEP achieves the maximum of 100% demand satisfaction. This remains true even for an additional set of 100 scenarios not in $\hat{\Omega}$.

The data set chosen is too forgiving on the available resources. Since resources have a

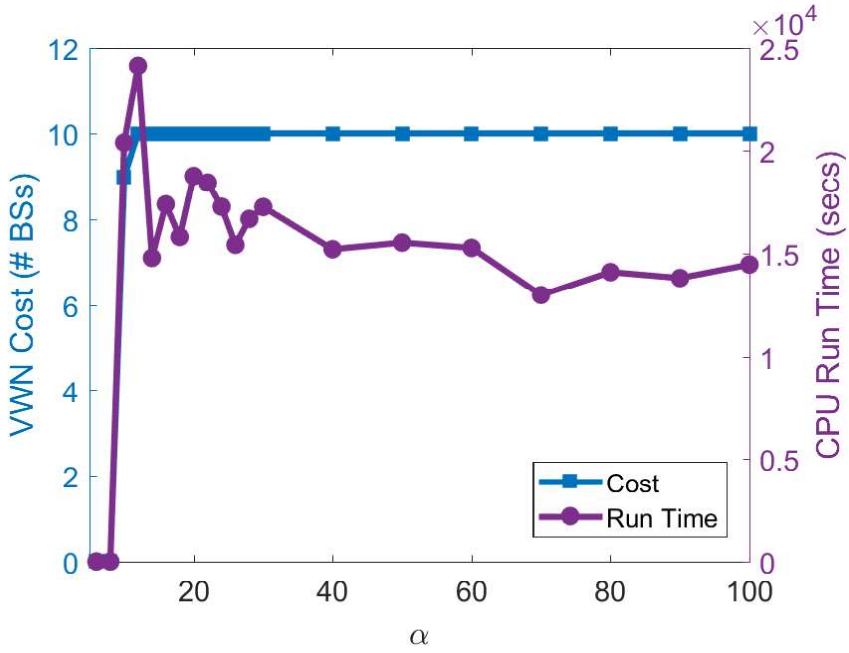


Figure 4.17: Comparison of sDEP CPU run time and VWN construction cost

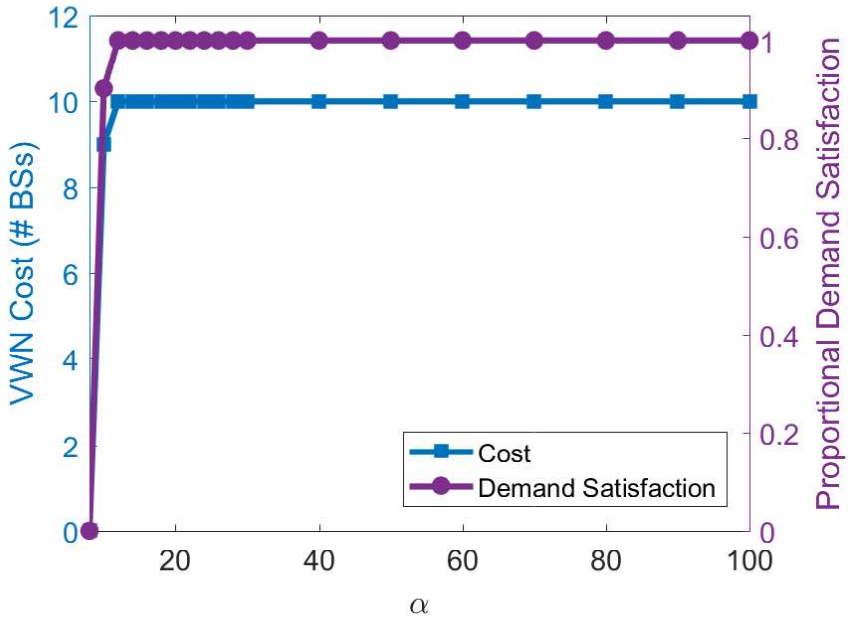


Figure 4.18: Comparison of sDEP VWN construction cost and associated proportional demand satisfaction. Both original scenarios (those in $\hat{\Omega}$) that the sDEP is optimized for and new scenarios (not in $\hat{\Omega}$) have identical demand satisfaction.

coverage radius of half of the region's side length—a single BS can span from one edge of the region to the other if centrally placed—most BSs can reach most demand points. This reduces the importance of coverage and simplifies slicing to solely a problem of capacity. The high time complexity of this specific set is likely due to the BSs' high proportional radii, as BSs have many possible demand points to be sliced to. This is shown in Figure 4.19, where resources are consistently linked with demand points over far distances relative to the size of the region. For instance, the largest demand spike of the region—located at approximately (1500, 1000)—is satisfied by slices from 7 separate BSs, 2 of which are in excess of a kilometer—or a third of the region's side length—away.

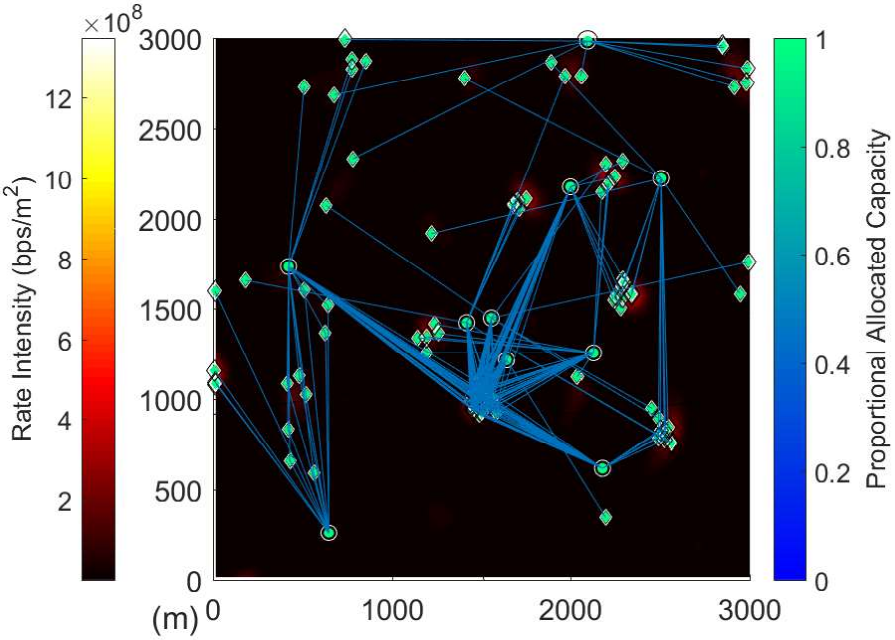


Figure 4.19: sDEP resource selection and slicing for one scenario in $\hat{\Omega}$. Circles indicate BSs and diamonds indicate demand points. Lines connecting BSs to demand points indicate that the BS is sliced to the demand point (i.e., that a link between the BS and demand point exists). The color filling a BS indicates the proportion the BS's capacity is sliced, and the color filling a demand point indicates the proportion of the demand point's demands that have been satisfied.

Approach 2: The Genetic Algorithm

The GA also performs similarly to what was shown in Section 4.1, with the primary differences of controlling β and increasing the number of generations before the halting condition occurs. This discussion will present these results with focus on the changes caused by controlling β similar to the focus spent on the sDEP with respect to α while presenting the preliminary simulation results.

The GA's run time (Figure 4.20) varies according to β with the solutions falling into three major categories. The first category of solutions ($\beta < 0.8$) is characterized by halting with the stronger and shorter fitness halting condition (i.e., the fitness of the fittest individual did not change for G_{fit} generations). The second category ($\beta > 0.8$), in contrast, is characterized by halting with the weaker genome halting condition (i.e., the genome of the fittest individual did not change for G_{gen} generations).

Figure 4.21 presents the run times of the various solutions in greater detail as histogram PDFs, and clearly shows the separation of these three categories. Except for $\beta = 0.5$, the GA is highly consistent in terms of CPU run time as a result of increasing the number of generations that must be satisfied for the halting conditions. The solutions of the first category each have an average of less than 300 seconds run time, the second at approximately 700 seconds and higher, and the third transitions between the two at just under 500 seconds. All solutions of the GA in the preliminary simulations terminated using the fitness halting condition, but did so almost universally below 200 seconds. This increase is due to the

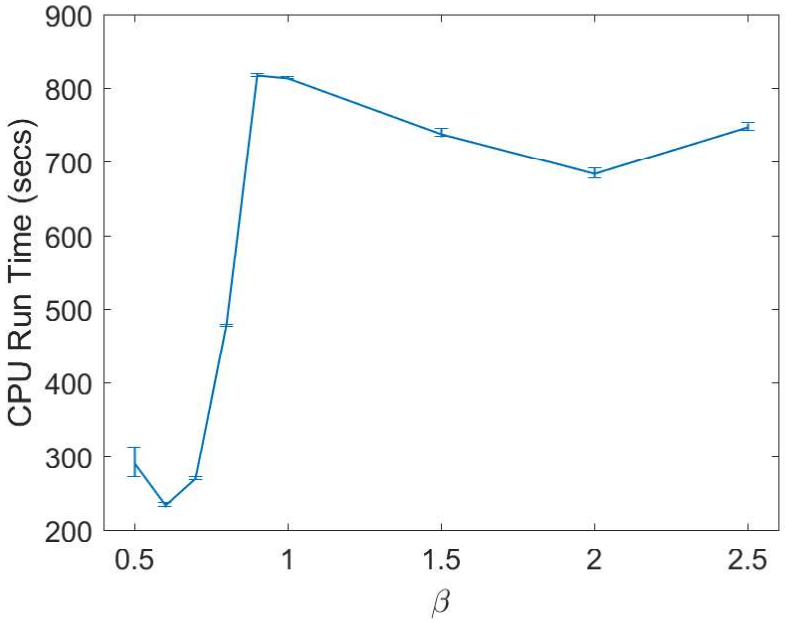


Figure 4.20: GA CPU run time trend as controlled by β with 95% confidence intervals

increased number of pixels in the field (10000 to 40000) and the increase to G_{fit} (150 to 300).

As β increases, the cost of the GA-selected network expectedly increases as a response to the increasing demand. Figure 4.22 presents the costs of the GA solutions in contrast to the run time taken to converge and halt. The trend with respect to β escalates fairly quickly and provides a variety of solutions with only two repeats. Like with run times, these solutions are highly consistent as a result of the longer-to-satisfy halting conditions. All but the $\beta = 0.5$ solutions have identical cost—though not identical selection—solutions across 50 executions of the GA. Even for the $\beta = 0.5$ solutions, the GA is very consistent, selecting 7 BSs in 42 of the GA's 50 executions (84%) and 8 BS for the remaining 8 (16%).

Figure 4.23 shows the level of demand satisfaction for the set of scenarios in $\hat{\Omega}$. The demand satisfaction for the additional evaluation set of scenarios not in $\hat{\Omega}$ is identical to the

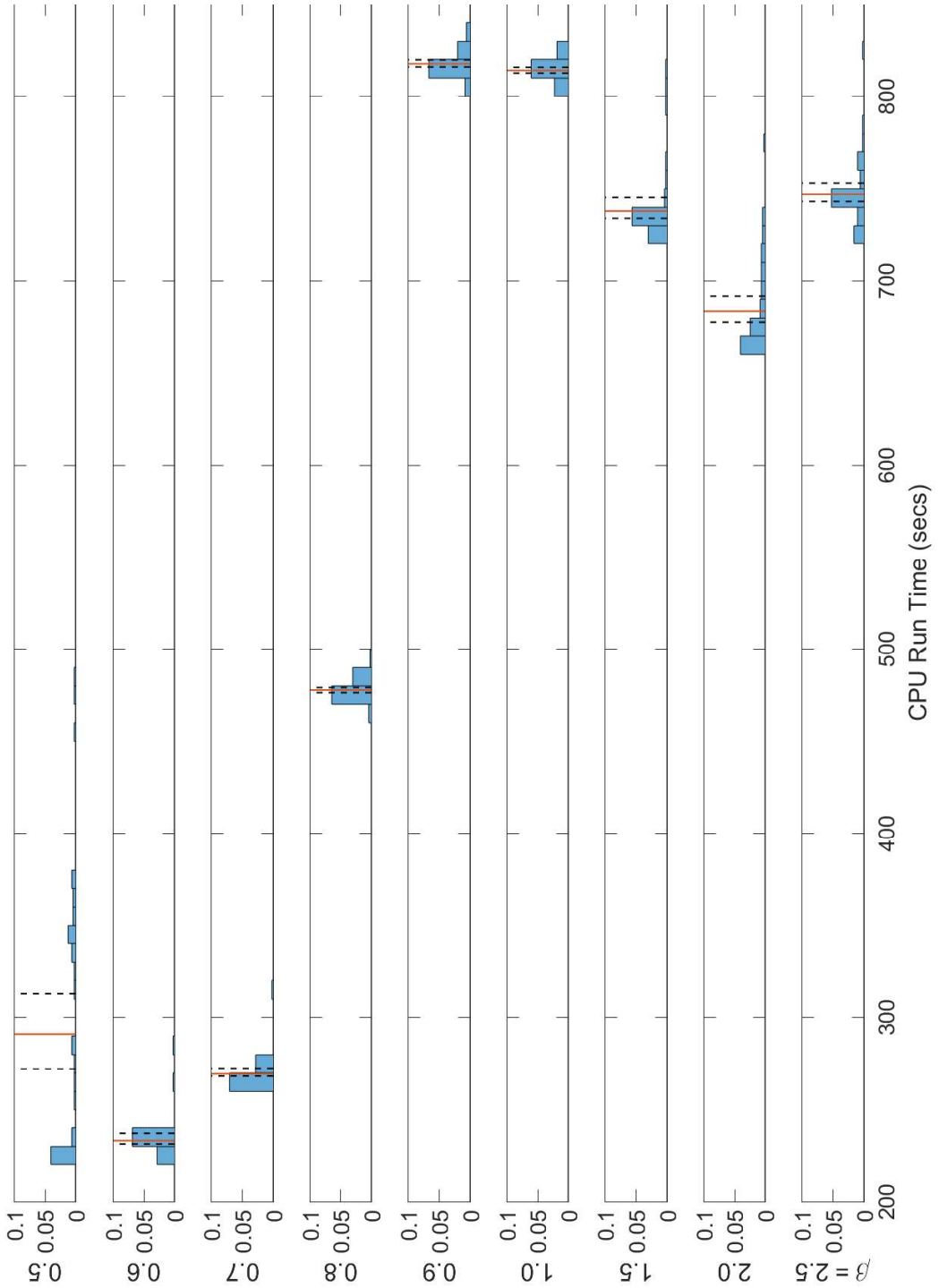


Figure 4.21: GA CPU run time trend in further depth than Figure 4.20 as histogram PDFs. For each β, means are shown in orange and 95% confidence intervals in dashed black.

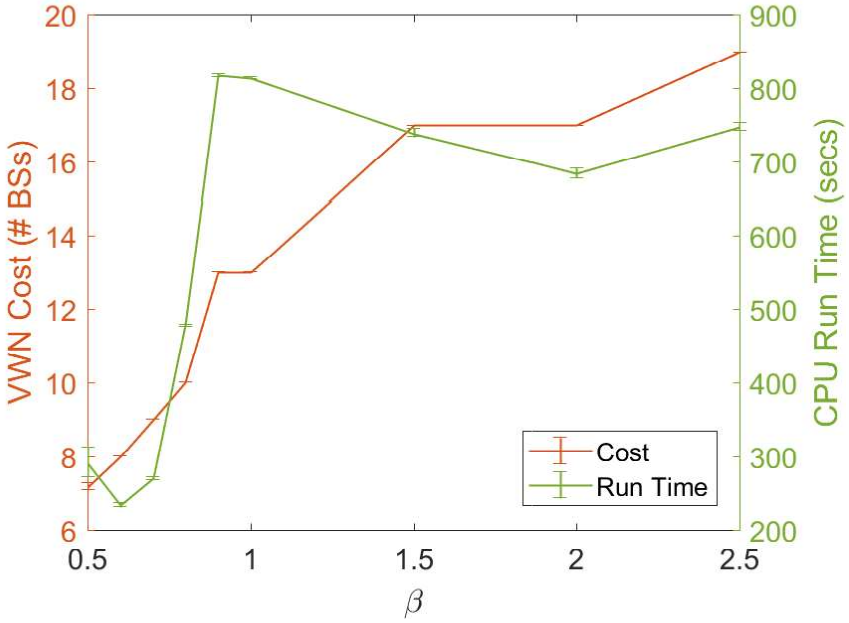


Figure 4.22: Comparison of GA cost and run time trends with 95% confidence intervals

demand satisfaction of $\hat{\Omega}$. Demand satisfaction is less than 100% for solutions selecting 7, 8, and 9 BSs ($\beta \in \{0.5, 0.6, 0.7\}$; first category solutions), with demand satisfaction increasing near linearly with 71.71%, 80.12%, and 90.14% demand satisfaction, respectively. For solutions selecting 10 or more BSs ($\beta \geq 0.8$; second category solutions), demand satisfaction universally reaches 100%.

The differences between the first and second categories of solutions is indicative of their satisfied halting condition. The first halted with the fitness halting condition, indicating that the GA was able to create a solution that was neither overcoverage nor overcapacity, and therefore could be expected to provide 100% demand satisfaction for the demand considered by the GA (i.e., $\beta \cdot \lambda$). Solutions that are neither overcoverage nor overcapacity should provide a minimum demand satisfaction equivalent to β. Figure 4.24 visualizes the demand

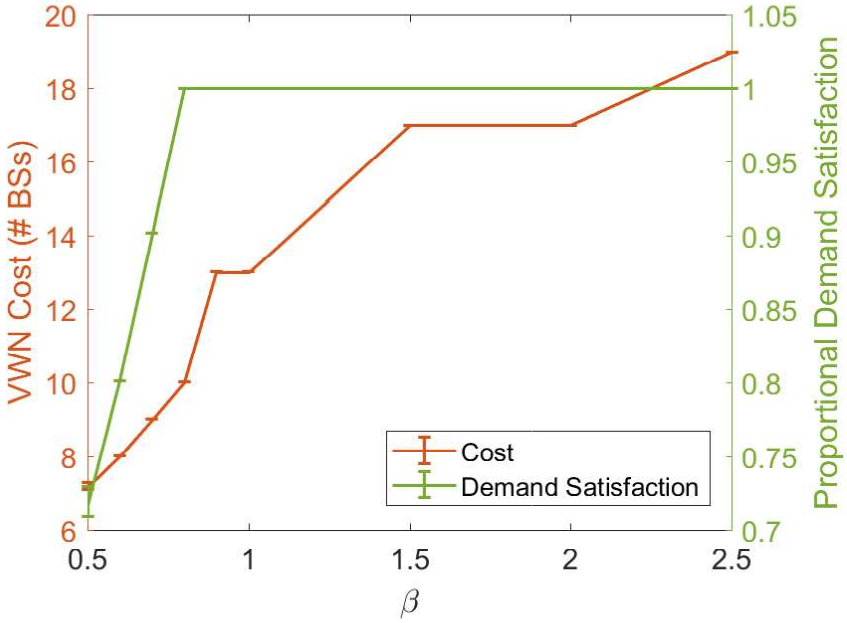


Figure 4.23: Comparison of GA cost and run time trends with 95% confidence intervals.

satisfaction of the various solutions relative to the demand actually considered by the GA (i.e., demand of each demand point is scaled by β before slicing is performed by the adaptive slicing model; point demands associated with $\beta \cdot \lambda$). For all of the non-overcapacity solutions, the GA allocated sufficient resources for 100% demand satisfaction of the demand scaled by β. This continues for the overcapacity solutions up to $\beta = 1.5$, above which there are insufficient resources to allocate to fully satisfy the highly inflated β-scaled demand field.

In contrast, the second and third categories halted with the genome halting condition, indicating that the GA was unable to find a solution that was neither overcoverage nor over-capacity for the demand considered. Solutions that are overcapacity are unable of targeting 100% demand satisfaction. However, with BSs sharing mutually covered points through the adaptive slicing model, it might still reach 100% for the demand seen by the GA. For this

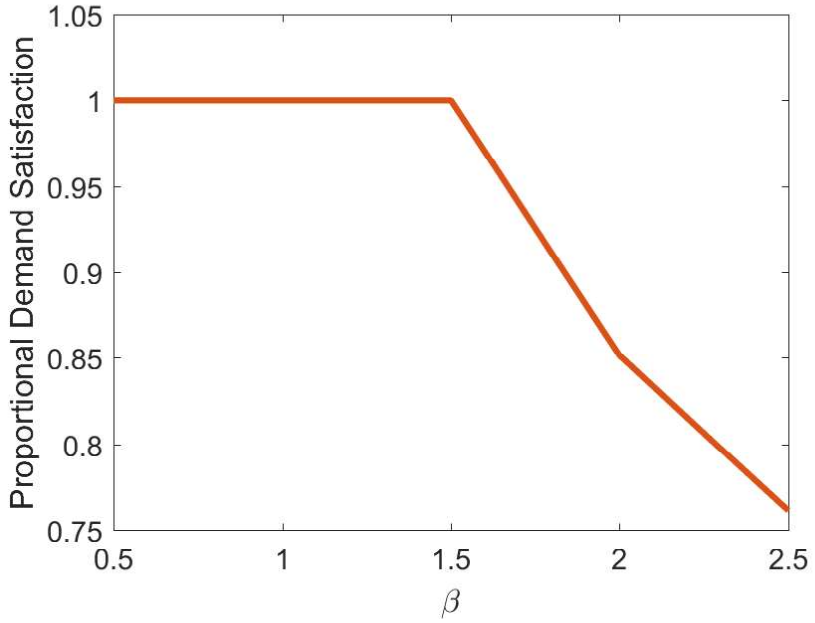


Figure 4.24: GA demand satisfaction with β-scaled demands

data set, all solutions for this category have 100% demand satisfaction, though all of the second category ($\beta > 0.8$) overallocated resources.

Comparisons of the Approaches

With the addition of the weighting factor β, the GA is capable of being tuned to provide a number of solutions, akin to the sDEP and its weighting coefficient α. Regardless of this tuning, the sDEP and GA both perform very similarly to what was shown by the preliminary simulations.

For all α and β, the GA has an improved run time over the sDEP. Figure 4.25 compares the run time trends of the two approaches in terms of the GA's speedup ratio. At its most optimal, the GA achieves a speedup ratio of up to 91.25x improvement over the sDEP.

Comparing equal cost solutions, the GA provides a speedup ratio of 77.08x for 9-BS solutions ($\alpha = 10$; $\beta = 0.7$, quick) and 31.12x for 10-BS solutions ($\alpha > 12$; $\beta = 0.8$, medium).

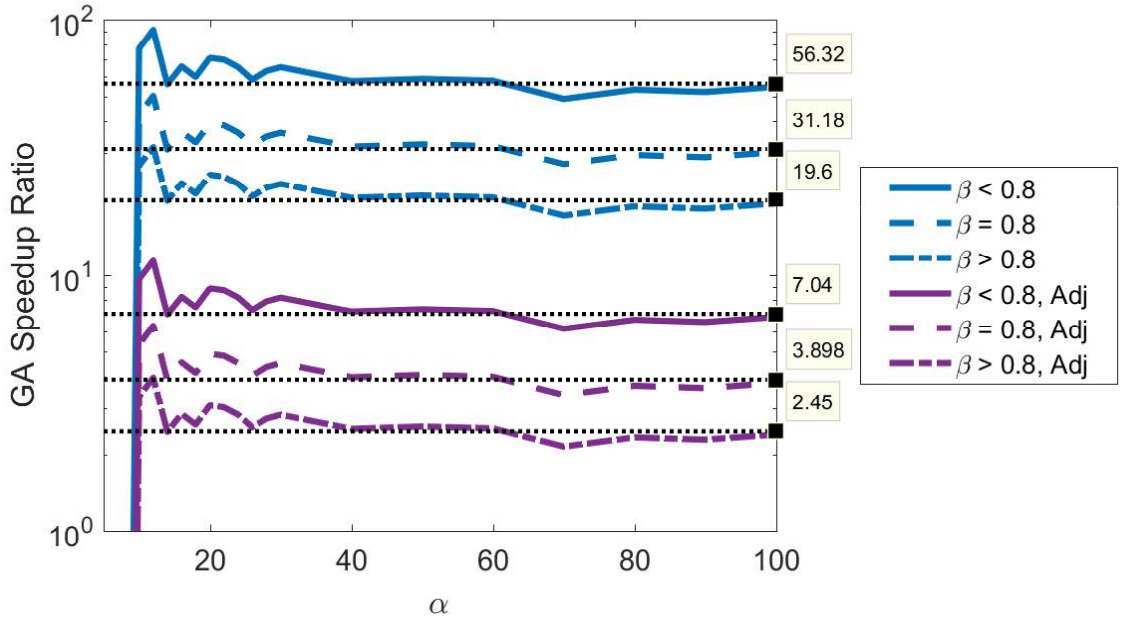


Figure 4.25: GA speedup ratios of different category solutions. Dotted black lines are the speedup ratio relative to the mean of the GA solution run times in that category. Adjusted speedup ratios are considering the adjusted run time of the sDEP, which is a lower bound for the wall clock time.

Both approaches are capable of providing similar cost networks. All of the GA solutions that select less than 10 BSs—the cheapest 100% demand satisfaction solution, as shown by the sDEP—converged with the fitness halting condition, whereas all of the genome halted solutions provide either the minimum-cost 100% demand satisfaction solution ($\beta = 0.8$) or more expensive (the second category, $\beta > 0.8$) solutions which overallocate. However, this is likely coincidental, as the line between the first and second category of solutions is tied to the parameters of the resources and demand, and the results of the preliminary simulations had solutions from the fitness halting condition that provided 100% demand satisfaction.

Further, as tabulated by Table 4.6, same cost networks of both approaches provide equal demand satisfaction. However, this lack of error between the sDEP and GA solutions is likely due to the high relative radii of the resources as described in the sDEP results. While coverage still matters, BSs have coverage that is near enough to universal that satisfaction is solely a question of how many BSs have been selected. Each BS provides approximately 10% demand satisfaction, which is approximately the portion of the demand of the overall field that can be satisfied by the BSs' rate capacities.

Table 4.6: Cost and Demand Satisfaction Results of Various sDEP- and GA-Constructed VWNs

Cost	**Demand Satisfaction**	**sDEP α**	**GA β (Number of Solutions)**
0	0.000%	≤ 8	n/a
7	71.71%	n/a	0.5 (42)
8	80.12%	n/a	0.5 (8) 0.6 (50)
9	90.14%	10	0.7 (50)
10	100%	≥ 12	0.8 (50)
13	100%	n/a	0.9 (50) 1.0 (50)
17	100%	n/a	1.5 (50) 2.0 (50)
19	100%	n/a	2.5 (50)

Case I Conclusions

The GA continues to show improvements over the sDEP in terms of run time due to not needing to scale according to the number of demand points or scenarios of $\hat{\Omega}$. With the inclusion of β, the GA is now capable of providing solutions that aim for less than 100% demand satisfaction; it is expected solutions that converge with the fitness halting condition

will provide at least β demand satisfaction, assuming that λ fairly approximates the demand the constructed VWN is expected to meet. Run times for the GA are consistent within the halting condition that terminates it, whereas the sDEP will spike in run time on the threshold between higher cost solutions.

Regarding demand satisfaction, the results of this test case are too consistent. Ignoring coverage, each BS is capable of providing capacity (2 Gbps) for approximately 10% of the demand within the field (19.97 Gbps). Due to the wide coverage radii of the resources within this test case, the problem was almost completely reduced to the point of ignoring coverage; each BS selected effectively provided 10% demand satisfaction with 100% being reached with 10 BSs.

The next test case counters this problem without reducing macrocells beneath realistic coverage radii, increasing the size of the demand field and the rate capacity of BSs to account for the new demand, or increasing the size of the data set beyond the point of computational feasibility for the sDEP.

4.3 Case II: Single SP with Heterogeneous Resources

In this section I present the second test case that builds on the work of the preliminary simulations comparing the sDEP and GA approaches. These simulations use primarily the same parameters used by the first test case, but expand the set of resources aggregated by the VNB. Instead of a single RP providing a set of homogeneous macrocell BSs, an additional

RP is available for the VNB providing a set of microcell BSs. These microcells have a lower coverage radius and rate capacity, but increased cost efficiency (i.e., rate capacity per cost). Thus, the VNB considers a heterogeneous set of resources to satisfy the SP's demands in this test case.

4.3.1 Case II Setup

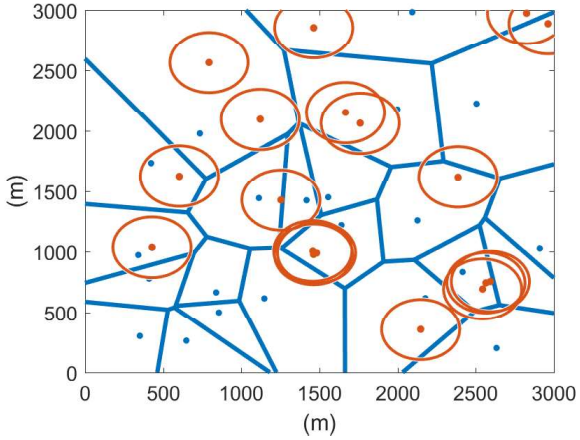
This test case is constructed with the same considerations and parameters as Case I (Section 4.2), with the following exceptions. Resources are heterogeneous, with most constructed to represent small macrocells and the rest constructed to represent microcells. Macrocells are placed according to a stationary PPP. Microcells are placed according to a non-stationary PPP using λ as the intensity function, representing BSs placed to satisfy specific demand needs. The size of the region and the number of BSs, demand points, and scenarios were chosen such that the sDEP would converge to a solution in just under a time limit of one hour wall clock time. The parameter values used are shown in Table 4.7, and the resources and demand in $\mathcal{R}$ is shown in Figure 4.26.

4.3.2 Case II Results

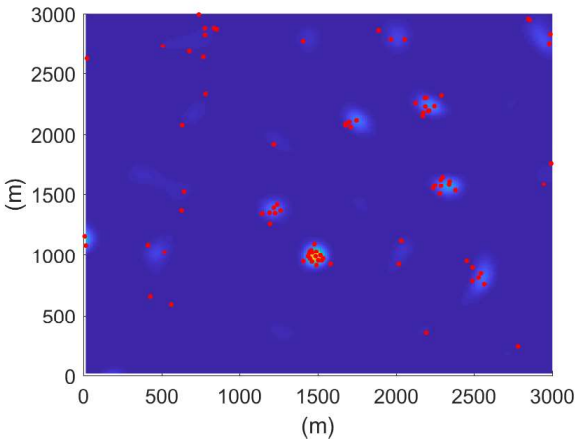
In this section I present the results of the Case II simulations. The results of the sDEP approach are presented with consideration of the changes caused by the use of heterogeneous resources. The results of the GA approach are similarly presented. Finally, comparisons for

Table 4.7: Numerical Values of Relevant Parameters for Case II

Width, Length of Geographic Area (X)	3 km x 3 km	
Number of BSs (S)	40 (Total)	22 (Macrocells) 18 (Microcells)
Number of Demand Points (M)	90	
Number of SPs (N)	1	
Number of Sampled Scenarios (O)	30	
sDEP Objective Weights (α)	$\{6, 8, \ldots, 30\}$ $\{40, 50, \ldots, 100\}$	
BS Cost (c_s)	1 (Macrocells) 0.04 (Microcells)	
BS Capacity (r_s)	2 Gbps (Macrocells) 100 Mbps (Microcells)	
BS Range (b_s)	1500 m (Macrocells) 250 m (Microcells)	
Point Traffic Demand (d)	221.89 Mbps	
SSLT Approximation Depth (L)	50	
SSLT Maximum Angular Frequency ($\omega_{\max}$)	0.19010 (0.012673)	
SSLT Location Parameter (σ)	18.93	
SSLT Scale Parameter (μ)	2.3991	
Maximum Number of Generations (G)	4000	
Minimum Number of Generations ($G_{\min}$)	450	
Unchanged Generations Before Fitness Halt (G_{fit})	300	
Unchanged Generations Before Genome Halt (G_{gen})	600	
Number of Individuals per Generation (I)	80	
Number of Elite Individuals per Generation	4	
Probability of Crossover (p_{xov})	0.7	
Probability of Mutation per bit (p_{mut})	$1/s = 0.0455$	
Overcoverage Cost (c_{cov})	3	
Overcapacity Cost (c_{cap})	1.015	
Pixel Grid Size	200 x 200, 15 m	
GA Demand Weights (β)	$\{0.5, 0.6, 0.7, 0.8, 0.9,$ $1.0, 1.5, 2.0, 2.5\}$	
Number of Additional Evaluation Scenarios ($\notin \hat{\Omega}$)	100	
Number of Additional Scenario Demand Points (M)	300	
Additional Demand Point Demand (d)	66.547 Mbps	



(a) Microcell locations (orange) and Voronoi tesselation of macrocell locations (blue)



(b) λ with one realization of demand points ($\mathcal{M}$)

Figure 4.26: Illustration of resources and demand in the region used for Case II

these approaches are provided, and conclusions made.

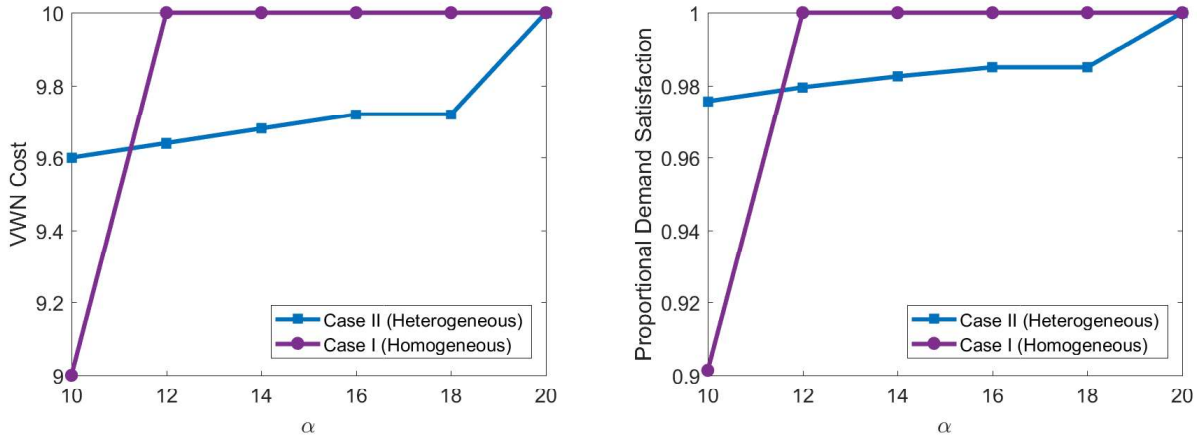
Approach 1: The Sampled Deterministic Equivalent Program

Due to the increased number of available resources, the sDEP approach took longer to solve for all values of α. For $\alpha \geq 12$ the sDEP approach exceeded the time limit of 1 hour wall clock time[2]. As a result, these solutions are not guaranteed to be optimal, only the most optimal solution found by CPLEX. $\alpha = 6$ presents a trivial, 0-BS selection solution, and $\alpha \in \{8, 10\}$ are the only sDEP solutions that provided optimal non-trivial solutions within the 1 hour time limit; $\alpha = 8$ completed in less than 1 second CPU time, and $\alpha = 10$ completed in 16,000 seconds CPU time. As a comparison, the sDEP Case I results averaged

[2]Ideally, there are 28,800 seconds CPU time (8 cores) in 1 hour wall clock time. These solutions terminated at 28,200 seconds CPU time, reaching 97.95% of the ideal and a parallelism speedup ratio of 7.8x. This is an improvement over the 7.1x speedup of the preliminary simulations (Section 4.1), and are closer to the limit presented by the 1/s adjusted run time in figures.

15,000 seconds outside of the transition between the 9- and 10-BS solutions.

With the inclusion of the microcells, the VNB is capable of more nuanced constructions. Figure 4.27a shows the cost trend of the sDEP approach for the range of $\alpha \in [10, 20]$; $\alpha = 8$ constructs a network of 0.44 cost, and $\alpha \geq 20$ constructs a network with 10 cost. At $\alpha = 10$, the sDEP constructs a network with 9.6 cost (9 macrocells, 15 microcells) (Figure 4.28a). This cost steadily increases with additional microcells up to $\alpha = 16$ with 9.72 cost (9 macrocells, 18 microcells) (Figure 4.28b).

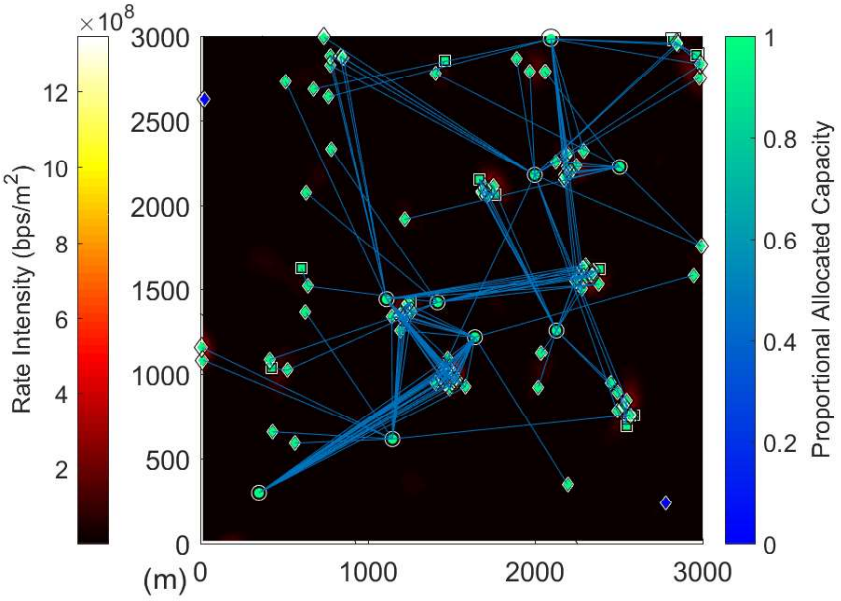


(a) Costs (b) Demand Satisfaction

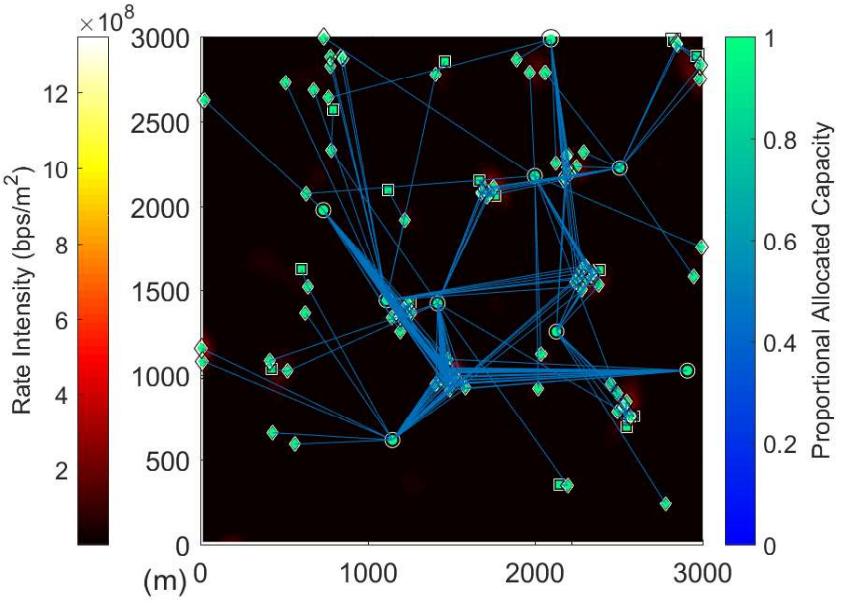
Figure 4.27: Comparison of sDEP solutions between Case I and Case II. The non-trivial $\alpha = 8$ Case II solution is not shown, but selected a network with 0.44 cost and 5.51% demand satisfaction. $\alpha = 8$ for Case I is a trivial 0-BS, 0% demand satisfaction solution.

Each additional microcell increases the overall demand satisfaction of the constructed network, but fails to fully satisfy without an additional macrocell. To achieve 100% demand satisfaction, the network must satisfy 19.97 Gbps[3] of demand, which requires 10 macrocells or 9 macrocells and 20 microcells to satisfy; since there are only 18 microcells available,

[3]One scenario includes 90 demand points with each demanding 221.89 Mbps.



(a) 9 Macrocells, 15 Microcells (97.57% Demand Satisfaction; $\alpha = 10$)



(b) 9 Macrocells, 18 Microcells (98.50% Demand Satisfaction; $\alpha = 16$)

Figure 4.28: sDEP resource selection and slicing for one scenario in $\hat{\Omega}$. Circles indicate macrocells, squares indicate microcells, and diamonds indicate demand points. Lines connecting BSs to demand points indicate that the BS is sliced to the demand point (i.e., that a link between the BS and demand point exists). The color filling a BS indicates the proportion the BS's capacity is sliced, and the color filling a demand point indicates the proportion of the demand point's demands that have been satisfied.

there are insufficient microcells to bridge the gap between 9 and 10 macrocells. This is further reflected in Figure 4.27b; for $\alpha \in [10, 18]$, the sDEP has approximately 98% demand satisfaction, and reaches 100% for $\alpha \geq 20$.

Testing the solutions for new scenarios not in $\hat{\Omega}$ shows that increasing the number of demand points continues to better distribute demand and increased demand satisfaction. Figure 4.29 shows the demand satisfaction trend for the same range as Figure 4.27 for both Case I and Case II with respect to the scenarios in $\hat{\Omega}$ and new scenarios not in $\hat{\Omega}$. While there was no room for changes in Case I, the imperfect satisfaction results that are available due to inhomogeneity show improvement with more distributed demand.

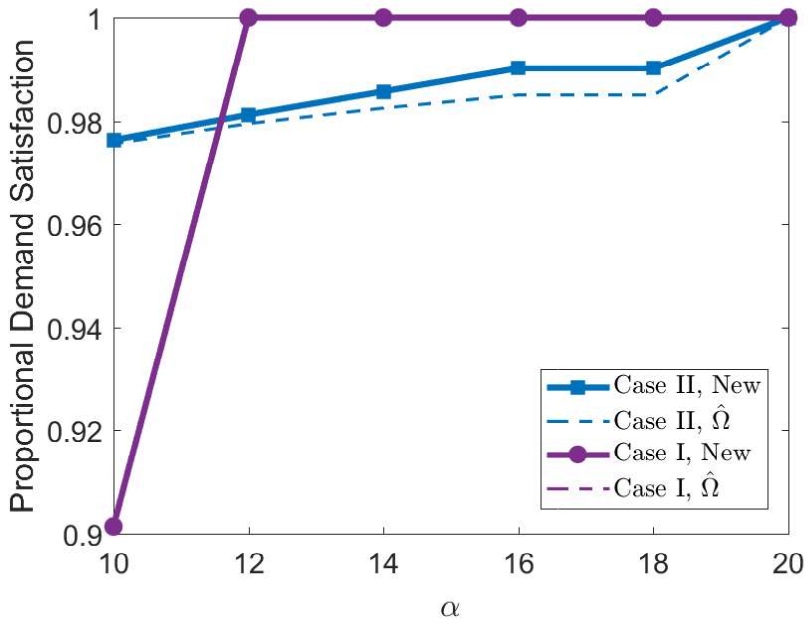


Figure 4.29: Comparison of demand satisfaction of scenarios both in and not in $\hat{\Omega}$

Approach 2: The Genetic Algorithm

The GA behaves very similarly for Case II as it did in Case I. As Figure 4.30 shows, Case II converges with low run times for $\beta < 0.8$ coordinating to the fitness halting condition, higher run times for $\beta > 0.8$ coordinating to the genome halting condition, and $\beta = 0.8$ as a transition. However, the GA for Case II took 25% to 50% additional run time to converge, reaching approximately 1050 seconds CPU time for $\beta > 0.8$. This coordinates with the 18 additional resources the GA must consider for Case II compared to Case I.

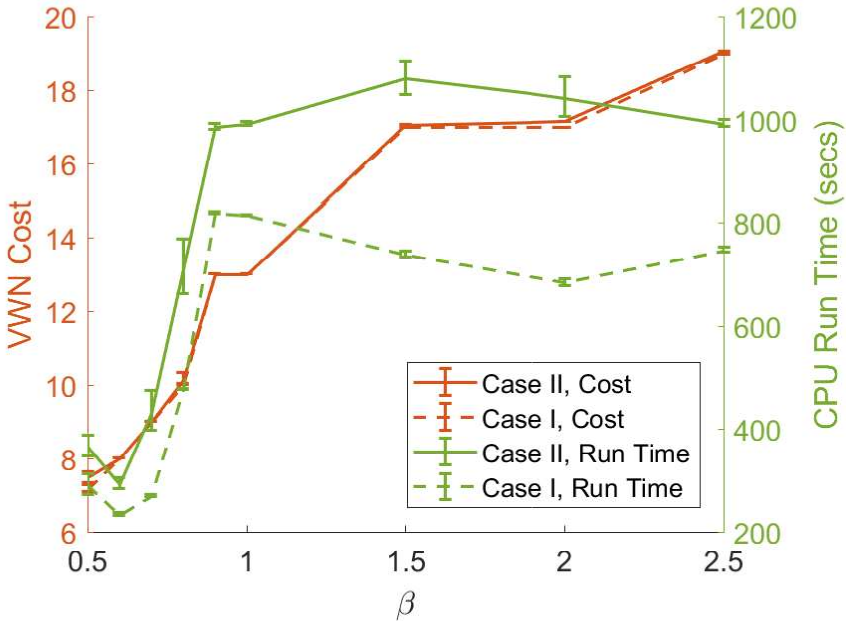


Figure 4.30: Comparison of GA cost and run time between Case I and Case II

Despite the increased computation time, the GA solutions barely changed in terms of cost. Case II is fairly consistent, though notably less consistent than Case I. The results are tabulated in Table 4.8, along with the associated demand satisfactions for those solutions; all equal cost GA solutions had equal demand satisfaction. Despite the additional microcell

resources to utilize, the GA largely ignores those resources, opting to use macrocells for $\beta < 0.8$ with only 6 exceptions. As with Case I, all VWNs of cost greater than or equal to 10 provided 100% demand satisfaction. The only cost outside of that range that was not found in Case I is 9.04, as provided from three runs of the GA with $\beta = 0.7$; these three results represent an outlying, less-than-optimal result compared to the other $\beta = 0.7$ solutions. Further, $\beta = 0.5$ deviated far more often than in Case I, converging to a 7-BS solution only 52% of the time, compared to 84% in Case I.

Table 4.8: Cost and Demand Satisfaction Results of Various GA-Constructed VWNs

Cost	**Demand Satisfaction**	**GA β (Number of Solutions)**
7	70.11%	0.5 (26)
8	80.12%	0.5 (24) 0.6 (50)
9	90.14%	0.7 (47)
9.04	90.52%	0.7 (3)
10	100%	0.8 (47)
12.08	100%	0.8 (3)
13	100%	0.9 (50) 1.0 (50)
17.08	100%	1.5 (50)
17.16	100%	2.0 (50)
19.08	100%	2.5 (50)

As the solutions for Case II trended towards the more expensive, the GA demand satisfaction was higher for the same value of β. This is primarily noticeable for $\beta = 0.5$, which had a higher mean cost over Case I; other deviations occur in the range of $\beta \geq 0.8$, which has universally shown more than 100% demand satisfaction. These trends can be seen in Figure 4.31.

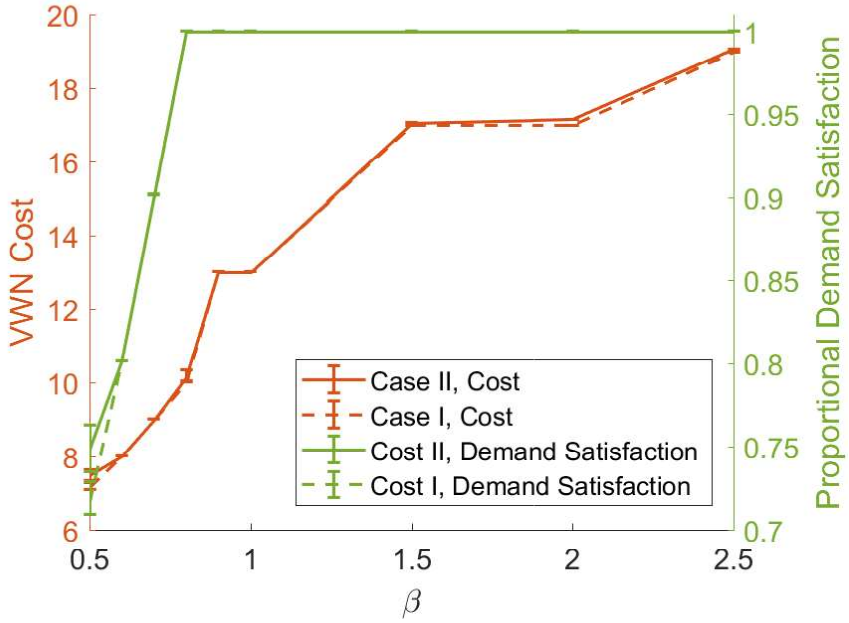


Figure 4.31: Comparison of GA cost and demand satisfaction between Case I and Case II

Comparisons and Conclusions

Both the sDEP and GA approaches appear to be largely unimpacted by the addition of heterogeneity to the VNB's resource pool, at least for this test case. Both approaches resulted in increased run times, which is expected with the increased number of resources; the GA's run time increased by approximately 25% to 50% while the sDEP failed to resolve within the allotted time limit of 1 hour wall clock time. Cost was impacted, but largely in the context of additional granularity. The sDEP was impacted more, as the addition of microcells allowed the sDEP to provide nuanced differentiation between 9 and 10 macrocell allocation. In contrast, the GA was impacted by making the results less consistent; Case I resulted in only 8 deviations (all $\beta = 0.5$) from the best solution across 450 different runs of the GA whereas Case II resulted in 30 (24 for $\beta = 0.5$, 3 for $\beta = 0.7$, 3 for $\beta = 0.8$).

That is, the addition of microcells resulted in 200% more error for $\beta = 0.5$ than the test without. This could be the result of changes in the data set that are not reflected solely by the inclusion of microcells, however.

It is expected to see the GA perform less efficiently than the sDEP as a result of its heuristic and approximations. However, results have shown thus far that the GA performs very similarly to the sDEP. Figure 4.32 shows the cost efficiency of the sDEP and GA approaches for both Case I and Case II. When only considering the homogeneous resources of Case I, the GA performs exactly as cost effectively as the sDEP, though the GA was able to force sub-90% demand satisfaction networks while the sDEP could not. With the addition of microcells in Case II, the sDEP performed slightly better through nuanced steps; this is shown as the slightest improvement in Figure 4.32 over the effectively identical sDEP Case I, GA Case I, and GA Case II results. For this data set, the GA has a visible, but insignificant amount of error compared to the sDEP. At the cost of this insignificant error, the GA provides a solution with a notable speedup ratio, shown in Figure 4.25 to reach in excess of 50x.

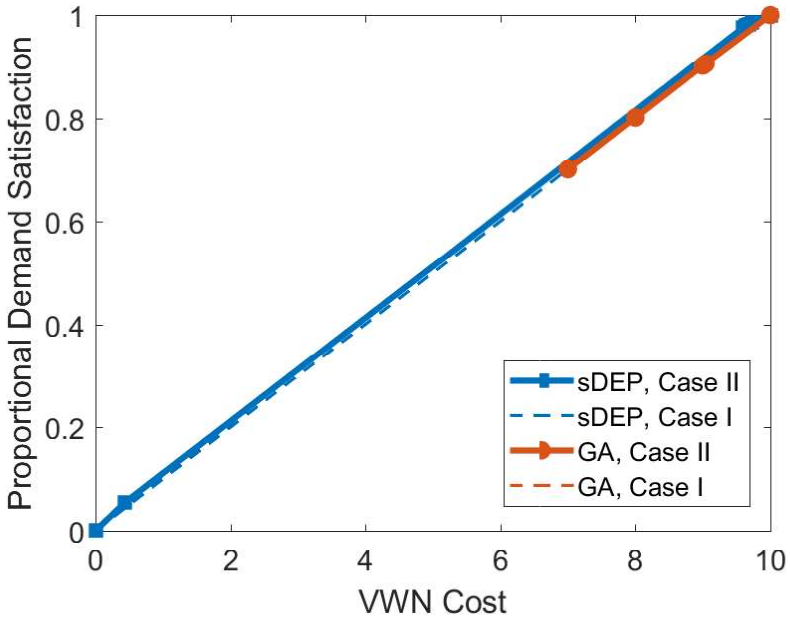


Figure 4.32: Cost effectiveness of sDEP and GA approaches demonstrated by Case I and Case II

Chapter 5

Summary and Conclusions

In this thesis I considered the problem of BS selection and dynamic slicing in the context of VWNs. In Chapter 1, this thesis presented virtualization as a necessity of the expanding requirements of cellular networks. It described a service-oriented virtualization architecture as an extension of the Networks without Borders paradigm, where RPs provide resources to VNBs which aggregate slices of resources to construct virtual networks optimized for the services offered by SPs. It established the problem of resource allocation as a primary task of the VNB.

Chapter 2 defined a system model providing context for the VNB. This model established that RPs provide resources as cellular BSs aggregated by the VNB. It further established the needs of SPs as a spatial demand intensity function, and provided an example using the SSLT demand model, which was designed to mimic the empirically discovered demands of an

urban cellular network. The problem the VNB considers was then presented as a two-stage stochastic optimization program, in which the first stage selected resources from the pool made available by the RPs optimizing for minimum cost, and the second stage sliced the resources selected by the first stage to the SPs optimizing for maximum demand satisfaction.

Chapter 3 provided two approaches for solving the VNB's optimization program. The first approach was a direct solution, converting the stochastic program into its deterministic equivalent program and sampling it into a tractable form. The second approach applied a genetic algorithm as a heuristic for determining resource selection. To provide a slicing of the selected resources, a simplified form of the deterministic equivalent program that solves only the second stage is applied to the solution of the genetic algorithm.

Chapter 4 considered the effectiveness of the two approaches via three separate simulations. The first simulation was a proof of concept and showed that the two approaches converge to a solution and that the SSLT demand model worked as expected. The latter two simulations served as test cases which modify RP resources and SP demand to more closely reflect real world resources and demand. The second simulation, Case I, simulated homogeneous (i.e., macrocell) resources and the third simulation, Case II, simulated heterogeneous (i.e., macrocell and microcell) resources. Except for highlighting the impact of resource coverage range on the problem, the differences between the various simulations do not cause significant changes in the approaches' effectiveness.

The GA approach has an improved run time compared to the sDEP. Across all simulations, the GA approach converged to a solution in far less CPU time than the sDEP

approach; as shown in Table 5.1, the GA exhibits a consistent and significant speedup ratio relative to the sDEP, even when comparing the adjusted[1] sDEP run time. The sDEP approach's run time is directly dependent on the number of possible connections between demand points and BSs across all scenarios; the sDEP is capable of ignoring demand points that are out of range of resources (i.e, where $u_{ms} = 0$). Generally, as the resources, demand points, and scenarios increase, run time increases exponentially; this is also impacted by BS coverage radius, as shown by the relatively low impact of the microcells added to Case II over Case I. The GA approach's run time is only directly dependent on the number of resources, the number of pixels of the demand intensity function, the number of generations before the GA halts, and the number of individuals per generation of the GA. Compared to the sDEP approach, the GA approach trends to be more time-feasible, as the number of generations[2], the number of pixels[3], and the number of resources[4] linearly impact run time. The number of individuals[5] impact run time with $O(n \log n)$ complexity.

Table 5.1: Average Simulation Speedup Ratios

Simulation	**Solution Category**	**CPU**	**Adjusted**
Preliminary	Transition ($\alpha \in \{25, 30, 35, 40, 45\}$)	40.9	5.11
	Steady ($\alpha \geq 50$)	10.3	1.29
Case I	GA Fitness Halt ($\beta < 0.8$)	56.3	7.04
	GA Genome Halt ($\beta > 0.8$)	19.6	2.45
Case II	GA Fitness Halt ($\beta < 0.8$)	77.5	9.69
	GA Genome Halt ($\beta > 0.8$)	27.7	3.46

[1]The adjusted time is the minimum limit of the wall clock time the sDEP approach could have taken to complete. The adjusted time is one-eighth the CPU time; in practice the actual wall clock time is greater than the adjusted time, but was shown in Chapter 4 to be less than one-seventh the CPU time and approaches one-eighth as the sDEP run time increases.

[2]Each generation is independent and happens sequentially.

[3]Each additional pixel can cause one additional addition in the integral of each fitness calculation.

[4]Each additional resource linearly increases the summation of the fitness calculation.

[5]For elitism, individuals are sorted according to fitness each generation.

GA-constructed VWNs have marginal to no demand satisfaction error compared to sDEP-constructed VWNs of the same cost. For the preliminary simulation, it was shown that the GA approach incurred a 0.3% demand satisfaction error compared to the sDEP approach when satisfying demand the sDEP was optimized for. However, when comparing against new scenarios with more distributed demand, the GA solution reached 100% demand satisfaction while the sDEP only reached 99.8%–99.9%. The GA approach marginally outperforming the sDEP approach for these scenarios highlights that the sDEP approach is capable of not providing an optimal solution if the set of demand scenarios is not representative of the original demand field; this is the aforementioned trade-off between optimality of the solution and the number of demand points and the number of scenarios. For Case I and Case II, the GA achieved 0.00% demand satisfaction error compared to the sDEP. This is due to increasing the number of generations of the halting conditions and the increased BS coverage radius size relative to the region the network was constructed for. For each same-cost solution the approaches provided, both approaches provided the same demand satisfaction. The addition of heterogeneous resources in Case II allowed for more quantization in the cost and demand satisfaction of the two approaches, but only allowed for the sDEP to marginally outperform the GA approach in terms of demand satisfaction.

Both the sDEP and GA approaches are capable of solving the problem of resource selection and adaptive slicing within the VNB. However, while the sDEP approach is theoretically more accurate than the GA approach, simulations show that GA approach introduces marginal to no additional error in terms of the constructed VWN's ability to satisfy SP

demand. Further, the GA approach consistently converges to these nearly-optimal solutions with significantly less run time than the sDEP approach. The GA approach signifies a significant speedup within the VNB over the sDEP approach without hindering the VNB's ability to construct VWN's that capably satisfy the SP's demand.

5.1 Considerations for Future Work

This thesis considered the problem of resource selection and adaptive slicing from the perspective of the VNB in a virtualized wireless networking context. To tackle this problem, it explored the problem in a very specific context and while making several assumptions in the system model. It is apparent that there is further work to fully investigate just this problem of resource selection within the VNB in the context of the presented architecture. Apparent avenues of investigation and further work follow.

Further Simulations; Additional Cases

Chapter 4 includes preliminary simulations from a paper submitted to a conference and two specific simulation cases (Case I analyzing one SP with homogeneous resources and Case II analyzing one SP with heterogeneous resources). A further investigation into additional simulations might provide further useful information regarding in context of virtualization. A hypothetical Case III could consider the impact of an additional SP with similar but separate demands to the first. Further, a hypothetical Case IV could consider several SPs,

each with distinct demands informed by possible services that could be provided in this context. These two simulation cases are of specific interest due to the context virtualization this architecture and work was built within.

Improved Normalized Link Capacity Modeling

A major assumption made in Chapter 2 is that u_{ms}, the normalized capacity of the link between BS $s \in \mathcal{S}$ to the demand at point $m \in \mathcal{M}$, is binary. The definition provided for the capability of a nuanced u_{ms}, however the simplification ensures that Ω is finite and simplified the definition of the DEP and sDEP of Section 3.1. However, by forcing u_{ms} to be binary, demand points are either capable of a perfect link with a BS that it is in range of, or incapable of any communication. This is inherently unrealistic and a weakness of the model.

Implementing other definitions of u_{ms}, such as a simple linear trend from full connectivity to null connectivity or other models that take into account real world limitations of these links, could prove valuable in simulation.

DEP Sample Average Approximation Estimation Analysis

The sDEP formulation removes the large (or infinite) set of scenarios, Ω, and replaces it with a set of scenarios sampled from Ω, $\hat{\Omega}$. As this is a sampling process, there is a loss of information; the sDEP optimizes not for Ω, but for $\hat{\Omega}$. With enough scenarios and demand

points within those scenarios, $\hat{\Omega}$ becomes a viable approximation of Ω. In the simulations of Chapter 4, values of M and O were selected to force the sDEP to be close approximation of the DEP. For the preliminary simulations this likely underestimated the required values, as indicated by the noticeable loss of demand satisfaction due to increasing the size and distribution of demand in the scenarios not in $\hat{\Omega}$ and the sDEP approach being outperformed by the GA approach. Further, the required values were likely overestimated for Cases I and II, decreasing the run time performance of the sDEP approach.

In this context, the sDEP is called the sample average approximation (SAA) of the DEP. An analysis of the accuracy of the SAA estimator would provide insight to how many scenarios and demand points are necessary for the sDEP to provide an arbitrarily close approximation of the DEP and, by extension, the original two-stage stochastic optimization program. Further work considering this estimator would improve the understanding of the approach and could improve its run time performance.

Demand Point Adjustments

Each scenario contained a fixed number of demand points per scenario where the demand of the demand intensity field is homogeneously distributed to all demand points of the scenario. This means that there are no scenarios in Ω where there is more or less demand than what is indicated by the demand intensity function. Allowing for the number of demand points per scenario to vary for each scenario would allow each scenario to more properly reflect a PPP and more accurately reflect real world demand.

GA Improvements

The GA fitness calculation calls for an integration of the demand contained within each Voronoi cell of selected resources. During simulations, this is accomplished by performing a pixel-by-pixel sum of each pixel's demand. In practical applications, this may be the best that can be done, but since perfect knowledge of the demand intensity function (the SSLT model) exists in the simulations, it may be possible to properly integrate over a specified domain of the function. The primary obstacle to this is that the log-normal distribution and the SSLT model do not have a closed-form integral over a two dimensional region.

As implemented, the GA is single threaded while the sDEP is multithreaded. Both approaches were primarily compared in terms of CPU run time, eliminating this difference. Updating the GA implementation to be parallelized would allow the approaches to be compared in terms of wall clock time. The GA's largest time expense is computing the individual fitness; as each individual's fitness calculation is independent, this could be parallelized.

Other Traffic Demand Models

In this work, it was assumed that all cellular traffic demand was distributed according to a log-normal distribution, and specifically used a proposed model—the SSLT demand model—to generate realistic-looking demand. This was done because it has been shown that log-normal distributions can approximate real-world cellular network traffic demand [31]. However, further research has shown that traffic can also be approximated according to a

Weibull distribution or more accurately described by a log-normal mixture [32, 33]. Further, Zhou et al. [34] have shown that distributions of deployed BSs can be accurately approximated as an α-stable distribution. This latter distribution is possibly explained through the central limit theorem, as the sum of power-law (Paretian tailed) random variables, such as those describing traffic in telephone networks [46], will tend towards an α-stable distribution.

None of these distributions were considered within the context of this thesis as they added complexity for likely little gain. For instance, α-stable distributions do not generally have a closed form PDF, rendering use non-trivial, and log-normal mixtures require tuning distribution parameters[6] for each distribution in the mixture. Using some of these distributions for deploying demand or resources could provide additional insight.

Alternative Heuristic Approaches

The only heuristic approach I considered in this work was the genetic algorithm. Other heuristic algorithms may also be highly appropriate for this task, such as particle swarm optimization, simulated annealing, or some variety of deep learning techniques (see Section 1.3.2). Investigating these approaches could inform that another heuristic is more appropriate or effective at solving the task within the VNB.

[6] [32] has empircal results for a three-distribution log-normal mixture.

SP Specific α and β

Through α and β, the sDEP and GA approaches are controllable to target specific values of demand satisfaction, allowing the VNB to construct a less expensive VWN if less than 100% demand satisfaction is viable for the SPs. However, as implemented, both α and β impact all SPs considered by the VNB equally; each SP would be provided a VWN targeting the same amount of demand satisfaction. It could be highly beneficial in the context of virtualization to modify the original stochastic program and approach implementations to allow α and β to vary according to each SP. Doing so would allow for SPs of varying demand satisfaction needs to coexist within a single VNB.